GAUSSIAN MIXTURE MODEL BASED SYSTEM IDENTIFICATION AND CONTROL

By

JING LAN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY UNIVERSITY OF FLORIDA 2006
Copyright 2006

by

Jing Lan
To my family.
ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor, Dr. José C. Principe, for his continuous guidance, support, and help throughout the past five years of my Ph.D. study. Without his invaluable encouragement and patience, this work would have been impossible. He gave me the privilege to work in the Computational NeuroEngineering Lab and allowed me the freedom to explore the unknown world, and was always attentive and pertinent in critical moments. My deep appreciation is also extended to Dr. John G. Harris, Dr. John M. Shea, and Dr. Loc Vu-Quoc for their interests and participation on my supervisory committee. My deep recognition goes to Dr. Deniz Erdogmus and Dr. Mark A. Motter for their help and informative comments during my stay in CNEL. I also owe many thanks to all my colleagues in the CNEL group for their discussion of ideas and friendship.

My wife, Yun Zhu, has my deepest appreciation for her great love and unconditional support. Finally, I am grateful to my parents and my brother for their love.
TABLE OF CONTENTS

| ACKNOWLEDGMENTS                         | iv  |
| LIST OF TABLES                          | vii |
| LIST OF FIGURES                         | viii|
| ABSTRACT                                | x   |

CHAPTER

1 INTRODUCTION ........................................... 1
   1.1 Mission of System Identification and Control .......... 1
   1.2 Local Modeling with Improved Gaussian Mixture Model ...... 5
   1.3 Dissertation Outline .................................. 8

2 NONLINEAR DYNAMICAL SYSTEM MODELING .............. 10
   2.1 Nonlinear Dynamical Systems .......................... 10
   2.2 Time Embedding State Space Reconstruction .............. 11
   2.3 Global Dynamical System Modeling ....................... 13
      2.3.1 Polynomial Modeling ............................. 14
      2.3.2 Echo State Network .............................. 15
   2.4 Local Dynamic Modeling ................................ 20
   2.5 Summary .............................................. 22

3 GAUSSIAN MIXTURE MODELS AND LOCAL LINEAR MODELS ... 24
   3.1 Introduction ......................................... 24
   3.2 Maximum Likelihood Estimation Training – EM Algorithm .... 29
   3.3 Estimation of Local Linear Model ....................... 32
   3.4 Improvement on GMM ..................................... 36
   3.5 Comparison of the GMM with the SOM ..................... 38
   3.6 Summary ............................................... 41

4 GAUSSIAN MIXTURE MODEL BASED CONTROL SYSTEM ........ 42
   4.1 Introduction of Mixture Model Controller ............... 43
   4.2 Inverse Error Dynamic Controller Using GMM ................ 46
   4.3 PID Controller ........................................... 50
   4.4 Optimal Robust Controller .............................. 52
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3–1</td>
<td>Performance on 32-D LoFLYTE data</td>
<td>38</td>
</tr>
<tr>
<td>5–1</td>
<td>Performance on the Lorenz system</td>
<td>67</td>
</tr>
<tr>
<td>5–2</td>
<td>Performance on the Duffing oscillator</td>
<td>71</td>
</tr>
<tr>
<td>5–3</td>
<td>Performance on the linear mass system</td>
<td>76</td>
</tr>
<tr>
<td>5–4</td>
<td>Performance on the missile model</td>
<td>80</td>
</tr>
<tr>
<td>5–5</td>
<td>Control performance comparison on missile system considering noise</td>
<td>84</td>
</tr>
<tr>
<td>5–6</td>
<td>Performance on the LoFLYTE simulation data</td>
<td>91</td>
</tr>
<tr>
<td>5–7</td>
<td>Control performance comparison on the LoFlyte system considering noise</td>
<td>92</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–1</td>
<td>1-D function estimation by GMM-LLM.</td>
<td>5</td>
</tr>
<tr>
<td>2–1</td>
<td>Block diagram of the ESN.</td>
<td>17</td>
</tr>
<tr>
<td>2–2</td>
<td>Distribution of ESN internal weights with maximum spectral radius being 0.9, with unit circle as a reference.</td>
<td>19</td>
</tr>
<tr>
<td>3–1</td>
<td>Structure of GMM with LLM.</td>
<td>26</td>
</tr>
<tr>
<td>3–2</td>
<td>Explanation of G-SOM algorithm.</td>
<td>37</td>
</tr>
<tr>
<td>3–3</td>
<td>Soft combination region using GMM (between dash line)</td>
<td>40</td>
</tr>
<tr>
<td>4–1</td>
<td>Structure of Control System based on GMM-LLM</td>
<td>43</td>
</tr>
<tr>
<td>4–2</td>
<td>T-S fuzzy logic membership function</td>
<td>45</td>
</tr>
<tr>
<td>4–3</td>
<td>Structure of Inverse Control System</td>
<td>46</td>
</tr>
<tr>
<td>4–4</td>
<td>Pole-placement PID control system</td>
<td>51</td>
</tr>
<tr>
<td>4–5</td>
<td>Optimal robust control system schematic diagram</td>
<td>59</td>
</tr>
<tr>
<td>5–1</td>
<td>Dynamics of the Lorenz system</td>
<td>64</td>
</tr>
<tr>
<td>5–2</td>
<td>System identification performance on the “x” of the Lorenz system</td>
<td>66</td>
</tr>
<tr>
<td>5–3</td>
<td>Distribution of the Lorenz system (dash line) and GMM weights (star)</td>
<td>67</td>
</tr>
<tr>
<td>5–4</td>
<td>Weights change of 8 GMM kernels;</td>
<td>68</td>
</tr>
<tr>
<td>5–5</td>
<td>Performance comparison between SOM and GMM</td>
<td>69</td>
</tr>
<tr>
<td>5–6</td>
<td>Compensation performance</td>
<td>69</td>
</tr>
<tr>
<td>5–7</td>
<td>Dynamics of the Duffing oscillator as an autonomous system</td>
<td>71</td>
</tr>
<tr>
<td>5–8</td>
<td>System identification performance on the Duffing system</td>
<td>72</td>
</tr>
<tr>
<td>5–9</td>
<td>Sample of weights change of 8 GMM kernels;</td>
<td>73</td>
</tr>
<tr>
<td>5–10</td>
<td>Control performance of different controllers on the Duffing system</td>
<td>74</td>
</tr>
<tr>
<td>5–11</td>
<td>Dynamics of the mass system</td>
<td>76</td>
</tr>
</tbody>
</table>
5–12 Sample of system identification performance on the Mass system.

5–13 Sample of weights change of 6 GMM kernels for mass system.

5–14 Set-point control performance on the mass system.

5–15 Missile system dynamics.

5–16 Sample of system identification performance on the missile system.

5–17 Sample of weights change of 8 GMM kernels for missile system.

5–18 Performance of PID controller with different modeling approaches.

5–19 GMM-LLM based control performance on the missile system.

5–20 ESN control performance on the missile system.

5–21 General description of an aircraft.

5–22 Dynamics of the LoFLYTE system.

5–23 System Identification performance on the LoFLYTE system.

5–24 Sample of weights change of GMM kernels for the LoFLYTE system.

5–25 Control performance on the LoFLYTE system (set-point, noise free).

5–26 Control performance on the LoFLYTE system (tracking, noise free).

5–27 Robust control performance on the LoFLYTE system (set-point, 30 dB SNR noise).

5–28 Robust control performance on the LoFLYTE system (tracking, 30 dB SNR noise).

A–1 Distribution of improved ESN internal weights with maximum spectral radius is 0.9, with unit circle as a reference.

B–1 Demonstration system with ESN reference model and controller.

B–2 Schematic diagram of control system with ESN as controller based on Gaussian models.
Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

GAUSSIAN MIXTURE MODEL BASED SYSTEM IDENTIFICATION AND
CONTROL

By

Jing Lan

August 2006

Chair: José C. Principe
Major Department: Electrical and Computer Engineering

In this dissertation, we present a methodology of combining an improved Gaussian
mixture models (GMM) with local linear models (LLM) for dynamical system identi-
fication and robust predictive model control. In order to understand the advantage of
the mixture model, its structure and training are discussed in detail. A growing self-
organizing map is utilized to improve the random initialization of mixture models,
which makes the GMM convergence more stable. To increase local modeling capa-
bility and decrease modeling error, local linear models are trained based on GMM as
one-step predictors. Following the local modeling approach, a series of controllers are
designed to realize a tracking application, among which the optimal robust control
shows better robustness over other controllers. Five application systems with differ-
ent dynamics are simulated in order to verify the modeling and control capability of
the improved Gaussian mixture model. Through experiments and comparison with
self-organizing maps, radial basis functions, and other methodologies, it is shown that
the improved GMM-LLM approach is a more flexible modeling approach with higher
computation efficiency than its competitors. The Gaussian model algorithm shares
with the self-organizing maps the ease of robust controllers design.
CHAPTER 1
INTRODUCTION

1.1 Mission of System Identification and Control

System theory, which has evolved into a powerful scientific discipline of wide applicability, deals with the analysis and synthesis of dynamical systems [1]. In particular, the analysis of linear identification and control for nonlinear dynamical systems has been studied by many researchers leading to a better understanding of various mechanisms which includes observability, controllability, stability, and robustness [1, 2, 3]. The best developed aspect of the theory treats systems defined by linear operators using well established techniques based on linear algebra, and the theory of linear differential equations. Nevertheless, academic research has been concentrating around problems involving the identification, stabilization, and control of nonlinear dynamical systems in the last several decades. These efforts have now matured into a broad group theories on nonlinear dynamical systems applications. Initial efforts in this area pursued parametric approached, inspired by the established linear system theory, in which the dynamical equations of the system are assumed to be known from physical principles, with some uncertainty in certain parameters. In this framework, the system identification and system control problems are decoupled. And therefore they can be solved sequentially. Recently, adaptive system identification and control methodologies have been investigated too, which leads to a good understanding of the adaptation in linear systems and a satisfactorily general insight to adaptation in nonlinear control systems [4, 5, 6, 7].
Control theory deals with the system manipulation of the behavior of dynamical systems to satisfy certain desired outputs constraints [8, 9]. A classical parametric design procedure follows the system modeling (identification) and controller selection stages. In the case of traditional identification based on model derived from physical principles, the data are used to estimate the unknown parameters, with physically motivated system, whereas modern approaches stemming from the advances in neural networks modeling introduce black-box function approximation schemes [4]. The neural network modeling capabilities may be further enhanced using multiple sub-models in the context of switching between multiple adaptive models to achieve close-loop system that enhance transient behavior and cope with modeling uncertainties better than single model. Following the system identification stage, depending on the chosen modeling approach, the controller is designed typically using classical techniques based on linear system theory as well as classical or neural-networks-based nonlinear techniques [1, 10].

Modeling techniques can be classified into global and local. Global modeling uses a single parametric system and fits it all over the state space. The global modeling methods estimate the dynamics of system in one framework. Using single kernel or multiple kernels, the model is trained with the whole training data set. The most distinguishing feature for global modeling is that it is trained with all the data points, and in the system identification phase, all kernels will take effect on every step of prediction. Good examples of global modeling are the multi-layer perceptron (MLP) and the radial-basis-function (RBF), even though the last one uses local kernels. When estimating the distribution in the input space, each kernel in the RBF’s hidden layer accepts all the data from the training data set. Recurrent neural networks
(RNN) are another kind of powerful computational structure for global modeling [3]. But because of their complex inner structure, efficient training algorithms are necessary to achieve successful learning for the given task. Another difficulty for global modeling is when dynamical system characteristics considerably vary over the operating regime, effectively bringing the issue of time varying nonlinear parameters into the model design. Because of the complexity and possibly time-variability of the nonlinear dynamical system, global modeling and designing corresponding controllers is a difficult and time-consuming task.

Local modeling, which utilizes a divide-and-conquer approach to solve complicated problems, breaks the fitting of the system into smaller and easier pieces which can be managed by simpler topologies. Using multiple kernels, the local modeling is trained in a way that a kernel only gets training information from those data which are located within the region that kernel defined. In system training phase, every input data is located first, and only the “winner” kernel will later take effect on identification. To better understand this point, let us take the self-organizing map (SOM) [11] as an example with its winner-takes-all criterion. Therefore the local modeling technique can be regarded as a piece-wise modeling approach, and the pieces can be grouped together when it is necessary to approximate global dynamics. Specifically when each model is linear, the resulting model is a piece-wise linear dynamical approximation to the globally nonlinear dynamical system. The advantages of such a partitioning approach are the following:

1. System identification complexity can be reduced significantly due to the scaling down of the estimation problem from one large task to multiple small and simple task;
2. the local models can easily scale up to encompass more volume in the state space by adding new models as new data portion is acquired, which is especially of benefit when training data cannot cover the whole state space;

3. the design of a control system for the nonlinear system can be reduced to the design of multiple simple local controllers among which cooperation is possible to generate a single controller to the current plant.

The Gaussian mixture model with local linear models (GMM-LLM) is a local modeling approach, yet GMM training basically follows the global modeling training criterion because of the distributing property of Gaussian density function. In the GMM approach, the model is generated by a multi-dimensional joint mixture of Gaussian distributions [12]. In other words, the distribution of state vectors is described by a soft combination of several Gaussian distributions. And those combination coefficients will be determined according to the Gaussian posterior probability of current state vector giving certain Gaussian distribution. Following the Gaussian posterior probability property, every Gaussian is trained by all the training data set. Yet due to the nature of the nonlinear Gaussian distribution function, the weights will be large when the data is close to the centers of some Gaussian, which make the Gaussian kernel’s training emphasize more on their neighbor data than the data far away. In the prediction part, it becomes clear that Gaussian model is taking effect locally. The benefit from this nature is that the GMM is globally well trained, and local dynamics are addressed as well. The latter is also the reason why we call the approach local modeling.

The problem of controller design can be further eased with the selection of local modeling. In this case, the global nonlinear controller can be transformed into multiple piece-wise linear controllers for nonlinear systems, for which many strong design tools are available [8, 9, 13, 14].
1.2 Local Modeling with Improved Gaussian Mixture Model

The framework proposed herein is an improved GMM with local linear model, which is shown in figure 1–1. The GMM uses the Gaussian family to reconstruct the distribution of the state space vectors applying a soft-combination instead of a winner-takes-all criterion. Each Gaussian component represents a certain characteristic into a specific location of system dynamic space. Since the Gaussian distribution needs only two parameters, mean and covariance, the computation efficiency can be largely increased in GMM training. Because of this modeling property, GMMs have been widely applied in speech recovery, spectral analysis, etc. Schoner applied GMM for function approximation using supervised training [15]. Initializing the Gaussian kernels’ weights from growing self-organizing maps (GSOM), a more stable global maximum convergence is achieved.
Local modeling is usually based on the *Nearest-neighbors* criterion. A clustering algorithm divides the training data set into a group of smaller sets based on the Euclidean distance or other similar criteria, and a model is trained corresponding to each data set. As Farmer and Sidorowich [16] have already shown, local linear models provide an effective and accurate approximation despite their simplicity. Local linear models have shown very good performance in comparative studies on time series prediction problems and in many cases have generated more accurate prediction than global approaches [16, 17].

Building local mappings in reconstructed state space is a time and memory consuming process even though the performance can be satisfactory. A simplified approach is to quantize the state space and build local mappings only in positions where prototype vectors are located. The improvement utilized in the Self-Organizing Maps (SOM) has been applied by Principe et al. to model autonomous and non-autonomous systems [18, 19]. Motter also utilized SOM based modeling to the design of switching controllers [20].

One possible problem the SOM-based modeling cannot avoid is the quantization error in the modeling phase, due to the Euclidean distance between the prototype vector and the data closest to the vector. The SOM-based approach assumes the prototype vectors can represent the distribution of the whole state space and the distance is negligible, yet that is true only when the amount of prototype vectors is very large. Meanwhile, large amount of prototype vectors will lead to further requirements on computational resource and training time.

The structure of GMM consists of a set of Gaussian models and a gating function. The weighting coefficient for a certain data point \( \vec{x} \) corresponding to the Gaussian
kernels is described by the gating function. As the distance between the state and kernels determines the gating function value, it makes modeling error decrease and produces smooth switching compared with the SOM method.

The parameters of the Gaussian mixture are initially unknown, and will be estimated based on incomplete data through the Growing-SOM method. This implies that parameters are updated in a cooperative rather than competitive way. The procedure is an updated weighting of the contribution of all data points given a particular Gaussian kernel. The training of GMM is based on maximum likelihood criterion through the expectation-maximization (EM) algorithm. Section 3.2 discusses the EM training algorithm in depth. Following the Gaussian distribution density function, one local linear model is generated from local data for each Gaussian kernel. Thanks to the gating rule, the mixture of Gaussians framework is able to continuously vary the infrastructure of each piece-wise linear local model.

When the GMM-LLM is compared with RBF, the advantage of GMM-LLM is the modeling of local dynamics. Even though both the GMM-LLM and RBF use a linear combination in the final stage, the mechanism of how they describe the local dynamics is different. RBF modeling uses a single fixed weighted combination of a group of nonlinear outputs from Gaussian kernels, the centers of the Gaussians are adapted, but not the covariance. Although the GMM-LLM uses a linear model to represent the local system dynamics, which makes the output changes in a linear manner instead of a nonlinear way. Several such local models exist distributed throughout the state space, and they are weighted by the gating function to increase performance.
1.3 Dissertation Outline

In this dissertation, we will setup a control framework for the problem of modeling and control of nonlinear dynamical systems. We present a modified mixture Gaussian modeling-based system identification and control framework. GMM, as a piece-wise linear predictive model, is trained to estimate the distribution of state space using the Expectation-Maximization algorithm in order to satisfy the maximum likelihood criterion. Following the mixture Gaussian distribution, local linear models (LLM) are generated based on Mean-Square-Error (MSE) criterion to make accurate one-step prediction, which is obtained by soft-combinational function evaluation. Under the assumption that the state dynamics are smooth and sufficient data samples are available, GMM can provide a good and efficient prediction performance compared with other competitive methodologies.

Based on the mixture Gaussian reference model, we develop a set of controllers for non-autonomous and nonlinear plants. Starting from the EM training, the system model can be represented by GMM-LLM and the modeling error is bounded. Several controller such as dynamic inverse and pole-placement PID controllers, optimal robust controller, echo state neural controller are realized and employed using the mixture model for several systems with different dynamics. We emphasize the optimal controller and the echo state neural controller. We will show that the optimal controller exhibits excellent robustness against model uncertainty and external disturbance, and the ESN controller has better generalization capability.

The dissertation is divided into five parts. In Chapter 2, some general description and definitions of nonlinear dynamic systems and local modeling, time-embedding delay system reconstruction, and dynamic system identification are given. The criterion
to select embedding dimension will be discussed. Echo state network, a new kind of recurrent neural networks, is present for the later use on controller design.

In Chapter 3, GMM training methodology analysis is presented, which also includes improved GMM training through GSOM initialization, the generation of local linear models (LLM), and the comparison of GMM-based approach to SOM-based approach in order to address the computational efficiency of Gaussian modeling.

In Chapter 4, the descriptions for a set of controller design approaches based on GMM, the controllers' mathematical backgrounds, and their stability issue are discussed in detail.

Chapter 5 gives application examples on a set of real system model, and main results of GMM-based applications are presented with comparisons and discussed in details. For different modeling methods, the modeling performance will be compared based on signal to error ratio criterion. For controllers, the control performance will be compared based on criteria of rising (falling) time, settling time, overshot, and the covariance of control error.

Chapter 6 gives conclusion, contribution and briefly summarizes future work.
CHAPTER 2
NONLINEAR DYNAMICAL SYSTEM MODELING

2.1 Nonlinear Dynamical Systems

General linear and nonlinear system theories provide tools which make it possible to fully estimate a nonlinear dynamic system from observation under the assumption that noise is zero or negligible. The enabling theoretical insight is the time delay embedding theorem, which will be briefly reviewed along with some important system-characterization methodology. Reconstruction of the physical state space is explicitly or implicitly fundamental to the prediction and characterization applications in chapter 5.

Dynamics describe how one system state develops into another state over the course of time. Technically, a dynamical system is a smooth function of the reals or the integers on another object (usually a manifold). “When the reals are acting, the system is called a continuous dynamical system, and when the integers are acting, the system is called a discrete dynamical system” [21]. A dynamical system expressed as a discrete time evolution rule can be described from the observation of its outputs. The dynamic of the system in state space can be defined as

\[
x(t + 1) = f(x(t))
\]

\[
y(t) = h(x(t))
\]

where \( x(t + 1) \) is the next step state for discrete-time systems. \( x(t) \) are the system states, \( f(\cdot) \) is a nonlinear transfer function and is usually referred as the vector field,
and $h(\cdot)$ is the output function. If the vector field $f(\cdot)$ is a linear function of the system states, the underlying system is linear; otherwise, the system is non-linear. It is also assumed that the systems here are deterministic. Thus the transition from the system’s state $x(t_1)$ at time $t_1$ to the state $x(t_2)$ at time $t_2$ is governed by deterministic rules.

2.2 Time Embedding State Space Reconstruction

Equation (2·1) describes a system that follows the internal governing equations $f(\cdot)$ given a certain initial condition. And the system’s dynamic is fully characterized by its manifold encoded in the function $f(\cdot)$. Because of Takens Theorem, transforming the time series into a state space form will result in a dynamical equivalent to the original space [22]. Since the mathematical description of $f(\cdot)$ is unavailable for most real-world systems, with the formulation of the time embedding theorem, Takens presented a methodology to recover an equivalent description of $f$ from observation of the system:

**Theorem 1.** Let $M$ be a compact manifold of dimension $m$. For pairs $(\phi, y)$, $\phi : M \to M$ a smooth diffeomorphism and $y : M \to R$ a smooth function, it is a generic property that the map $\Phi_{\phi,y} : M \to R^{2m+1}$, defined by

$$\Phi_{\phi,y}(x) = (y(x), y(\phi(x)), \ldots, y(\phi^{2m}(x))) \quad (2·2)$$

is an embedding; by “smooth” we mean at least $C^2$.

There are several methods for estimating the optimal embedding dimension. Buzug and Pfister designed two methods called fill factor and local deformation [23]. The first is a procedure that yields a global measure of phase-space utilization for
(quasi) periodic and strange attractors and leads to a maximum separation of trajectories within the phase space. The second describes the local dynamical behavior of points on the attractor and gives a measure of homogeneity of the local flow. Gao and Zheng proposed the local exponential divergence plot method for a chaotic time series, based on which the embedding dimension, delay time, and the largest Lyapunov exponent are estimated [24].

We determine the minimum time embedding dimension using Lipschitz quotient [25, 26]. If the function is assumed to depend on \( n - 1 \) inputs and it actually depends on \( n \) inputs, the data set may contain several points that are very close in the space spanned by the \( n - 1 \) inputs but differ significantly in the \( nth \) input. Assuming the system function is smooth, if the \((n - 1)\)-D inputs are close, it is expected that the outputs from them are also close. If one or several relevant inputs are missing, outputs corresponding to different input will take different values. In this case, it can be concluded that \((n - 1)\)-D input is not long enough.

The Lipschitz quotients in 1-D case are defined as

\[ l_{ij} = \left| \frac{y(i) - y(j)}{\phi(i) - \phi(j)} \right|, \quad i, j \in \mathbb{R}^N, \quad i \neq j \tag{2–3} \]

where \( N \) is the number of samples, \( \phi \) is input. For the multidimensional case, equation is extended as

\[ l^n_{ij} = \frac{|y(i) - y(j)|}{\sqrt{(\phi_1(i) - \phi_1(j))^2 + \cdots + (\phi_n(i) - \phi_n(j))^2}}, \quad i, j \in \mathbb{R}^N, \quad i \neq j \tag{2–4} \]

where \( n \) is the number of input.

The Lipschitz index can be defined as the maximum occurring Lipschitz quotient

\[ l^n = \max_{i \neq j} (l^n_{ij}) \tag{2–5} \]

As discussed before, if \( n \) is not big enough, the Lipschitz index will be large. If all relevant inputs are included, the Lipschitz index will stay near constant.

### 2.3 Global Dynamical System Modeling

System identification is a technique to build mathematical models of dynamics based on input-output measurements. An important property for system identification is its capability of making short term prediction of system dynamics. Normally, the nonlinear dynamical behavior of real-world systems complicates the construction of models that can accurately describe dynamics. Moreover, the knowledge about the inner structure of systems is often unavailable. Under this situation, the modeling training can only be driven through a finite collection of system’s input and output measurement. This kind of identification is referred as *black box* approach.

Several approaches have been studied for global modeling. One representative of global nonlinear modeling is the time delay neural network (TDNN) which was first developed by Weibel and Lang in 1987 [27]. Using time delay segment of state as input data, the TDNN is structurally close related to the multi-layer perceptron (MLP). The TDNN has been shown to be a universal mapper in multi dimensional functional spaces with finite memory (especially myopic functional space) [28]. Another method is the polynomial modeling. Polynomial modeling is applied as the basis for the realization of nonlinear mapping. Fuhrmann uses it in geometric control theory [29]. Recurrent neural networks (RNN) represents a kind of nonlinear modeling methodology. Researchers use RNN when regular mathematical equations cannot generate satisfying results [30, 31].

In the following sections, we will introduce polynomial modeling briefly as we will use its first order expansion as a linear model in chapter 3. And we will discuss
the echo state network (ESN) with modified design method as a new global modeling approach.

2.3.1 Polynomial Modeling

Taking the single-input-single-output (SISO) system as an example, the linear input-output polynomial model is the autoregressive moving average with exogenous input (ARX) model

\[ \tilde{y}(n) = b_1 u(n - 1) + \cdots + b_D u(n - D) - a_1 y(n - 1) - \cdots - a_D y(n - D) \quad (2-6) \]

The polynomial modeling can be extended to a nonlinear ARMAX (NARMAX) model by replacing the linear mapper in (2-6) with nonlinear function \( f(\cdot) \)

\[ \tilde{y}(n) = f(u(n - 1), \ldots, u(n - D), y(n - 1), \ldots, y(n - D)) \quad (2-7) \]

A more straightforward way to utilize polynomials for nonlinear system identification is to apply a polynomial for approximation of the one-step prediction function (2-7) with low order expansion, which is called Kolmogorov-Gabor polynomial models. And the benefit from polynomial model is that all the nonlinear coefficients calculations are avoided. A simplified version of the Kolmogorov-Gabor polynomial model uses less high-order information, also called Volterra Series model, describes the nonlinearity only for the input

\[ \tilde{y}(n) = f[u(n - 1), \ldots, u(n - D)] - a_1 y(n - 1) - \cdots - a_D y(n - D) \quad (2-8) \]

Thus the second order second degree polynomial function can be expressed as

\[ y(n) = \theta_1 + \theta_2 u(n - 1) + \theta_3 u(n - 2) + \theta_4 y(n - 1) + \theta_5 y(n - 2) + \theta_6 u^2(n - 1) + \theta_7 u^2(n - 2) + \theta_8 u(n - 1)u(n - 2) \quad (2-9) \]
With this simplification the number of regressors is reduced, thus the computation of prediction is simplified. The price paid here is the necessary increase of order $D$ in order to describe the nonlinearity of the original system. For linear model, there will be no high order expansion, and the embedding length will be increased when necessary.

2.3.2 Echo State Network

Under the condition that the system model is treated as a black-box, time embedding input-output data will incur some technical problems besides its advantage of simpleness. One of them is the need for large memory to access the system’s history, another is the delayed system response. The Echo State Network (ESN) gives us an alternative for system modeling, which does not need time embedding data and can supply instant system response [32, 33, 34]. Comparing with other nonlinear neural networks (recurrent networks especially), the ESN has less amount of parameters need to be trained. Moreover, the parameters are linear and they can be trained through the Wiener-Hopf approach [35].

Heterogeneous ordered networks or, more specifically, recurrent neural networks (RNN) are convenient and flexible computational structures applicable to a broad spectrum of problems in system modeling and control. There are many ways to make this statement more precise. For instance, RNNs can be cast as representations of the vector field of a differential systems, or be trained to embody some desired attractor dynamics [31]. In those applications, systems without input (autonomous) are modeled. Since we wish to model input-driven systems, we adopt a standard perspective of system theory and view a deterministic, discrete-time dynamical system
as a function, which yields the next system output as a prediction, given the input and output history [36].

As a special case of RNN, the ESN was first proposed by Jaeger [36, 37]. The ESNs have interconnected and recurrent topology of nonlinear PEs which constitute a “reservoir of rich dynamics”. The interesting property of ESN is that only the memoryless readout of ESN needs training, whereas the internal recurrent topology has fixed connection. This key property dramatically reduces the complexity of RNN training to simple linear regression while preserving some of the computational power of RNN.

We consider the recurrent discrete-time neural network as given in figure 2–1 with $M$ input units, $N$ processing units, and $L$ output units. The value of the input unit at time $n$ is $\mathbf{X}(n) = [X_1(n), X_2(n), \cdots, X_M(n)]^T$, that of internal units is $\mathbf{s}(n) = [s_1(n), s_2(n), \cdots, s_N(n)]^T$, and that of output units is $\mathbf{y}(n) = [y_1(n), y_2(n), \cdots, y_L(n)]^T$. The system transfer function of ESN can be expressed as

$$
\begin{align*}
\mathbf{s}(n+1) &= f(W_{in} \mathbf{X}(n+1) + W \mathbf{s}(n) + W_{back} \mathbf{y}(n)) \\
\mathbf{y}(n+1) &= f_{out}(W_{out} \cdot \mathbf{s}(n+1))
\end{align*}
$$

(2–10)

where, for most of the case, nonlinear output function $f_{out}(\cdot)$ is eliminated, and only linear output is used.

The critical difference between ESN and RNN lies in the adaptation of the recurrent weights. ESN compensates for not training the recurrent connection weights by increasing the dimensionality of the hidden layer. In this way, ESN creates a diversity of dynamics with a large number of PEs in its reservoir and the readout is simply trained to pick up dynamical components required for a given desired output signal. Hence, it is crucial for the operation of the ESN to include a reservoir with “enough
richness” of dynamics. And in most engineering applications of ESN, e.g. system identification, the optimal linear mapper is obtained through recursive algorithms which minimize mean square error (MSE) or higher order error momentum.

In the ESN, the echo states form basis functions that are a nonlinearly filtered version of the input signals [35, 38]. The linear independency of the echo states is almost always satisfied because of the nonlinear PEs except rare pathological cases when \( W \) and \( W_{in} \) has symmetrical entries with respect to any two PEs in the recurrent network. Note that, the echo states are not orthogonal to each other since there are connections between PEs making the states correlated. With this interpretation, all what the linear read-out is doing is to combine the basis functions derived from the input to create a given desired signal. The span of the ESN can be defined as the set of all functions defined by (2–10), where \( f_{out} \) is linear, for all values of \( W_{out} \). Here, the critical question is how big the span of the echo states is. This is directly related to the selection of the weights of the ESN reservoir. Currently, there are two ways

Figure 2–1. Block diagram of the ESN.
to generate $W$. The first method selects $W$ randomly from a sparse matrix with the help of the echo state condition $\|W\| < 1$ [36, 37], where $\|\cdot\|$ is the spectral radius (SR). For the second method, the $W$ is selected such that the maximum entropy is achieved [35]. The echo state condition is a stability condition that guarantees that the states are only controlled by the input whereas the sparseness is required to avoid much correlation between the states by minimizing the connections among them.

Assuming unknown dynamical system transfer function is $g(\cdot)$, the objective of system identification is to generate an estimation function $\tilde{g}(\cdot)$ such that the error prediction is small,

$$\|y_{k+1} - \tilde{y}_{k+1}\| = \|g(y_k, u_k) - \tilde{g}(y_k, u_k)\| \leq \varepsilon \quad (2-11)$$

Let the linear mapper (assume a single output) be denoted by the weight vector $W_{out}$. Considering the cost criterion is MSE, the readout vector can be easily computed through Wiener-Hopf function as

$$W_{out} = R^{-1} \cdot P \quad (2-12)$$

where $R = E[s^T \cdot s]$, $P = E[s^T \cdot d]$, and $s$ is ESN internal states trained from input sequence, $d$ is corresponding desire sequence from training data.

We now present a more general formulation of the training procedure for the case with no input feed forward and output feedback.

**Task.** Given a training I/O time series $(u_{train}(n), y_{train}(n))_{n=1, \ldots, n_{max}}$, where the inputs come from a compact set $U$. We want a RNN whose output $y(n)$ approximates $y_{test}(n)$. 
Figure 2–2. Distribution of ESN internal weights with maximum spectral radius being 0.9, with unit circle as a reference.

1. Procure an ESN. Construct a RNN that has echo states in an admissible state set $A$ with respect to input. (i). setup normalized random input weights since it shows that echo state property is independent of input weights; (ii). setup sigmoid network whose weight matrix $W$’s spectral radius satisfies $|\lambda_{max}| < 1$. Experience shows that the weight matrix with maximum entropy has the best system identification performance in the sense of MSE. The sample result is shown in 2–2.

2. Run network with training, dismiss initial transient. Start with an arbitrary networks state $s(0)$, and feed input to ESN through (2–10) to obtain internal states $ss$. Then dismiss from this run a suitably long initial transient period.

3. Compute output weights which minimize the training error. Use back-propagation or any favorite linear regression algorithm to train the readout weight matrix. With these weights, the network is ready for use.
2.4 Local Dynamic Modeling

Dynamic modeling seeks to qualify the system that created the time series. Following the discussion in chapter 1, global modeling utilizes the whole training data set to describe the whole system dynamics. Yet the problem is that the global modeling needs large amount of memory, and every data point will take effect on all the memory. Consequently, global models will show some difficulties to bring the time-varying issue into the design when system characteristics change considerably over the operating regime.

Local modeling approaches, on the other way, divide the dynamic space into several parts. Each local model is trained fitting one part of the data from the state space. The local linear approximation can be considered a simplified alternative to nonlinear modeling. The approach of locally linear ARX has been widely discussed in the time series literature. It is a state-dependent sequential type of recursive algorithm for identifying dynamic models using simple linear regression or interpolation [2]. Under the assumption that \( f(\cdot) \) is smooth in the vicinity of \( x(n) \), \( f(\cdot) \) can be approximated by the first order term of its Taylor series expansion,

\[
\begin{align*}
\tilde{f}(x(n)) & \approx \sum_{n} \hat{a}_n^T x(n) + \hat{b}_n \\
& = \tilde{a}^T x(n) + \tilde{b}_n
\end{align*}
\]

where \( \tilde{a}_n \) is the Jacobian of \( f \) at \( x(n) \), and \( \tilde{b}_n \) is a constant vector at \( x(n) \). The implicit assumption of the method here is that the time-steps for state dynamics are small. The parameters \((\tilde{a}, \tilde{b})\) can be estimated for the local linear mapping in the least square sense from the local data samples. Casdagli et al. stated that local modeling approaches can provided approximations of higher accuracy than global methods,
especially for systems with complex behavior. Casdagli also stressed that the great advantage of local modeling is the flexibility in building the model [39].

As the center mission of modeling, the predictive mapping \( f : \mathbb{R}^D \rightarrow \mathbb{R} \) can be expressed through time embedding as

\[
x(n + 1) = f(x(n))
\]

where \( f(\cdot) \) is a deterministic ARMA model with time-embedded input vector \( x(n) = [x(n), x(n - 1), \ldots, x(n - D + 1)] \). With the method of local modeling the state space is partitioned into small regions and a local model is established to describe the dynamics in different region. Thus local models vary from region to region.

The approach of locally ARMA models has been discussed in the time series literature by Priestley [40]. Improved approach was proposed by Farmer and Sidorowich [16], who also extended ARMA models to locally polynomial approximations of higher order. The first order expansion, which is a linear model, is an effective local modeling method. And the approximation with high-order polynomials are not significantly better than those obtained with first order. In our work, we will use the first order expansion.

In many local model applications, the self-organizing map (SOM) has been utilized to divide the operating region into small parts, where one local model is trained corresponding to one part of the state [5, 11, 18, 19]. The SOM is appropriate for switching nonlinear relationships of high-dimensional data into simple geometric relationships that preserve the topology in the state space [41]. Principe and Wang modeled a chaotic system with SOM-based local modeling method [19]. Cho
et al. realized competitive local controller using SOM-based local modeling approach [18, 20].

The role of GMM in system identification is to estimate the distribution of state space. According to the distribution density of each Gaussian kernel within GMM, the current state of the system is described by one or several Gaussian models. To finish the predictive modeling, a set of local models also trained based on GMM. Following the GMM’s criterion, one local model is linked to one Gaussian kernel. And these two make up the identification model pair. The detail of improved GMM construction and training and corresponding LLM training will be discussed later in chapter 3.

2.5 Summary

Treating dynamical system as a black box, polynomial modeling and time embedding data structure build the foundation for the application of GMM. The simplified polynomial expression of system dynamic makes linear modeling feasible. However, it also limits the application range of each single model especially when large modeling error is presented. In chapter 3, GMM will be use to estimate the application range of each local model, and a comparison will be made between GMM and SOM. In chapter 5, GMM-LLM based system identification and modeling will be applied to a series of applications and construct the basis of modeling predictive control (MPC).

From the nature of ESN, we can see one advantage of ESN is its less-training construction procedure even though it is not linear modeling any more. Since its internal states are randomly generated, the only part which needs training is its readout vector, and there is no training that is necessary for the states. And since the readout is of linear form, the training for the readout vector can be easily done
through Wiener-filter similar methods. Thus through the functionality of “internal states”, the ESN will require less training computation compare with other recurrent network. Another advantage of ESN is its input structure over time-embedding data structure. Since ESN has a “reservoir” of internal states and pre-defined feedback, these internal states can be deemed an “internal delay” to input for a certain system, ESN can record it yet time-embedding may need longer data structure. We will also show that the ESN with maximum entropy can accomplish system identification work with minimum MSE.
CHAPTER 3
GAUSSIAN MIXTURE MODELS AND LOCAL LINEAR MODELS

3.1 Introduction

The Gaussian Mixture Model (GMM), also called mixture of expert models, is a framework for unsupervised learning. GMM, along with local linear models (LLM), constitutes the infrastructure to solve one of the fundamental problems of our research, the system identification (SI) step. Conceptually, however, we cannot use GMM alone to solve the SI problem. SI, as the discipline of making mathematical models of unknown dynamical systems, must extract the structure in the joint space (input & desire). GMM, as an “information” clustering approach, estimates the distribution of data in the input space. GMM is similar to the Partition Algorithm (PA) (Hilborn & Lainiotis 1969; Lainiotis 1971), which includes a group of Gaussian kernel systems and a gating function [42], however it is based on a well defined statistical framework for density estimation. The GMM is also similar to the first layer of the Radial Basis Function (RBF) network used in nonlinear system approximation. In fact the kernels in the RBF are also Gaussian functions, which are spread over the data space using a clustering algorithm [42, 43]. Normally the covariance of these Gaussian kernels are not trained from the data. The RBF output layer combines these internal states linearly into a single projector that is trained with the desired response to extract the information in the joint space. The GMM-LLM approach combines these two basic strategies, by creating a local partition of the input space as in the PA algorithm, and employs multiple local linear models attached to each
partition, instead of nonlinear combining the intermediate nonlinear states as in the RBF.

Figure 3–1 depicts the elements of the GMM-LLM architecture. The input data is modeled by a mixture of Gaussian’s model, and the influence of each Gaussian is evaluated by a weight that represents the likelihood of the data being generated by the corresponding Gaussian mode. During training the data from each Gaussian cluster is used to train the LLMs. During testing, the input is directly sent to all the LLMs and their output weighted with the probability that the sample is generated by each model before being combined by the output adder. Note that during testing these weights change with the data providing a greater flexibility when compared with other local modeling alternatives. This architecture is principled using well established statistical methods, very flexible, and still very simple because it uses only linear models.

The GMM-LLM approach is conceptually an alternative to the Self Organizing Map (SOM)-LLM reviewed in chapter 2. The SOM clusters the system state space into a group of small regions using a winner-takes-all operator [42]. Each small input region, which is also called Voronoi tessellation, is associated to one linear model that is trained with the data from the tessellation and that attempts to approximate the desired response when instantiated. Globally the modeling equation could be described as

\[ f(x) = \sum_{k=1}^{K} a_k f_k(x), a_k = \begin{cases} 1, & x \in \text{cluster}_k; \\ 0, & \text{otherwise} \end{cases} \]  

(3–1)

Using soft-combination instead of winner-takes-all criterion, GMM does the local modeling in a cooperative manner. Even though it still follows the description of the
first part of (3–1), it doesn’t limit the coefficients to be “1” or “0”. Consequently, the GMM-LLM approach can reach similar SI performance with less PE compared with the SOM-LLM approach. Moreover, soft-combination can generate smooth-switching among different models, and can alleviate the difficulty faced by winner-takes-all, when it chooses the wrong winner, where the corresponding LLM will generate worse SI result.
Another competitor to the GMM-LLM is the fuzzy logic approach. The most popular fuzzy logic approach is the Takagi-Sugeno (T-S) fuzzy model [44] which is also a local modeling method. Similar to the GMM-LLM, T-S fuzzy model uses multiple linear models for SI and *soft-combination* of linear membership functions to weight the outputs from linear models. The T-S fuzzy model divides the state space into several overlapped regions using a set of simple “if-then” rules, one rule corresponds to one linear membership function. Besides the order of membership function, the difference between the T-S fuzzy model and the GMM-LLM lies in the training method. The T-S fuzzy model needs pre-knowledge about the system for both membership function training and linear model training. Unlike the unsupervised training in GMM, the T-S fuzzy model training is in a supervised manner which increases the complexity of modeling.

Assuming the training data set covers the state space of the target system, the state space could be described by the embedded (reconstructed) data in the appropriate dimensional space with an optimal delay. The goal is to use a set of Gaussian kernels to estimate the structure of the eventually complex data distribution in reconstructed space. Since the only necessary parameters for a Gaussian distribution are its mean and covariance which can save time and computation on model training. This simplicity is one reason why Gaussian density function is chosen as the kernel function.

Setting \( p(\vec{x}) \) as the distribution density of the current state \( \vec{x} \), \( p(\vec{x}) \) can be expanded in a sum over Gaussian clusters \( c_k \). All the Gaussian clusters cover the whole state space defined by \( (\mu_k, \Sigma_k) \). The probability density of each cluster reflects the
domain of influence of each cluster.

\[
p(\vec{x}) = \sum_{k=1}^{K} p(\vec{x}, c_k) = \sum_{k=1}^{K} p(c_k)p(\vec{x}|c_k) = \sum_{k=1}^{K} p(c_k)g(\vec{x}|c_k)
\] (3–2)

where \(g(\cdot)\) indicates that we explicitly state the form of the conditional Gaussian distribution densities

\[
g(\vec{x}|c_k) = \frac{1}{(2\pi)^{D/2}|\Sigma_k|^{1/2}} e^{-\frac{(\vec{x}-\mu_k)^T \Sigma_k^{-1}(\vec{x}-\mu_k)}{2}}
\] (3–3)

where \(\vec{x}\) is current state, \(\mu_k\) and \(\Sigma_k\) are mean vector and covariance matrix of cluster \(c_k\) in the feature space. Also the summation over all the clusters is normalized by the cluster probabilities \(p(c_k)\) to satisfy

\[
\sum_{k=1}^{K} p(c_k) = 1
\] (3–4)

If the elements of input \(\vec{x}\) are independent of each other or the cross correlation is negligibly small, the covariance matrix could be reduced to a diagonal matrix, which in turn simplifies the computation of Gaussian density function.

\[
g(\vec{x}|c_k) = \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi\sigma_d,k}} e^{-\frac{(x_d-\mu_{d,k})^2}{2\sigma_{d,k}^2}}
\] (3–5)

where \(D\) is the input dimension.

Function (3–3) and (3–5) show that GMM uses the linear combination of non-linear estimation models so as to build a powerful global system model. Meanwhile,
GMM describes the local data feature with relatively simple models which satisfy description of (3–2).

3.2 Maximum Likelihood Estimation Training – EM Algorithm

The goal of the GMM approach is to find an optimal representation for the distribution density $p(x)$ of each point while maintaining a low computational cost. In order to simplify the problem, the optimization criterion is set as the maximum log-likelihood function over the whole data set. Figueiredo et al. use non-parametric EM training approach for Gaussian mixtures [45]. We describe the maximum log-likelihood function as

$$L(X, \mu, \Sigma) = \sum_{n=1}^{N} \ln[p(x_n)]$$

(3–6)

$$= \sum_{n=1}^{N} \ln[\sum_{k=1}^{K} p(c_k)g(x_n|c_k)]$$

where $N$ and $K$ are the total amount of data and Gaussian clusters respectively.

Since the Gaussian model parameters are calculated from the observed data set and describe the local distribution of data points, a recursive search will be necessary to find the optimal Gaussian model parameters. During the search, the maximum likelihood function is used as an optimization criterion, so there will be two estimators combined in each joint update, one for the estimation of likelihood, and one for distribution of each Gaussian kernel. The Expectation-Maximization (EM) algorithm [46] is applied to search for the optimal parameters of the Gaussian kernels. The EM algorithm has gained wide appreciation as an efficient algorithm for probabilistic networks training. The EM assumes that the known states (the observed data) and the unknown states (the Gaussian parameters) characterize the whole system model.
As an unsupervised approach, the EM training based GMM only needs the number of Gaussian kernels \( K \). All other parameters are obtained through training.

The recursive process of EM algorithm starts with the **E-step** (Expectation calculation), during which the cluster parameters are assumed correct. Expectation of the log-likelihood (3–6) of the complete conditional probability by the observed samples is computed. According to Bayesian Theorem, the posterior probabilities are proportional to the likelihood function.

\[
p(c_k|\vec{x}) \propto p(\vec{x}|c_k) \equiv L(X, \mu, \Sigma)
\]

So equivalently, the posterior probabilities, which relate each Gaussian cluster to each data point in the conditional probability form \( p(c_k|\vec{x}) \), can be evaluated as

\[
p(c_k|\vec{x}) = \frac{p(\vec{x}|c_k)p(c_k)}{p(\vec{x})} = \frac{p(\vec{x}|c_k)p(c_k)}{\sum_{i=1}^{K} p(\vec{x}|c_i)p(c_i)}
\]

The sum in the dominator normalizes among the Gaussian clusters, and the numerator determines the largest possible density function for each data point. Thus the maximum likelihood of the whole data set will monotonically increase.

In the **M-step** (Maximization calculation), the distribution density function of each data point, which describes the likelihood of data given certain cluster, is temporarily fixed. The update improves the parameters of each Gaussian cluster: unconditional cluster probability \( p(c_k) \), mean vector and covariance matrix \( (\mu_k, \Sigma_k) \). Thus the likelihood will be maximized recursively. In equation (3–8), considering that the amount of data points are limited and the values are discrete in reality, the
integral is replaced by the sum over all training data.

\[ p(c_k) = \int p(c_k|x) p(x) dx \approx \frac{1}{N} \sum_{n=1}^{N} p(c_k|x_n) \]  

(3–8)

The mean vector and covariance matrix of each Gaussian cluster are updated according to the stationary points of the likelihood to the unknowns \((\mu_k, \Sigma_k)\). The derivative equation of likelihood to the mean is

\[ \frac{\partial}{\partial \mu_k} \sum_{n=1}^{N} \ln[p(x_n)] = 0 \]  

(3–9)

\[ \sum_{n=1}^{N} \frac{\partial}{\partial \mu_k} \ln[\sum_{i=1}^{K} p(c_i) g(x_n|c_i)] = 0 \]

\[ \sum_{n=1}^{N} \frac{\partial}{\partial \mu_k} \ln[\sum_{i=1}^{K} p(c_k) g(x_n|c_i)] = 0 \]

\[ \sum_{n=1}^{N} \frac{p(c_k) g(x_n|c_k) 2(x_n - \mu_k)}{2\Sigma_k} (-1) = 0 \]

The answer to update the mean will be obtained through maximum likelihood criterion:

\[ \sum_{n=1}^{N} p(c_k|x) \cdot \mu_k = \sum_{n=1}^{N} p(c_k|x) \cdot x_n \]  

(3–10)

\[ \mu_k = \frac{\sum_{n=1}^{N} \bar{x} \cdot p(c_k|x)}{\sum_{n=1}^{N} p(c_k|x)} \]
The same result is achieved through the integration method as:

\[
\mu_k = \int \vec{x} p(\vec{x}|c_k) d\vec{x} \quad (3-11)
\]

\[
= \int \vec{x} \cdot p(\vec{x}) \cdot \frac{p(c_k|\vec{x})}{p(c_k)} d\vec{x}
\]

\[
\approx \frac{1}{N \cdot p(c_k)} \sum_{n=1}^{N} \vec{x} \cdot p(c_k|\vec{x})
\]

\[
= \frac{\sum_{n=1}^{N} \vec{x} \cdot p(c_k|\vec{x})}{\sum_{n=1}^{N} p(c_k|\vec{x})}
\]

Similarly, the Gaussian cluster covariance is:

\[
\Sigma_{k,ij} = \int (\vec{x}_i - \mu_{k,i}) \cdot (\vec{x}_j - \mu_{k,j}) p(c_k|\vec{x}) d\vec{x} \quad (3-12)
\]

\[
\approx \frac{1}{N \cdot p(c_k)} \sum_{n=1}^{N} (\vec{x}_i - \mu_{k,i}) \cdot (\vec{x}_j - \mu_{k,j}) \cdot p(c_k|\vec{x})
\]

\[
= \frac{\sum_{n=1}^{N} (\vec{x}_i - \mu_{k,i}) \cdot (\vec{x}_j - \mu_{k,j}) \cdot p(c_k|\vec{x})}{\sum_{n=1}^{N} p(c_k|\vec{x})}
\]

The basic model estimation procedure can be summarized into five phases:

1. Pick initial conditions;
2. Estimate the data distribution density \( p(\vec{x}|c_k) \);
3. Estimate the posterior probability of each cluster for each data \( p(c_k|\vec{x}) \);
4. Estimate the cluster probability \( p(c_k) \), Gaussian cluster parameters mean \( \mu_k \) and covariance \( \Sigma_k \);
5. Go back to step (2) until the increment of likelihood function (3–6) is within certain range.

### 3.3 Estimation of Local Linear Model

Because GMM-LLM is a group of globally optimized local model based on distribution of training data set, it preserves topological properties of target system in the state space. Since the soft combination of Gaussian clusters can be used to weight the
estimation of the current system’s dynamic, the dynamic estimation can be extended to a one-step linear mapping \( \tilde{y}(n+1) = \tilde{f}(\tilde{x}(n)) = \tilde{L}_{opt}^T \cdot \tilde{x}_n \), where \( \tilde{f} \) stands for a certain reference model. And \( \tilde{L}_{opt} \) is the optimized current local model based on Gaussian gating function. LLM is the most simplified reference model of the real system and it will simplify the design procedure of controller [16]. From equations (3–2), (3–7), and (3–10)-(3–12), we thus can express one-step linear prediction function as

\[
\tilde{y}_{n+1} = \sum_{k=1}^{K} \frac{p(\tilde{x}_n|c_k)p(c_k)}{\sum_{i=1}^{K} p(\tilde{x}_n|c_i)p(c_i)} \cdot \tilde{L}_k^T \cdot \tilde{x}_n
\]  

(3–13)

where \( \tilde{L}_k \) is the \( k \)th local linear model (LLM) corresponding to Gaussian cluster \( k \), which has same dimension of \( \mu_k \) and \( \tilde{x} \). \( \tilde{y}_{n+1} \) is the estimation of \( y_{n+1} \). For simplicity, we denote \( p_{k,n} \) as posterior probability of cluster \( k \) corresponding to data \( n \).

Equation (3–13) can be switched to matrix form in order to simplify the calculation of local linear model \( \tilde{L}_ks \).

\[
\tilde{y}_{n+1} = P_n^T \cdot L^T \cdot \tilde{x}_n
\]  

(3–14)

where

\[
P_n^T = [p_{1,n}; p_{2,n}; \cdots; p_{K,n}]
\]

\[
L^T = \begin{bmatrix} 
\tilde{L}_1^T \\
\vdots \\
\tilde{L}_K^T 
\end{bmatrix}_{K \times D}
\]

For the training data, the desire signal \( y_{n+1} \) will be available. Therefore the optimal coefficients for each local model can be determined by deriving the mean
square error criterion (MSE)

\[
\min_L (J) = \frac{1}{N} \sum_{n=1}^{N} (y_{n+1} - \tilde{y}_{n+1})^2 = \frac{1}{N} \sum_{n=1}^{N} (y_{n+1} - P_n^T \cdot \vec{L}_n \cdot \vec{x}_n)^2
\] (3–15)

The extreme (minimum) happens when the first order derivative of \( J \) to \( L \) is "0"

\[
\frac{\partial J}{\partial L_k} = -\frac{2}{N} \sum_{n=1}^{N} (y_{n+1} - \tilde{y}_{n+1}) P_{k,n} \cdot \vec{x}_n = 0
\]

\[ e_n = y_{n+1} - \tilde{y}_{n+1} \]

\[
\Rightarrow \frac{\partial J}{\partial L} = \begin{bmatrix}
-\frac{2}{N} \sum_{n=1}^{N} e_n \cdot P_{1,n} \cdot \vec{x}_n \\
\vdots \\
-\frac{2}{N} \sum_{n=1}^{N} e_n \cdot P_{K,n} \cdot \vec{x}_n
\end{bmatrix} = 0
\] (3–16)

From (3–17), using the result from (3–13), each row of \( \frac{\partial J}{\partial L} \) can be expanded as

\[
\sum_{n=1}^{N} (y_{n+1} - \sum_{k=1}^{K} P_{k,n} \cdot \vec{x}_n^T \cdot \tilde{L}_k) \cdot P_{r,n} \cdot \vec{x}_n = 0
\]

\[
\sum_{n=1}^{N} y_{n+1} \cdot P_{r,n} \cdot \vec{x}_n = \sum_{k=1}^{K} (\sum_{n=1}^{N} P_{k,n} \cdot P_{r,n} \cdot \vec{x}_n \cdot \vec{x}_n^T) \cdot \tilde{L}_k
\] (3–18)

From (3–18), we obtain the MSE-based optimization function for local linear model \( \tilde{L}_k \). The difference between winner-takes-all solution and this solution is that \( P_{k,n} \) will not be always 0 or 1, and it determines where and how much cluster \( k \) will take effect.
In order to obtain the values of all the local models, we can expand equation (3–18) to matrix format to simplify the notations

\[ \vec{P}_r = \sum_{n=1}^{N} y_{n+1} \cdot P_{r,n} \cdot \vec{x}_n \]

\[ = \begin{bmatrix} \sum_{n=1}^{N} P_{1,n} \cdot P_{r,n} \cdot \vec{x}_n \cdot \vec{x}_n^T \\ \vdots \\ \sum_{n=1}^{N} P_{K,n} \cdot P_{r,n} \cdot \vec{x}_n \cdot \vec{x}_n^T \end{bmatrix} \cdot \begin{bmatrix} \vec{L}_1 \\ \vec{L}_2 \\ \vdots \\ \vec{L}_K \end{bmatrix} \]

\[ = R_r \cdot \vec{L} \]

where \( \vec{L} \) is a vector representation of \( \vec{L} \).

Repeat (3–18) for all the Gaussian clusters, the local linear model can then be described as

\[ \vec{P} = \begin{bmatrix} \vec{P}_1 \\ \vec{P}_2 \\ \vdots \\ \vec{P}_K \end{bmatrix} = \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_K \end{bmatrix} \cdot \begin{bmatrix} \vec{L}_1 \\ \vec{L}_2 \\ \vdots \\ \vec{L}_K \end{bmatrix} = R \cdot \vec{L} \]

\[ \Rightarrow \vec{L} = R^{-1} \cdot \vec{P} \]

Here the \( k \)th Gaussian cluster’s local linear model is the vector \( \vec{L} \)’s \( (k-1) \times D+1 \) to \( k \times D \) elements. With all the formulations above, the basic GMM based local linear model estimation procedure can be summarized as:

1. Accomplish state modeling using the EM algorithm, obtain distribution parameters \( (\mu_k, \Sigma_k) \) for every Gaussian cluster, conditional probability \( p(\vec{x}_n | c_k) \), and the cluster probability \( p(c_k) \);

2. Compute the posterior probability \( p(c_k | \vec{x}) \) through the Bayesian function for each cluster;
3. Compute the local linear models using distribution information from previous two steps and least square criterion;

4. In the testing phase, re-calculate the $p(\vec{x}_n|c_k)$ and $p(c_k|\vec{x})$ given new data point, then estimate the system output through local linear models.

### 3.4 Improvement on GMM

Even though the EM algorithm improves the likelihood at every iteration, the random initial conditions usually slow down the converge speed, and many times, make the model converges to local maximum [47, 48]. In order to optimize the GMM initial conditions, shorten the EM training time, and stabilize the final GMM performance, a Growing-Self-Organizing Maps (G-SOM) algorithm is applied. Another benefit from G-SOM is that it supplies a flexible structure to change the number of Gaussian kernels in the GMM. Vlassis and Likas set up a criterion to determine optimal number of Gaussian kernels [49].

The initial topology of the G-SOM network is a two-dimensional structure with 3 PEs, which will be used to describe clusters of patterns in the D-dimensional data vector $\mathbb{R}^D$. For this reason, every PE $c$ has a D-dimension weights vector $\vec{w}$ indicating its virtual position in $\mathbb{R}^D$. During the self-organizing process, one new PE will be added to the map at the end of each training iteration, preserving a triangle connection scheme at all time as shown in figure 3–2.

The adaptation in the G-SOM model can be formulated as:

1. For every input data, locate the winning PE through the Euclidean distance $s = \min[\text{dis}(\vec{x} - \vec{w})]$;

2. Increase the matching performance for $s$ and its direct neighbors $N_s$ with $\Delta \vec{w}_s = \varepsilon_b(\vec{x} - \vec{w}_s)$, $\Delta \vec{w}_c = \varepsilon_n(\vec{x} - \vec{w}_c)$ (for all $c \in N_s$);

3. Increase the winning frequency counter $\tau_s$ of $s$ with increment $\Delta \tau_s = 1$;
Figure 3–2. Explanation of G-SOM algorithm.
Left: Training of G-SOM; Right: Growing of G-SOM

4. Decrease all signals’ $\tau_s$ counters by a small value so that the mean of winning frequency doesn’t change too much;

5. After feeding in all data, one new PE is added between the PE with the highest $\tau_s$ and its farthest neighbor until the pre-defined PE count is reached, and all the winning frequencies are set to zero.

After training, the G-SOM PEs’ weights $\vec{w}$ are utilized as GMM’s means’ initial value $\mu$ with balanced winning frequency. The GMM’s covariance matrices are initially set to a large value to make every Gaussian kernel cover the largest possible area, and the optimal value of covariance matrix will be trained through EM algorithm. The improved algorithm was extensively tested on the LoFLYTE data that will be fully presented in Chapter 5. The LoFLYTE simulation uses 32-D time embedding space, and the difference of performance between the conventional GMM and the GMM with a G SOM initialization is shown in table 3–1. Every SER result is an average of five independent experiments. Overall, the G-SOM initial weighted GMM makes system identification performance $2.4dB$ better than random initialized GMM with smaller variance and decreases the EM training iterations.
Table 3–1. Performance on 32-D LoFLYTE data

<table>
<thead>
<tr>
<th>40 dB SNR noise</th>
<th>GMM</th>
<th>GMM with G-SOM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SER (dB)</strong></td>
<td><strong>Variance</strong></td>
<td><strong>SER (dB)</strong></td>
</tr>
<tr>
<td>p</td>
<td>23.637</td>
<td>0.4338</td>
</tr>
<tr>
<td>q</td>
<td>26.012</td>
<td>0.2849</td>
</tr>
<tr>
<td>r</td>
<td>24.663</td>
<td>0.3859</td>
</tr>
<tr>
<td>u</td>
<td>21.016</td>
<td>0.3574</td>
</tr>
<tr>
<td>v</td>
<td>22.682</td>
<td>0.2723</td>
</tr>
<tr>
<td>w</td>
<td>22.829</td>
<td>0.3404</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>23.473</strong></td>
<td><strong>0.3458</strong></td>
</tr>
<tr>
<td><strong>Training loops</strong></td>
<td>17</td>
<td>4</td>
</tr>
<tr>
<td>0 noise</td>
<td></td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>25.514</td>
<td>0.1968</td>
</tr>
<tr>
<td>q</td>
<td>27.828</td>
<td>0.2189</td>
</tr>
<tr>
<td>r</td>
<td>22.053</td>
<td>0.3631</td>
</tr>
<tr>
<td>u</td>
<td>19.065</td>
<td>0.2817</td>
</tr>
<tr>
<td>v</td>
<td>26.530</td>
<td>0.1977</td>
</tr>
<tr>
<td>w</td>
<td>23.315</td>
<td>0.2661</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>24.051</strong></td>
<td><strong>0.2523</strong></td>
</tr>
<tr>
<td><strong>Training loops</strong></td>
<td>14</td>
<td>2</td>
</tr>
</tbody>
</table>

### 3.5 Comparison of the GMM with the SOM

We will make a detailed comparison of the GMM with the SOM as SOM-LLM has been such a popular approach for SI and these two are so close to each other. The Self-Organizing Map (SOM) is another local modeling algorithm proposed by Kohonen and extensively utilized in the Computational NeuroEngineering Laboratory, with considerable success to system identification and control. Using the clustering idea applied to the reconstructed data in state space, the SOM sets up representative vectors locally, with a neighbor relationship determined by the multi-dimensional Euclidean distance.

Yet the major problems that SOM cannot overcome are: the proliferation of prototype vectors in particular in high dimensional spaces, the sensitivity to noise,
the non-smooth switching between models because of the *winner-takes-all* mechanism, and the local minima entrapment during SOM training [48, 50]. Nothing can be done for the first problem except increasing the network size, which brings also slow training, requirement of large amounts of data, and many local linear models. Although the SOM is insensitive to noise below the quantization produced by the voronoi tessellation, when the noise is higher the wrong model can be instantiated. The SOM preserves neighborhoods so the difference is not drastic, but contributes to modeling error. Because of the difference between the LLMs, when the models switch there may be transients in the response that also add to the modeling error for more than one time step.

The GMM approach divides state space in coarser regions, which improves the poor scaling up of the SOM with dimension, and also improves the noise robustness. However, each region may not be as well modeled by a linear model as in the case of the SOM. However, only experimental evidence can judge this trade-off. Instead of using *winner-takes-all* criterion, GMM-LLM combines multi models into the current data. In the case the Euclidean distance between data and the closest Gaussian kernel is large, maximum likelihood criterion makes sure more than one kernel takes effect. This phenomenon is shown in figure 3–3. Also, figure 3–3 shows that compared with other linear membership functions, the use of non linear Gaussian distribution can guarantee smooth switching between models. An important characteristic of the GMM-LLM is that the weighting is changing during testing, unlike the SOM-LLM. This will help in the overall modeling accuracy.

The local minimum in the SOM training is difficult to control during the annealing process. Also the SOM cannot control the winning frequency which may
also provide added difficulties in the clustering. The GMM is also sensitive to local maxima. However, in order to make GMM converge to global optimal, in a more stable manner, G-SOM makes the weights such that every cluster is generated with similar amount of winning frequency [51, 48]. This means that for the whole training data set, each Gaussian kernel occupies about the same portion as a winner with large output weight. Fritzke, Chinrungrueng, and Séquin proved the weights growing criterion and the optimal validity respectively [52, 53].

Since the SOM algorithm is derived from heuristic ideas and this leads to a number of limitations, some of the advantages of GMM over SOM are listed below.

1. GMM defines an explicit probability density function in data space to estimate the distribution of the state. In contrast, SOM uses winner-takes-all criterion based on the Euclidean distance [18]. GMM’s soft-combination mechanism makes smooth switching possible.

2. In GMM, the neighborhood-preserving nature of the mapping is an automatic consequence of the choice of Gaussian functions which improves GMM’s noise robustness. Smooth neighborhood-preservation is not guaranteed by the SOM procedure.
3. Through the relative winning frequency criterion from G-SOM construction, the GMM training avoids the local minima entrapment which SOM alone can not overcome.

4. GMM improves the scalability of SOM through a coarser clustering of the state space. For the same reason, the training efficiency of GMM is increased.

3.6 Summary

With detailed explanation EM algorithm, GMM has been theoretically proven feasible on modeling analysis, and its practical modeling capability could tested on various system. Gaussian distribution makes smooth switch between multiple models possible. And using the soft combination mechanism, as we made the comparison previously, GMM can largely decrease computation burden on modeling problem compared with SOM based algorithm. Gaussian models part is trained in un-supervised way as there is no desire distribution pre-assigned. The LLM part, as desire system response is available, will be trained in a supervised way to make sure the convergence of error square (MSE). Since GMM based modeling algorithm can ease the work of controller design, the benefits will be explained in chapter 4 and the simulation results and the detailed performance comparison will be shown in chapter 5.
CHAPTER 4
GAUSSIAN MIXTURE MODEL BASED CONTROL SYSTEM

Within the framework of predictive control, the proposed GMM based local linear modeling approach simplifies the design of control systems for nonlinear plants. It is difficult to design globally stable nonlinear controllers with satisfactory performance at every point in the state space of the closed loop system, which is especially true in the case of unknown plant dynamics. However, the GMM local linear modeling approach presented above, coupled with strong controller design techniques [54] and recent theoretical results on switching control systems [55, 56], achieves the control goal. The topology of local controllers that naturally arise from the local linear modeling approach is illustrated in figure 4–1, where the local controllers are directly linked to each local model. Within the framework of GMM-LLM, the system dynamics can be described piece-wise-linearly as

\[
\tilde{y}_{n+1} = \sum_k \alpha_k L_k^T \cdot X_n
\]

\[
= L_{OPT}^T \cdot X_n
\]

\[
= A_0 y_n + A_1 y_{n-1} + \cdots + A_D y_{n-D} \\
+ B_0 u_n + B_1 u_{n-1} + \cdots + B_D u_{n-D}
\]

where \(X_n\) is the time embedding vector \([\cdots, y_i, \cdots, u_i, \cdots]\), \(L_{OPT}\) stands for the current optimal model coefficients \([\cdots, A_i, \cdots, B_i, \cdots]^T\), \(y_i\)'s are the system states, \(u_i\)'s are the control signals.
In section 4.1, a brief review on mixture model based controller design will be given. In section 4.2, the GMM-based dynamical inverse controller is explained in detail, which is later demonstrated feasible on nonlinear system in chapter 5. A pole-placement PID control approach will be discussed in section 4.3.

Considering modeling error, measurement noise, and model uncertainty, robust approaches to control become very appropriate for real-world applications. Therefore various approaches have been proposed to solve the robust control problem, including the $H_\infty$ approach [57], Lyapunov approach [58], geometric approach [59], etc. Lin, Brandt, and Sun proposed an optimal control based robust control scheme [60, 61], and we will modify it and fit it into our GMM-LLM framework.

### 4.1 Introduction of Mixture Model Controller

The design of predictive controllers, especially for large-scale and complex applications, depends integrally on the model of the physical plant [62, 63, 64]. Indeed, the
model constitutes the vital link between the physical problem in which the optimal control will be used, and the mathematical realm in which it needs to be designed. The classical approach to the control design problem assumes a known model, even if dramatically reduced. In the case that the model complexity is reduced unrealistically, the consequence will be the generation of ineffective controls. For global nonlinear modeling control approach, which is more difficult compared with local or linear modeling control, neural networks based control scheme have been studied extensively [65, 54, 66].

Using the Partition Algorithm (PA), Lainiotis first introduced the multi-model partition (MMP) methodology for the design of adaptive/intelligent systems, where the control problem is decomposed into a set of easier derived subproblems [64, 67]. Given a set of local models, MMP designs the adaptive control as

\[ u(n) = \sum_{k=1}^{K} \alpha_k \cdot \hat{u}_k(n) \]  

(4-2)

where \( \hat{u}_k(n) \) is the optimal control corresponding to \( k \)th sub-model.

Fuzzy control is another popular design methodology which attracts wide interest these days [68]. Among them, Takagi-Sugeno (T-S) fuzzy logic model, which applies fuzzy rules and membership function, is a widely studied and applied fuzzy modeling and control methodology. In the T-S fuzzy model, the system is described by a set of if-then rules which have local models in the consequent parts. The \( ith \) rule in the model is of the form: If \( z_1(t) \) is \( M_{i,1} \), and . . . , and \( z_P(t) \) is \( M_{i,P} \), then

\[ s \mathbf{x}(t) = A_i \mathbf{x}(t) + B_i \mathbf{u}(t) \]
The effective regions of *if-then* rule are overlapped with each other, and the weights of each rule is measured by linear membership function which is shown in figure 4–2. The control signal for fuzzy logic controls take the form of (4–2), where $\alpha_k$ is coming from membership function too. Tanaka, Iwasaki, and Wang use T-S logic switching fuzzy logic model to control of an R/C hovercraft [69].

Based on the idea of a group of simple controllers instead of a single complex controller, switching control has been applied in recent years. A switching control consists of a high level discrete-event supervisor and a family of low level feedback controllers [70, 7]. Following the local modeling theorem, multiple controllers are applied to certain regions of data. Motter used a SOM based system identification model to design a switching controller for high speed wind tunnel control [20]. Li and Lee use a SOM structure to cluster the state space, and then apply fuzzy rules to design the control signals [41]. Cho et al. uses a SOM and sliding mode controller to realize robust control [71, 72]. Based on those approaches, our research will try
4.2 Inverse Error Dynamic Controller Using GMM

The mission of controller design for nonlinear dynamical system could be alleviated using local linear model generated from GMM. Once the optimal local linear models have been chosen from the input-output data available for system identification, inverse error dynamic controller design can be applied to meet goals based on the desire signal, the state signal, and the local model. Compared with the basic structure of a mixture model controller, inverse control is an extended version of single-model control since the *soft-combination* happens before the combination of control weights and *inverse* is applied to the unique current model. The structure of the controller is shown in figure 4–3.
The principle behind inverse error dynamic control (IEDC) is the asymptotic convergence of system error. The control methodology can be described as fixing a set of stable poles for the tracking error signal dynamics. If at time \( n \) the desire plant output is denoted as \( d_n \), and the actual plant output is \( y_n \), the instantaneous error is defined as \( e_n = d_n - y_n \). And the control goal is to guarantee that the error signal follows the \( l \)-D error dynamic function:

\[
e_{n+1} + \lambda_1 e_n + \lambda_2 e_{n-1} + \cdots + \lambda_l e_{n-l+1} = 0 \tag{4-3}
\]

the parameter vector \( \vec{\lambda} = [\lambda_1, \ldots, \lambda_l]^T \) is selected such that the roots of the polynomial \( 1 + \lambda_1 x + \cdots + \lambda_l x^l = 0 \) are all within the unit circle and thus the error dynamics is stable.

If the order of polynomial is zero, the error function is simplified as \( e_{n+1} = 0 \). The error function corresponds to an exact tracking controller, which determines the necessary control input for the next error value to be “0”. This strategy uses only current model information and current control error, so its response will be instantaneous. And obviously it is not robust to modeling errors or external disturbances as there is no further model and dynamic information to smooth the disturbance from model uncertainty. Through 2nd order error dynamical function, the error signal is
filtered first, and the local model based control signal can be obtained as

\[ e_{n+1} + \lambda_1 e_n + \lambda_2 e_{n-1} = 0 \]

\[ \Rightarrow [d_{n+1} - y_{n+1}] + \lambda_1 e_n + \lambda_2 e_{n-1} = 0 \]

\[ \Rightarrow [d_{n+1} - \tilde{y}_{n+1}] + \lambda_1 e_n + \lambda_2 e_{n-1} = 0 \]

\[ \Rightarrow \left\{ d_{n+1} - \begin{bmatrix} \bar{L}_{n,x} \bar{L}_{n,u} \end{bmatrix}^T \begin{bmatrix} \bar{x}_n \\ \bar{u}_n \end{bmatrix} \right\} + \lambda_1 e_n + \lambda_2 e_{n-1} = 0 \]

\[ \bar{u}_n = \begin{bmatrix} \bar{L}_{n,u}^T \cdot \bar{L}_{n,u} \end{bmatrix}^{-1} \cdot \bar{L}_{n,u}^T \cdot [d_{n+1} - \bar{L}_{n,x} \bar{x}_n + \lambda_1 e_n + \lambda_2 e_{n-1}] \]

where model \([\bar{L}_{n,x} \bar{L}_{n,u}]^T = \bar{L}_{\text{current}}^T = (P_n^T \cdot \bar{E})^T\) is coming from \((3-14)\) and \((\lambda_1 e_n + \lambda_2 e_{n-1})\) is from weighted error signals of last two steps.

For different choices order in \((4-3)\), the control law will drive the tracking closed-loop system in various ways, meanwhile the error signal will always satisfy the linear difference equation given in \((4-3)\). As long as the location of poles are within the unit circle, the controller will guarantee the stable convergence of the dynamics. Considering that the modeling error from the GMM-LLM will affect the actual pole locations, the poles in \((4-4)\) cannot be chosen close to one in order to guarantee stability.

Now we discuss the stability issue for the controller. From \((4-4)\), \(\lambda_i\)'s are coefficients to be designed such that two poles \(p_i\)'s are within the unit circle where \(p_1 + p_2 = -\lambda_1\) and \(p_1 \times p_2 = \lambda_2\). Since the LLMs are trained based on MSE criterion, the modeling errors at each step can be deemed a bounded i.i.d. random variable with zero mean. It can also be assumed that \(E[e_{M,n}^2] < \infty\), which is reasonable if the system identification is done properly, and the measurement noise levels are finite.
From the experimental results in the simulation section, we will show that the modeling error power is finite. Under those conditions and assumptions about nonlinear uncertainties on \(e_n\)s, the error dynamic function is changed to

\[
e_{n+1} + \lambda_1 e_n + \lambda_2 e_{n-1} = -(e_{M,n+1} + \lambda_1 e_{M,n} + \lambda_2 e_{M,n-1}) \tag{4-5}
\]

where \(e_{M,i}\)s are the modeling errors at each step.

We define the right side of equation (4-5) as \(\Delta\). Considering the worst case when modeling errors reach their bound, \(\Delta\) is set to its upper bound \(\Delta_u\) (\(|\Delta| \leq |\Delta_u|\) and \(\Delta_u\) is a finite constant). The expression of \(e_n\) can be obtained through \(Z\) transform and inverse-\(Z\) transform to yield

\[
E(z) = \frac{\Delta_u}{1 - z^{-1}} \cdot \frac{z^{-1}}{1 + \lambda_1 z^{-1} + \lambda_2 z^{-2}}
\]

\[
e_n = \frac{\Delta_u}{(1 - p_1)(1 - p_2)} - \frac{\Delta_u}{(1 - p_1)(p_1 - p_2)p_1^{n+1}} - \frac{\Delta_u}{(1 - p_2)(p_2 - p_1)p_2^{n+1}} \tag{4-6}
\]

As \(n \to \infty\), the mean of the tracking error is obtained as

\[
\lim_{n \to \infty} E[e_n] = \lim_{n \to \infty} E\left[\frac{\Delta_u}{(1 - p_1)(1 - p_2)}\right]
\]

\[
= \lim_{n \to \infty} \frac{1 + \lambda_1 + \lambda_2}{(1 - p_1)(1 - p_2)} E[-e_{M,n}]
\]

\[
= E[-e_{M,\infty}] \tag{4-7}
\]

As for the power of tracking error, the expression can be written as

\[
\lim_{n \to \infty} E[e_k^2] = \lim_{n \to \infty} E\left[\frac{\Delta_u^2}{(1 - p_1)^2(1 - p_2)^2}\right]
\]

\[
= \lim_{n \to \infty} \frac{1 + \lambda_1^2 + \lambda_2^2}{(1 - p_1)^2(1 - p_2)^2} E[e_{M,n}^2]
\]

\[
= \frac{1 + \lambda_1^2 + \lambda_2^2}{(1 + \lambda_1 + \lambda_2)^2} E[e_{M,\infty}^2] \tag{4-8}
\]
Under the assumption that the modeling error (and measurement noise) contributions \( e_{M,n} \) are wide sense stationary random variables, we can compute the asymptotic (finite) tracking error in terms of the power spectral density of \( e_{M,n} \) and the coefficients \( \lambda_i \). As discussed before, the mean of modeling error is zero or small value. Consequently, the modeling error will take effect on tracking performance and the pole-placement problem must be solved considering the trade off between convergence speed (faster poles, wider pass-band) versus better disturbance rejection (slower poles, narrower pass-band).

### 4.3 PID Controller

PID stands for Proportional, Integral, and Derivative. PID control became a traditional control approach in the 20th century because of its simplicity and accuracy. Nowadays, researchers are trying to find new methods to better fine-tune the PID parameters. Considering our system identification framework, the design of the piece-wise linear PID controller will be accomplished once the system identification is finished. Based on the GMM-LLM framework, the pole-placement PID design technique could be realized based on \([73]\). Lately, Skoczowski et al. proposed a new method based on two-loop model following control (MFC) to improve robustness \([74]\). The mixture model based PID control scheme is shown in figure 4–4.

Simplifying the model function (4–1) to a second order time-embedded SISO form

\[
\tilde{y}_{n+1} = a_1 y_n + a_2 y_{n-1} + b_1 u_n + b_2 u_{n-1}
\]

The model’s current \( Z \)-domain transfer function can be written as

\[
F(z) = \frac{B(z)}{A(z)} = \frac{b_1 z^2 + b_2 z}{z^2 - a_1 z - a_2}
\]
Defining the desire signal and controller transfer function as $D(z)$ and $G(z) = D(z)/C(z)$ respectively. The close-loop control signal transfer function $U(z)$ is given by

$$U(z) = \frac{D(z)}{C(z)}(D(z) - Y(z))$$

From the definition of pole-placement, the goal of this approach is to map the coefficients of close-loop system function from open loop coefficients $(a_1, a_2, b_1, b_2)$ to predesigned controller coefficients $(c_i, d_i)$, whose characteristic polynomial could be described as

$$z^2 + \lambda_1 z + \lambda_2$$

The close-loop system transfer function is then $\frac{1}{1+zA(z)C(z)/B(z)D(z)}$, and the system characteristic polynomial becomes

$$A(z)C(z) + z^{-1}B(z)D(z)$$
Using the characteristic polynomial for the system model as \((4-10)\), along with a second order observer polynomial term \(z^2\), the design equation for pole-placement PID is given as

\[
A(z)C(z) + z^{-1}B(z)D(z) = z^2(z^2 + \lambda_1 z + \lambda_2)
\]

\[
(z^2 - a_1 z - a_2)(z - 1)(z + c) + z^{-1}(b_1 z^2 + b_2 z)(d_1 z^2 + d_2 z + d_3)
\]

\[
= z^2(z^2 + \lambda_1 z + \lambda_2)
\]

(4–12)

where a \((z - 1)\) factor is added to the denominator of controller function \(C(z)\) in order to filter the low-frequency noise. This equation could be solved for \(c\),\(s\) and \(d\),\(s\).

Consequently the control signal is obtained from (4–12) as

\[
u_n = (1 - c)u_{n-1} + c \cdot u_{n-2} + d_1 e_n + d_2 e_{n-1} + d_3 e_{n-2}
\]

(4–13)

Given the LLM on the current time step, the PID controller could be easily obtained through (4–13). Its stability will be assured as we discussed in previous section for the dynamic inverse controller. One possible problem caused under this framework is that the PID coefficients need to be calculated on every step as the system’s local model and the gating weights from GMM are changed.

### 4.4 Optimal Robust Controller

In order to achieve better control performance, we are going to first translate a robust control problem into an optimal control problem and then apply optimal control approaches to solve the robust control problem. We will show later that, by properly choosing a set of cost functions that reflects the modeling uncertainty, noise, and control requirements, the solution to the optimal control problem will be also a solution to the robust control problem [60, 61].
**Definition 4.1: Robust control problem**

Find a feedback control law \( u = k(x) \) such that \( x = 0 \) of the closed-loop system

\[
\dot{x} = A(x) + B(x)N(x) + B(x)k(x) \tag{4-14}
\]

is globally asymptotically stable for all admissible perturbations \( N(x) \). \(^1\)

**Definition 4.2: Optimal control problem**

Find a feedback control law \( u = k(x) \) that minimize the following cost function

\[
\int_0^\infty (N_{\text{max}}^2(x) + x^T x + u^T u) dt
\]

where \( N_{\text{max}}^2(x) \) is the power of uncertainty compensation; \( x^T x \) is the cost of states change; \( u^T u \) is the cost of control signals. \( x^T x \) and \( u^T u \) can be replaced by \( x^T Q x \) and \( u^T R u \) with \( Q \) and \( R \) being symmetric and positive semi-definite to adjust the relative weights of states and controls.

**Theorem:** Assume that the solution to the optimal control problem exists. Then the solution to the optimal control problem is also a solution to the robust control problem.

**Proof:** Define

\[
V(x_0) = \min_u \int_0^\infty (N_{\text{max}}^2(x) + x^T x + u^T u) dt
\]

to be the minimum cost of bringing the system \( \dot{x} = A(x) + B(x)u \) from \( x_0 \) to the equilibrium \( “0” \). The Hamilton-Jacobi-Bellman equation \([75]\) gives

\[
\min_u [N_{\text{max}}^2(x) + x^T x + u^T u + V_x^T(x)(A(x) + B(x)u)] = 0
\]

\(^1\) We assume the perturbation is admissible, so we use the same \( B(x) \) as coefficients.
where $V_x(x) = \partial V(x)/\partial x$. Thus, if $u = k(x)$ is the solution to the optimal control problem, then

$$N_{\text{max}}^2(x) + x^T x + u^T u + V_x^T(x)(A(x) + B(x)k(x)) = 0$$

$$2k^T(x) + V_x^T(x)B(x) = 0 \quad (4-15)$$

Since $V(x)$ satisfies

$$V(x) > 0, x \neq 0$$

$$V(x) = 0, x = 0$$

Also, $\dot{V}(x) = dV(x)/dt < 0$ for $x \neq 0$, because

$$\dot{V} = (\partial V(x)/\partial x)^T(dx/dt)$$

$$= V_x^T(x)[A(x) + B(x)N(x) + B(x)k(x)]$$

$$= V_x^T(x)[A(x) + B(x)k(x)] + V_x^T(x)B(x)N(x)$$

$$= -N_{\text{max}}^2(x) - x^T x - k(x)^T k(x) + V_x^T(x)B(x)N(x)$$

$$= -N_{\text{max}}^2(x) - x^T x - k(x)^T k(x) - 2k(x)^T N(x)$$

$$= -N_{\text{max}}^2(x) + N(x)^T N(x) - (N(x) + k(x))^T (N(x) + k(x)) - x^T x$$

$$\leq -(N_{\text{max}}^2(x) - N(x)^T N(x)) - x^T x$$

$$\leq -x^T x < 0$$

Thus the condition of the Lyapunov local stability theory are satisfied. The equilibrium $x = 0$ of (4–14) is globally asymptotically stable for all possible uncertainty $N(x)$, i.e. $u = k(x)$ is a solution to the robust control problem. Consequently, there exists a neighborhood $\mathcal{N} = \{x : \|x\| < c\}$ for some $c > 0$ such that if $x(t)$ enters $\mathcal{N}$,
then
\[
\lim_{t \to \infty} x(t) = 0 \quad (4-16)
\]
But \(x(t)\) cannot remain forever outside \(\mathcal{N}\), otherwise \(\|x(t)\| \geq c\) for all \(t \geq 0\). Therefore,
\[
V(x(t)) - V(x(0)) = \int_0^t \dot{V}(x(\tau))d\tau \\
\leq -\int_0^t \|x(\tau)\|^2d\tau \\
\leq -\int_0^t c^2d\tau = -c^2t
\]
Let \(t \to \infty\), we have
\[
V(x(t)) \leq V(x(0)) - c^2t \to -\infty
\]
which contradicts the fact that \(V(x(t)) \geq 0\) for all \(x(t)\). Therefore \((4-16)\) is true no matter where the trajectory begins \([60]\). Consequently the stability is assured without considering the initial condition.

4.4.1 LLM-based Optimal Set-Point Controller

Typically, the solution of the optimal controller problem for time-invariant time-delayed (TITD) system is obtained by solving the infinite-dimensional algebraic Riccati equation which requires the solution of simultaneous partial differential equations. For the mixture model control case, we have proven that, \(\text{GMM-LLM modeling error} \) is bounded and not larger than single ARX modeling error. And the control error, under the condition that the controller is stable, will asymptotically converge to the modeling error bound. Moreover, as the Gaussian local model is changing at every step, the Riccati equation will need to be calculated on every step.
The computational technique, which was originally proposed by Yashiki and Matsumoto [76] on SISO single delay system and later expanded by Klein and Ramirez et al.[77] on MIMO multiple delays system, needs to be transformed to our one-step prediction model into state variable format for the purpose of controllability checking. The general case of MIMO systems exhibiting multiple delays is written as (4–1). And the corresponding state space model for MIMO system can be expressed as

\[
\begin{pmatrix}
    y_{1,n+1} \\
    \vdots \\
    y_{D,n+1} \\
    y_{n+1}
\end{pmatrix} = \tilde{A} \mathbf{x}_n + \tilde{B} \mathbf{u}_n \\
(4-17)
\]

\[
= \begin{pmatrix}
    0 & \cdots & 0 & A_1 \\
    \vdots & \ddots & \vdots & \vdots \\
    0 & \cdots & 0 & A_D \\
    I & \cdots & I & A_0
\end{pmatrix}
\begin{pmatrix}
    y_{1,n} \\
    \vdots \\
    y_{D,n} \\
    y_{n}
\end{pmatrix} + 
\begin{pmatrix}
    B_1 \\
    \vdots \\
    B_D \\
    B_0
\end{pmatrix} \mathbf{u}_n
\]

where only the \( y_{n+1} \) is the actual system state. The remaining vector \( y_1, \cdots, y_D \) is called pseudo-state vectors, which represent the time-delayed portion of the system states. The pseudo-state is defined as

\[
y_{i,n} = A_i \cdot y_{i,n-1} + B_i \cdot u_{n-i} \quad i = 1, \cdots, D \quad (4-18)
\]
A main advantage of this transformation is for the controllability test. Defining \( \tilde{A} \) and \( \tilde{B} \) as the coefficient matrix and vector in (4–17)

\[
\tilde{A} = \begin{pmatrix}
0 & \cdots & 0 & A_1 \\
\vdots & \ddots & \vdots & \vdots \\
0 & \cdots & 0 & A_D \\
I & \cdots & I & A_0
\end{pmatrix} \quad \tilde{B} = \begin{pmatrix}
B_1 \\
\vdots \\
B_D \\
B_0
\end{pmatrix}
\]

the standard state controllability test is simplified to check the rank of the state matrix \( S \)

\[
\text{rank}(S) = \text{rank}[ \tilde{B} \tilde{A} \tilde{B} \cdots \tilde{A}^{D+\sum d_i-1} \tilde{B} ]
\]

In our experiments, all the LLM give non-zero results, and since the rank of \( S \) is related to the dimension of LLM is relatively low (\( S = [\tilde{B}, \tilde{A} \tilde{B}] \) for 4-D LLM case), the controllability is assured. Only in the case when LLM have a large part of close-to-zero coefficients, which also means LLM are not well trained, the state matrix will not be full rank.

To realize the design of robust control signal, we start the cost function \( J \) as the basic design rule:

\[
J = \sum_n (X_n^T Q X_n + U_n^T R U_n) \quad (4–19)
\]

Considering the modeling uncertainty and measurement noise, (4–19) can be modified as

\[
J = \sum_n (N(X_n)^T V N(X_n) + X_n^T Q X_n + U_n^T R U_n) \quad (4–20)
\]

\[
= \sum_n (X_n^T Q^1 X_n + U_n^T R U_n)
\]
Here the modeling uncertainty and measurement noise are combined with the state itself. In other words, the physical dynamical changes because of the model uncertainty can be recognized as the increment of system dynamics which will in turn cost more energy and larger control signal to overcome them. This is a reasonable explanation as modeling error appears as a warping of system states. The solution from this kind of function forms the traditional linear quadratic regulator (LQR) problem and can be obtained through the solution from the discrete algebraic Riccati equation (DARE)

\[
A^T X A - X - A^T X B (B^T X B + R)^{-1} B^T X A + Q = 0
\]  (4–21)

There have been several methods to compute the unique stabilizing numerical solution \( P = X \) of the discrete-time algebraic Riccati equation [77], from which the gain vector and consequently the control signal are computed as

\[
G = (B^T P B + R)^{-1} B^T P A
\]  (4–22)

\[
U_n = -GX_n
\]  (4–23)

where \( X_n \) is the re-formulated current input with pseudo state elements, \( A \) and \( B \) are from state space form of real system model or \( \tilde{A} \) and \( \tilde{B} \) from state space form of LLM. The schematic diagram of the optimal robust control system is shown in figure 4–5.

One possible disadvantage for the optimal robust controller is that the related computation is larger than those of inverse controller, PID controller, etc. Especially under the condition of mixture model, theoretically there will always be a change on the local model as system dynamics are changing. Since DARE computation needs
to be finished right after the generation of Gaussian mixture model, consequently the DARE (4–21) and the control gain matrix (4–22) need to be re-calculated at every time step, which might be a challenge under some on-line training environment. Necessary trade-offs need to be made when resources are limited.

4.4.2 LLM-based Optimal Tracking Controller

Designing from the decrement of the cost function, the control signal(s) will make the system dynamics converge to a zero equilibrium point. For many real control problems, we want the system to be capable of taking some actions instead of converging to zero. In order to make the system follow non-zero desired tracks, the cost function needs to calculate the control gain matrix such that it can calculate the convergence of not only modeling uncertainty, but dynamical error and the change of control signal as well. Thus the new cost function (4–20) needs to be further modified.
for state errors and change of control signals

\[
J = \sum_{n} (N(X_n)^T V N(X_n) + E_n^T Q E_n + dU_n^T R dU_n) \tag{4-24}
\]

\[
= \sum_{n} (E_n^T Q^1 E_n + dU_n^T R dU_n)
\]

where \( E_n \) is the dynamical error of system states, \( dU_n \) is the difference of current control signals and their previous record, \( Q \) and \( R \) are symmetric and positive semi-definite weighting matrix. Again, we assume the modeling uncertainty and noise can be combined with state errors since errors can be described as distance between states and non-zero equilibrium. Correspondingly, the transfer coefficients for error signal and control signals need to be changed. Starting from the system state transfer function in (3–14), the error transfer functions can be described as

\[
\begin{align*}
\tilde{y}_{n+1} &= a_1 x_n + a_2 x_{n-1} + b_1 u_n + b_2 u_{n-1} \\
e_{n+1} &= d_{n+1} - y_{n+1} \approx d_{n+1} - \tilde{y}_{n+1} \\
&= d_{n+1} - (a_1 x_n + a_2 x_{n-1} + b_1 u_n + b_2 u_{n-1}) \\
&= d_{n+1} - a_1 (d_n - e_n) - a_2 (d_{n-1} - e_{n-1}) - b_1 u_n - b_2 u_{n-1} \\
&= (d_{n+1} - a_1 d_n - a_2 d_{n-1} - b_2 u_{n-1}) + a_1 e_n + a_2 e_{n-1} - b_1 u_n \\
&= a_1 e_n + a_2 e_{n-1} - b_1 (u_n - \frac{1}{b_1} (d_{n+1} - a_1 d_n - a_2 d_{n-1} - b_2 u_{n-1})) \\
&= a_1 e_n + a_2 e_{n-1} - b_1 (u_n - \hat{u}_n) \tag{4-25}
\end{align*}
\]

Defining \( \hat{u}_n = \frac{1}{b_1} (d_{n+1} - a_1 d_n - a_2 d_{n-1} - b_2 u_{n-1}) \) as a pseudo control signal, the new state space transfer function for the error dynamics and the control increment \([E, dU]\) is

\[
E_{n+1} = A \cdot E_n + B \cdot dU_n; A = \begin{bmatrix} 0 & a_2 \\ 1 & a_1 \end{bmatrix}; B = \begin{bmatrix} 0 \\ -b_1 \end{bmatrix} \tag{4-26}
\]
Considering the modeling uncertainty, the none-zero set point control and tracking control problems can be realized through optimal control law (4–24). The new discrete Riccati equation could be calculated in the same form with modified coefficients \((A, B)\) in (4–26). When the gain matrix from discrete Riccati equation is calculated, the current step control signal can be expressed as

\[
\begin{align*}
    dU_n &= -G \cdot E_n \\
    U_n &= \hat{U}_n + dU_n
\end{align*}
\]  

(4–27)

In (4–24) and (4–27), we use the upper bound of modeling error to describe the model uncertainty, and the real the model uncertainty and noise will be a random values which on average are smaller than that. Consequently under small noise and model uncertainty situations, the transient performance from optimal controllers, compare with other control approaches, might not be the best. However, since optimal controller is considering the long-term effect of disturbance, its performance on robustness will outrun all the other competitors in large noise cases. In chapter 5, we will simulate and discuss the performance of optimal controller on different real systems with disturbance-free case and the case with noise and/or model uncertainty presents. Neglecting the computational cost, optimal robust controller is the optimal choice for model based prediction controller.

### 4.5 Summary

GMM based mixture model controllers are close to switching controller, yet they consider the situation of combining weights, which dramatically decrease its computational cost comparing with SOM-similar vector quantization methods. For different controllers, the Gaussian models provides a simple linear description of the current
system dynamics as reference model. Under the condition that the Gaussian modeling systems are stable, all controllers can give a stable result where optional robust controller will theoretically give the best tracking performance. A plant model is rarely a fully accurate description of the real plant. Neglected dynamics, approximately known parameters, and changes in operating conditions all contribute to plant-modeling errors that can jeopardize controller performance. In the following chapter, we will answer the problems including the performance of the controllers in real systems, which includes accuracy and robustness, performance curve vs. number of PE, and performance comparison of GMM-LLM with other modeling approaches.

Among all the options, ESN controller is quite different compared with others, which will be discussed in details in appendix. Many papers mentioned that the training of ADC and/or BPTT algorithms will sometime inevitably lead to unstable control results because of its computational complexity. Besides the neural networks (ESN here), BPTT length and the change of learning rate will also take effect on the final performance. We will start from some nonlinear systems with “simple” setup, use it for set-point control.
CHAPTER 5  
APPLICATIONS AND DISCUSSIONS

The GMM based local linear modeling and control techniques which have been presented in previous chapters need to be tested on various system in order to prove their advantages including feasibility, simplicity, and robustness, etc. As an independent candidate, ESN will be used for system identification and partly be used for system control. In this chapter, different GMM-based control approaches will be tested on a set of system including chaotic time series prediction, synthetic SISO linear system identification, SISO nonlinear system identification and control, NASA LoFLYTE wave-rider aircraft identification and control. A complete set of simulation and experimental results will be provided from the application of these principles to the listed problems. Among all the systems, the Lorenz system, as a traditional chaotic system with high order nonlinear dynamics, and the missile system, as a traditional SISO control system, will be discussed in details about system identification performance and control performance respectively. For all the sample systems, we will assume that the system transfer functions are unknown, and the reference models will be constructed based on I/O measurements.

5.1 Chaotic Systems

The two chaotic systems we present here are the Lorenz system and the Duffing system which demonstrate irregular, complex behavior. Even though they are deterministic systems, their dynamics show unpredictable character. Recently considerable effort has been in the attention of nonlinear dynamic literature since removing chaos
can improve system predictability and avoid fatigue failure of the system. Several methodologies have been applied for the identification and synchronization of a variety of chaotic systems. For example, a generalized synchronization of chaos through linear transformation [78], adaptive control and synchronization of chaos using Lyapunov theory [79], and an adaptive variable structure control system for the tracking of periodic orbits [80]. The contribution of our work lies in the design of different control systems based on GMM-LLM approach.

5.1.1 The Lorenz System

One widely studied chaotic system which is also autonomous system, the Lorenz system equation (Lorenz, 1963), formulated to model the convection process in the atmosphere [81]. The Lorenz system can be described as:

\[
\begin{align*}
\dot{x} &= \sigma \cdot (y - x) \\
\dot{y} &= \rho \cdot x - y - x \cdot z \\
\dot{z} &= -\beta \cdot z + x \cdot y
\end{align*}
\]

\[(5-1)\]
where \( x, y, \) and \( z \) are the measurements of fluid velocity and horizontal and vertical temperature variations. The Lorenz system can exhibit quite complex dynamics depending on the parameters. In our experiment, parameters are set as \( \sigma = 10, \rho = 28, \) and \( \beta = \frac{10}{3} \) to obtain chaotic dynamics and the first state \( x \) is set as the output. Using \( 4th \) order Runge-Kutta integration with time step 0.01, 9000 samples were generated for analysis. Using the Lipschitz index analysis the embedding dimension is determined to be 3 and the embedding delay \( \tau = 2 \). The GMM training data is constructed as

\[
\bar{x}_n = [x(n-1), x(n-3), x(n-5)] \tag{5-2}
\]

The first 8000 samples from the time-series are used to create the reconstructed state space through eight-kernel GMM algorithm, and local linear models are trained for one-step prediction. Another 1000 samples are used for signal to error (SER) performance test. Figure 5.1.1 shows that the estimated output is quite close to the original Lorenz signal. And figure 5–3 shows the optimal distribution of Gaussian kernel among the state space. In order to further compare the identification performance between GMM and other approaches, we plot the figure to verify the performance increment of GMM-LLM as number of PE increase. From figure 5–5 and table 5–1, we find that “soft-combination” enable GMM to use fewer PEs. The total amount of

\[\text{Performance is evaluated by } \text{SER} = \frac{\sum_n (d_n^2)}{\sum_n (e_n^2)}\]
parameters in SOM-LLM and GMM-LLM approaches can be expressed as

\[ P_{SOM-LLM} = N \times D \times 2 \]
\[ P_{GMM-LLM} = N \times D \times 3 + N \]

where \( N \) is the number of PE, \( D \) is the dimension of each PE. Considering the comparison of GMM with SOM based modeling performance mentioned in chapter 3 and some other methodologies, it can be concluded that, as GMM uses far less number of PE, GMM based modeling algorithm can achieve similar performance with less burden of computation.

In order to make a complete comparison, along with traditional methods RBF and TDNN as two references, two 300-PE ESNs are constructed. One ESN is designed with even distribution of spectral radius \([36, 37]\), the other comes with optimized design criterion where larger part of internal weights distributed along the maximum spectral radius. Both use the same MSE training criterion. In table 5–1 we see that ESN can make better one-step prediction compared with the other approaches. Since
Table 5–1. Performance on the Lorenz system

<table>
<thead>
<tr>
<th>Approaches (# of PEs)</th>
<th># of parameters</th>
<th>SER (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBF (50)</td>
<td>203</td>
<td>31.05</td>
</tr>
<tr>
<td>TDNN (50)</td>
<td>250</td>
<td>29.23</td>
</tr>
<tr>
<td>SOM (10 × 10)</td>
<td>600</td>
<td>33.67</td>
</tr>
<tr>
<td>GMM (8)</td>
<td>80</td>
<td>29.16</td>
</tr>
<tr>
<td>ESN1 (300, 1D input)</td>
<td>300</td>
<td>35.05</td>
</tr>
<tr>
<td>ESN1 (300, 3D input)</td>
<td>300</td>
<td>43.40</td>
</tr>
<tr>
<td>ESN2 (300, 1D input)</td>
<td>300</td>
<td>36.54</td>
</tr>
<tr>
<td>ESN2 (300, 3D input)</td>
<td>300</td>
<td>45.20</td>
</tr>
</tbody>
</table>

Figure 5–3. Distribution of the Lorenz system (dash line) and GMM weights (star)

the ESN training is focused on the linear readout vector, the training for larger size of ESN can still be realized within reasonable amount of time. An interesting phenomenon in the result is that ESN rely less on the embedding dimension compare with other approaches, as one dimensional input already generate comparable prediction performance. And that complies with our theoretical explanation in chapter 2 and 4.

Once the GMM-LLM structure is optimized according to the proposed methodology, the controller design can be accomplished using standard linear feedback controller design techniques. Since the Lorenz system is originally an autonomous system,
the control could be realized for dynamic compensation. Given a desired signal $d_n$, the controller needs to find a control signal $u_n$ such that the system output error converges to zero. Under the condition that model prediction is close enough to the real dynamics, the Lorenz system will follow the control signal(s) to the equilibrium point.

$$\lim_{n \to \infty} |d_n - y_n| = 0 \quad (5-3)$$

For the Lorenz system, the 3-dimensional control signal vector is

$$u_n = x_{n+1} + T(d_n - x_n) \quad (5-4)$$

$$\approx \tilde{x}_{n+1} + T(d_n - x_n)$$

where one control signal corresponds to one state variable, and the predicted state vector $\tilde{x}_{n+1}$ is obtained from GMM-LLM prediction. $T$ is the error compensation coefficient between [-1,1], which confirms the error converges to zero. Following the
condition of desire signals as equilibrium points or a totally different dynamics, we can realize a stabilizing and synchronizing control. Figure 5–6 shows stabilizing and synchronizing performance where the controller takes effect at point 1000. We found that for both cases, the controller can compensate the dynamic error very fast and guarantees the convergence. One factor needs to be emphasized here is that the Lorenz system, being a chaotic system, is very sensitive to appropriate modeling.
Otherwise the controller will not converge the dynamics to equilibrium in case the system identification performance is not close enough to the real values.

5.1.2 The Duffing Oscillator System

We now consider another kind of chaotic oscillator system, the Duffing oscillator system. The most general forced form of the Duffing equation is

$$\ddot{x} + \delta \dot{x} + (\beta x^3 \pm \omega_0 x) = \gamma \cos(\omega t + \phi)$$  \hspace{1cm} (5-5)

Depending on the parameters chosen, the equation can take a number of special forms. For example, with no damping and no forcing, \(\delta = \gamma = 0\) and taking the plus sign, the equation becomes

$$\ddot{x} + \omega_0 x + \beta x^3 = 0$$

This equation can display chaotic behavior. For \(\beta > 0\), the equation represents a “hard spring”, and for \(\beta < 0\), it represents a “soft spring”. If \(\beta < 0\), the phase portrait curves are closed [21].

Without losing generality, we simplify the duffing oscillator with control signal. The system function can be re-written into state space format as

\[
\begin{align*}
\dot{x}_1 &= x \\
\dot{x}_2 &= u - \delta x_2 - \omega_0 x_1 - \beta x_1^3 + \gamma \cos(\omega t)
\end{align*}
\]

(5–6)

with corresponding constant coefficients \([\delta, \omega_0, \beta, \gamma] = [0.4, -1.1, 1.0, 1.8]\) and \(\omega = 1.8\) [82, 25, 21], and the control signal is set as \(|u| \leq 5\).

Using 4th order Runge-Kutta integration with time step 0.2s, 8000 samples were generated for analysis. The GMM-LLM is trained with the first 5000 points, and the
remaining is used for modeling testing and control tracking testing. The embedding dimension is determined as $X_k = [y_k, y_{k-1}, u_k, u_{k-1}]^T$. From experiments, the size of Gaussian models is set as eight PEs. For ESN part, the same amount of data is used for both training and testing. The data embedding is set as “1” dimension ($X_k = [y_k, u_k]^T$), and 300 internal weights are used for echo states.

The identification performance on duffing system is shown in figure 5–8, and sample of Gaussian distribution is shown in figure 5–9. Their SER performance long with comparison with single model performance are listed in table 5–2. Considering the structural difference, we will mainly compare GMM-LLM with RBF and SOM based model. We find that GMM is using far less amount of models, which means a simpler structure compare with other modeling paradigm, and the prediction performance is

Table 5–2. Performance on the Duffing oscillator

<table>
<thead>
<tr>
<th>MODEL (# of PEs)</th>
<th># of parameters</th>
<th>SER (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBF (50)</td>
<td>254</td>
<td>27.94</td>
</tr>
<tr>
<td>SOM (64)</td>
<td>512</td>
<td>28.06</td>
</tr>
<tr>
<td>ESN (300)</td>
<td>300</td>
<td>30.75</td>
</tr>
<tr>
<td>GMM (8)</td>
<td>104</td>
<td>26.46</td>
</tr>
</tbody>
</table>
Figure 5–8. System identification performance on the Duffing system
Left: GMM-LLM results; Right: ESN results.

still very good. Figure 5–9 shows that for a large part of the testing, *soft-combination* is taking effect among the Gaussian models, which complies with our explanation in chapter 3. ESN, on the other way, is using less input dimension, and the prediction is one of the best. Modified ESN, from now on, will be our primary option for the echo state setup as it is producing better results than its previous design.

Next, we try to realize two control missions based on Gaussian models. The first sets system dynamics to the zero equilibrium point; the second makes the system track a different dynamics. As the original dynamic is set as \([\delta, \omega_0, \beta, \gamma] = [0.4, -1.1, 1.0, 1.8]\) and \(\omega = 1.8\), the new one is set as \([\delta, \omega_0, \beta, \gamma] = [0.41, -1.1, 1.0, 2.0]\) and \(\omega = 1.9\).

For these control missions, we use the PID controller, sliding mode controller, and optimal robust controller. Considering the significance of the problem, the robustness comparison of different controller will be discussed in the nonlinear SISO system. For PID controller, the poles are set as \(p_{1,2} = 0.6\) to ensure stability and good converging speed. The parameters for sliding mode controller are set as \(c_1 = 1, c_2 = -2.5, \epsilon T = 0.0001, qT = 0.3\) [83, 84, 85]. For the optimal controller, the gain matrix is designed
Figure 5–9. Sample of weights change of 8 GMM kernels;

as

\[ Q^1 = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix} ; R = 0.01 \]

We compare the performance of controllers using 200s trajectory regarding falling time, settling time, and steady state error. Because of the relatively low SER system identification performance, none of the controller can make the Duffing system dynamic converge to desire track perfectly, as shown in figure 5–10 where 0s ~ 10s of dynamics is shown. Roughly speaking, the control performance of all three controllers are close to each other. In the set point control mission, however, optimal robust controller and sliding mode control clearly have shorter settling time and smaller steady state error over PID controller. Regardless rising (falling) time and settling time in the second figure, the PID controller shows large offset where system dynamics reach highly nonlinear range, e.g. at the start and near the extrema. Since the modeling for
the Duffing system does not consider noise, optimal robust controller does not show unique advantage here. Considering the robustness of optimal controller over disturbance, we can expect that it will outperform other competitors under the situation where modeling uncertainty and noise are both present.
5.2 SISO System

5.2.1 Linear SISO Mass-Spring System

Now we start to study single-input-single-output systems. In order to keep the consistency of our examples, we will start with a linear system to check the basic capability of our approach and then follow with nonlinear systems. The linear SISO system example is designed based on a single mass spring damper system, where the dynamics of the system can be easily described by two states as

\[
\begin{align*}
\dot{x}_1 &= \frac{1}{m} x_2 \\
\dot{x}_2 &= \frac{1}{m} [-k_1 x_1 - b x_2 + u] \\
y &= x_1
\end{align*}
\]  

(5–7)

where input \(x_i\)’s are states and \(x_1\) describes the movement of the mass, \(y\) is the output, and \(u\) is the external force [86]. In the experiment here \(u\) is limited within \(\pm 0.5\), other parameters are set as \(m = 5\), \(k = 100\), and \(b = 0.1\). The sample of system dynamics is shown in figure 5–11.

The amount of Gaussian models is chosen as 6 from experiments. The 6-kernel GMM was trained with 5000 samples of input-output pairs obtained from equation (5–7) using 4th order Runge-Kutta integration with time step of \(T_s = 0.05s\), with \(u\) being \(i.i.d.\) uniformly distributed random signal. The system identification data was constructed with embedding dimension of 4, according to Lipschitz index, for input and output both being 2. Thus \(\tilde{X}(n) = [x_1(n), x_1(n - 1), u(n), u(n-1)]\). For ESN modeling, the input dimension is again decreased to \(\tilde{X}(n) = [x_1(n), u(n)]\).

The identified models were tested on original 2000 samples data, generated using a new sequence of random input signal. The actual plant output, the model predicted
Figure 5–11. Dynamics of the mass system. top: X1; middle: X2; bottom: control signal.

Table 5–3. Performance on the linear mass system

<table>
<thead>
<tr>
<th>Model (# of PE)</th>
<th># of parameters</th>
<th>SER (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOM-based (8 × 8 = 64)</td>
<td>512</td>
<td>50.75</td>
</tr>
<tr>
<td>RBF (64)</td>
<td>324</td>
<td>47.30</td>
</tr>
<tr>
<td>GMM-based (6)</td>
<td>78</td>
<td>47.24</td>
</tr>
<tr>
<td>ESN (200)</td>
<td>200</td>
<td>49.05</td>
</tr>
</tbody>
</table>

output and the estimation error for the GMM-based LLM models are provided in figure 5–12. Figure 5–13 shows that cooperative modeling is taking effect during testing. The SER performance is 47.2388 dB. The same data is applied to SOM and RBF to make a comparison. Table 5–3 shows that GMM-LLM does not generate the best result. Yet considering the amount of PE used in GMM-LLM, GMM-LLM is a highly efficient modeling method.

The control mission for the plant can be accomplished through a series of controllers. The zero order inverse controller will directly replace next step prediction with desire signal. The PID controller design is carried out in the way of standard pole-placement technique. The corresponding PID coefficients are determined to
Figure 5–12. Sample of system identification performance on the Mass system. Left: GMM-LLM results; Right: ESN results.

bring the close-loop response poles to $0.05 + i0.3$ and $0.05 - i0.3$ in order to decrease the transient overshot, which are all verified in the simulation of cross-validation. The sliding mode controller, as a strong candidate for the robust control, is considered as a reference with parameters $c_1 = 1, c_2 = -2.5, eT = 0.0002, qT = 0.5$, which are fine-tuned results from simulation. For the optimal robust controller, the gain matrix are designed as

$$Q^1 = \begin{bmatrix} 1 & 0 \\ 0 & 5 \end{bmatrix}; R = 0.25$$

The set-point control mission is set as: system starts at a value of 0.5, and desire signal is 0 for 3 second, then the desire signal is changed to 0.3. In order to identify the robustness of different controllers, another set of experiment with 30 dB SNR noise is added to the system dynamics as modeling uncertainty. The final performance is shown in figure 5–14. We can see that in the noise free case, all controllers have similar settling time and small steady state error. PID controller has big overshoot, short falling time. Sliding mode controller has longer falling time yet small overshoot. Optimal controller has both short falling time and small overshoot. In 30dB noise case,
all three controllers do not have much difference on falling (rising) time, settling time, and overshoot. The optimal controller obviously outperform other two on the criterion of steady state error as the error variance from the optimal controller is smaller than those from other controllers.

5.2.2 Nonlinear SISO Missile System

Another SISO example considered here is a nonlinear dynamic missile model where only the yaw dynamic of the missile is studied. The simplified two states dynamic can be described by

\[
\begin{align*}
\dot{x}_1 &= x_2 - 0.1 \cos(x_1)(5x_1 - 4x_1^3 + x_1^5) - 0.5 \cos(x_1)u \\
\dot{x}_2 &= -65x_1 + 50x_1^3 - 15x_1^5 - x_2 - 100u \\
y &= x_1
\end{align*}
\]

where input \( u \) is the rudder deflection, and in practice \( u \) is limited within ±1 which corresponds to the range of rudder as ±90 degree.
Figure 5–14. Set-point control performance on the mass system.
Left: noise-free case; Right: 30dB SNR noise case.

Figure 5–15. Missile system dynamics.
Left: state space description; Right: X1, X2, and control (top to bottom).

The GMM was trained with 6000 samples of input-output pairs obtained from equation (5–8) using 4th order Runge-Kutta integration with time step of $T_s = 0.05s$, which corresponds to 300 seconds of flight time. In order to excite a rich variety of dynamical modes in the plant, the system identification data was generated using a uniformly distributed random input signal within the specified limits [2]. GMM-based LLM approach developed a eight-mode mixture models, resulting in eight cooperative linear models. The embedding dimension, according to Lipschitz index, for input and output were both selected to be 2, resulting in 4-coefficient local models. This
Figure 5–16. Sample of system identification performance on the missile system. Left: GMM-LLM results; Right: ESN results.

Table 5–4. Performance on the missile model

<table>
<thead>
<tr>
<th>Model (# of PE)</th>
<th># of parameters</th>
<th>SER (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOM-based ((8 \times 8 = 64))</td>
<td>512</td>
<td>31.70</td>
</tr>
<tr>
<td>RBF (100)</td>
<td>504</td>
<td>33.03</td>
</tr>
<tr>
<td>GMM-based (8)</td>
<td>104</td>
<td>31.00</td>
</tr>
<tr>
<td>ESN (300)</td>
<td>300</td>
<td>36.25</td>
</tr>
</tbody>
</table>

embedding delay dimension was also chosen in accordance with the dimensionality of the state dynamics.

The identified models were also tested on original 50s-length data (1000 samples), generated using a new sequence of random input signal. The actual plant output, the model predicted output and the estimation error for the GMM-based LLM models are provided in figure 5–16. The ESN is constructed with 300 PEs, and input is set to state without time embedding. The SER performance, compared with 31.7dB from 64-PE SOM in table 5–4, is 31 dB [18].

The control mission for the plant can be accomplished through a series of controllers. The zero order inverse controller will directly replace the next step prediction
Figure 5–17. Sample of weights change of 8 GMM kernels for missile system

with the desired signal. The PID controller design is carried out in the way of standard pole-placement technique. The corresponding PID coefficients are determined to bring the close-loop response poles from the plant output to $0.5 + i0.1$, and $0.5 - i0.1$. The sliding mode controller, as a strong candidate for the robust control, is considered as a reference with parameters $c_1 = 1, c_2 = -1.85, e_T = 0.0001, qT = 0.3$ [83, 84, 85]. For the optimal robust controller, the gain matrix are designed as

$$Q^1 = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}; R = 0.55$$

As we discussed in previous chapter, control performance will be determined by modeling performance as well as controller performance. Under the condition that the controller is stable, the control performance will eventually converge to the minimum of modeling error. In order to deliver the results more clearly, we show the figures in
Figure 5–18. Performance of PID controller with different modeling approaches.

the following sequence: we show the different-model-same-controller results first to
verify the effect from modeling part, then the same-model-different-controller results.

Figure 5–18 displays, under noise-free condition, the set-point control perfor-
mance of PID controller with different modeling approaching, plus TDNN controller
as a reference. Recalling the system function (5–8), the region around -1.5 is where
nonlinear dynamics is taking the largest effect. And figure 5–18 clearly shows that,
1). the single model PID (PID-MA) has longer settling time and the largest vibra-
tion before the convergence because of its modeling performance; 2). TDNN, due to
its structural difference, has the shortest falling time yet the longest settling time;
3). the Gaussian mixture model PID controller (MPID-GMM) has very close per-
formance to the SOM multiple model PID controller (MPID-SOM); 4). MPID-SOM
gives the most stable performance and it needs more training. For now we can con-
clude that GMM-LLM, when compared with other modeling approaches, can save
much computational cost and the final control performance to a large extent.
Figure 5–19. GMM-LLM based control performance on the missile system. (a): Step response without noise (output and desire); (b): Step response with noise (output and desire); (c): Tracking response without noise (output and desire); (d): Tracking response with noise (output and desire).

Now we test the performance of controllers under the condition that modeling uncertainty and measurement noise is present. The closed-loop nonlinear system is tested in two cases: one with various step changes to the desire output and one with a smoothly changing desired output, which is the sum of several sinusoids with different amplitudes, phases and frequencies. To simulate the worst condition, the disturbance is considered modeling uncertainty. After generating the training data, we add 30dB SNR noise to the system model during testing. In that case, modeling.
Table 5–5. Control performance comparison on missile system considering noise

<table>
<thead>
<tr>
<th></th>
<th>Set-point Error Mean</th>
<th>Set-point Error Variance</th>
<th>Tracking Error Mean</th>
<th>Tracking Error Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPID-GMM</td>
<td>-0.0289</td>
<td>0.0272</td>
<td>0.0070</td>
<td>0.0108</td>
</tr>
<tr>
<td>Optimal</td>
<td>-0.0222</td>
<td>0.0186</td>
<td>0.0048</td>
<td>0.0072</td>
</tr>
<tr>
<td>Sliding mode</td>
<td>-0.0272</td>
<td>0.0202</td>
<td>-0.0024</td>
<td>0.0101</td>
</tr>
</tbody>
</table>

Figure 5–20. ESN control performance on the missile system.
(a): Noise free set-point performance; (b): Set-point performance with 30dB noise.

effect is negligible and control performance is emphasized. From figure 5–19 it shows that the step and tracking performance of all the designed nonlinear control schemes in close-loop operation with actual plant model is very satisfactory. GMM-LLM modeling error is insignificant in the final performance. Comparing with other two approaches, PID needs the least calculation for coefficients. Considering the detail of figure 5–19, as in table 5–5, the difference among those approaches are noticeable, in which the optimal robust controller demonstrates the superior performance for disturbance rejection compared with others.

Finally, we try the ESN controller for set-point control in order to verify the feasibility of ESN controller. Following the structure in chapter 4, we construct a 300-PE ESN controller with random readout vector. The input to the controller is the current system states without time embedding, and the model information is
derived from the Gaussian mixture reference model. BPTT length is set to 100 step, which corresponds to 5 second in real system. The whole system is recursively trained in 15 iterations. The final control performance is show in figure 5-20. In the figure, we can see that the ESN controller can accomplish the regular control mission, yet its performance is not comparable to the optimal robust controller. Since the advantage of ESN is its generalization capacity, simplicity for training, and its independency of time embedding input, further investigation needs to be made for ESN based robust controller.

5.3 Nonlinear MIMO System: LoFLYTE System

The last and the most complex application we have considered is a MIMO nonlinear dynamic system, the LoFLYTE System from NASA. The LoFLYTE System has 6 degrees of freedom, 3-dimensional moving and 3-dimensional rolling, and 4-channel inputs. Since the flight motion can be divided into longitudinal motion and lateral motion, the modeling and control mission can be simplified into 2-input-2-output mission. We will design MIMO system identification model first, and realize inverse controller and optimal robust controller based on that.

5.3.1 System Introduction

The nonlinear MIMO dynamic system discussed here is the NASA LoFLYTE testbed aircraft, an unmanned aerial vehicle (UAV), which is designed and tested by Accurate Automation Corporation (AAC). The LoFLYTE testbed flies slower than the speed of sound, Mach one, but they have the same “waverider” shape designed for Mach five flight. The term “waverider” refers to the fact that aircraft of this type ride on the shock waves that they create as they fly above the speed of sound. Other supersonic and hypersonic aircraft suffer reduced performance because they
Figure 5–21. General description of an aircraft

fight against the shock waves. The waverider shape improves fuel consumption by decreasing air resistance at speeds greater than Mach one. This full scale aircraft will take off horizontally, then it will use air-breathing engines to accelerate to a cruising speed of Mach five at a very high altitude. And it will end its flight by landing on a conventional runway. The task here is to develop modeling and control strategies for LoFLYTE based solely on input-output data.

According to classical aerodynamics, the flight dynamics of any rigid body are determined by movements along and around three axes: roll, pitch (longitudinal motion), and yaw (lateral motion). Besides the strong coupling between system states, the main control effect can still be classified. The elevator $\delta_e$ is the main effect for controlling the longitudinal motion state variables (pitch angle, $\theta$, and pitch rate, $q$). The rudder $\delta_r$ is primarily controls the lateral motion state variables (yaw angle, $\psi$, and yaw rate, $r$). The aileron $\delta_a$ mainly controls the roll motion state variables (roll angle, $\phi$, and roll rate, $p$). Finally, the throttle $\delta_t$ largely controls the aircraft’s longitudinal speed, and for some planes, deflectable thrust vectors might allow yaw
Figure 5–22. Dynamics of the LoFLYTE system. Left: $p$ (top), $r$ (bottom); Right: Control signal, aileron (top), rudder (bottom).

and roll contributions from the engine power in some case. Under certain symmetry assumptions for the aircraft body, the state dynamics of the rigid-body aircraft can be described as follows

\begin{align*}
\dot{u} &= -(wq - vr) - g \sin \theta + F_x/m \\
\dot{v} &= -(ur - wp) + g \cos \theta \cdot \sin \phi + F_y/m \\
\dot{w} &= -(vp + uq) + g \cos \theta \cdot \cos \phi + F_z/m \\
\dot{p} &= [(I_{yy} - I_{zz})qr + I_{zz}(qr - pq) + L]/I_{xx} \\
\dot{q} &= [(I_{zz} - I_{xx})rp + I_{xx}(r^2 - p^2) + M]/I_{yy} \\
\dot{r} &= [(I_{xx} - I_{yy})pq + I_{zz}(pq - qr) + N]/I_{zz} \\
\dot{\phi} &= p + q \sin \phi \cdot \tan \theta + r \cos \phi \cdot \tan \theta \\
\dot{\theta} &= q \cos \phi - r \sin \theta \\
\dot{\psi} &= q \sin \phi \cdot \sec \theta + r \cos \phi \cdot \sec \theta
\end{align*}
where $u$, $v$, and $w$ are the speed components of the aircraft along its body axes $x$, $y$, and $z$ respectively. $p$, $q$, and $r$ are the angular speeds along those axes. And $\phi$, $\theta$, and $\psi$ are the Euler angles that define the rotation matrix between the body coordinate frame and the inertial coordinate frame. The gravity $g$ is along the down direction of the inertial coordinate. The throttle, engine power, and aerodynamic effects generate the forces $F_x$, $F_y$, and $F_z$ as well as the moments $L$, $M$, and $N$. The $m$, $I_{xx}$, $I_{yy}$, and $I_{zz}$ are the aircraft mass and moment of inertia respectively which are determined by the aircraft’s geometry.

The LoFLYTE program is an active flight test program at the Air Force Flight Test Center at Edwards air force base. The LoFLYTE aircraft is simulated using a software, C++ version or Matlab version, by ACC and is assumed to be close to the true plant. Without losing the generality, the throttle is assumed constant and the state variables $p$, $q$, $r$, $u$, $v$, $w$ are available for external measurement. The goal of the system identification and control problem is to determine local linear models from four inputs (aileron, elevator, rudder, and throttle) to six state variables and to control them in order to track a desired trajectory of flight.

5.3.2 System Identification and Control experiment

The experiment here is designed to use the Matlab version of the ACC flight simulator. The control signal is generated by manually flying with 3-dimensional joystick to imitate a test flight. The system transfer function is assumed un-known and state variables, output data, are obtained from the Matlab output.

In order to simplify the problem, the longitudinal motion of the system, axis $x$ (forward) and $z$ (down) and pitch rate $q$, are considered decoupled with lateral motions, consisting the axis $y$ (right), roll rate $p$, and yaw rate $r$. Instead of dealing
Figure 5–23. System Identification performance on the LoFLYTE system. (a): desired $p$ and GMM-LLM prediction (top), error (bottom); (b): desired $r$ and GMM-LLM prediction (top), error (bottom); (c): ESN on $p$ (top), error (bottom); (d): ESN on $r$ (top), error (bottom).

with six variable at the same time, the whole system is broken up into two parts. And the problem is simplified to modeling the LoFLYTE’s $p$ and $r$ dynamics using input-output data which includes aileron, rudder, $p$, and $r$. Figure 5–22 shows the dynamics of the LoFLYTE using the corresponding control signals. And local linear format of system transfer function can be expressed as

$$x_{n+1} = \sum_{i=0}^{1} A_i^T x_{n-i} + \sum_{i=0}^{1} B_i^T u_{n-i}$$

$$y_{n+1} = x_{n+1}$$

(5–10)
where $\mathbf{x}_n = [p_n, r_n]^T$, $\mathbf{u}_n = [\delta_{a,n}, \delta_{r,n}]^T$, $\mathbf{A}_i, \mathbf{B}_i$ come from LLM $\mathbf{W}_n$. Thus the training data is constructed as

$$\tilde{\mathbf{x}}_n = [p_n, p_{n-1}, r_n, r_{n-1}, \delta_{a,n}, \delta_{a,n-1}, \delta_{r,n}, \delta_{r,n-1}]$$

Through the LoFlyte Matlab model, totally 6000 data are generated. 4000 of the data are used as training data, 2000 are used for system identification testing, and the last 1000 of testing data is used as desire signals for controller testing. A 10-cluster GMM is trained based on the training data set, two sets of LLM are calculated from data referring $p$ and $r$ next-step signal as desire signals. Correspondingly, one 300-PE ESN is constructed for signal “p” and “r” with coupling input and without time embedding. The modeling performance is listed in the table 5–6. Comparing with RBF, GMM is still a comparable and computation efficient method. Again we emphasize here that because of its computational efficiency, GMM-LLM will bring a simple, piece-wise linear platform for nonlinear robust control approach.
Table 5–6. Performance on the LoFLYTE simulation data

<table>
<thead>
<tr>
<th>MODEL (# of PEs)</th>
<th># of Parameters</th>
<th>SER on p (dB)</th>
<th>SER on r (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RBF (100)</td>
<td>908 each</td>
<td>41.33</td>
<td>49.98</td>
</tr>
<tr>
<td>SOM (64)</td>
<td>1536</td>
<td>42.17</td>
<td>50.74</td>
</tr>
<tr>
<td>GMM (10)</td>
<td>330</td>
<td>40.09</td>
<td>50.58</td>
</tr>
<tr>
<td>ESN (300)</td>
<td>600</td>
<td>44.55</td>
<td>53.62</td>
</tr>
</tbody>
</table>

We now study the robust control problem with GMM-LLM local modeling. Based on GLL-LLM, we will apply second order inverse controller and optimal robust controller. When we design the controller, we will assume the minimum coupling between the lateral and longitudinal movement. Here we perform a simulation to control the roll-rate \( p \) and the yaw-rate \( r \) of the LoFlyte by aileron and rudder, setting the elevator to zero and throttle to certain constant. The design of optimal robust controller will follow the detail in pervious chapter. For the inverse controller, once the LLMs are available for \( p \) and \( r \), and their desire values, the second order dynamic inverse controller can be expressed as

\[
\ddot{e}_{n+1} + \lambda_1 \ddot{e}_n + \lambda_2 \dot{e}_{n-1} = 0
\]

\[
\Rightarrow \bar{u}_n = \begin{bmatrix}
u_{\text{aileron},n} \\
v_{\text{rudder},n}
\end{bmatrix} = \begin{bmatrix}
\bar{W}_{p,(a,n)} & \bar{W}_{p,(r,n)} \\
\bar{W}_{r,(a,n)} & \bar{W}_{r,(r,n)}
\end{bmatrix}^{-1}
\]

\[
\begin{bmatrix}
d_{p,n+1} - \bar{W}^T_{x-u,(p,n)} \bar{x}_{x-u,n} + \lambda_1 e_{p,n} + \lambda_2 e_{p,n-1} \\
d_{r,n+1} + \bar{W}^T_{x-u,(r,n)} \bar{x}_{x-u,n} + \lambda_1 e_{r,n} + \lambda_2 e_{r,n-1}
\end{bmatrix}
\]

For different system dimensions, the poles of the dynamic inverse controller are designed as \( p_{1,2} = 0.7 \), which corresponds to second order polynomial coefficients \( \lambda_1 = -1.4; \lambda_2 = 0.49 \). Considering the coupling between the two parameters, an 2nd order inverse control dynamic controller is designed for \( p \) and \( r \) following the algorithm in chapter 4, where the 2nd pole of error dynamic guarantee the convergence of the
error. The cost functions for the optimal robust controller for both dimensions are designed as

\[ Q^1 = \begin{bmatrix} 10 & 0 \\ 0 & 20 \end{bmatrix}; \quad R = 0.05 \]

Table 5–7. Control performance comparison on the LoFlyte system considering noise

<table>
<thead>
<tr>
<th></th>
<th>Set-point Error Var. (p)</th>
<th>Set-point Error Var. (r)</th>
<th>Tracking Error Var. (p)</th>
<th>Tracking Error Var. (r)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverse</td>
<td>0.0802</td>
<td>0.0026</td>
<td>0.1833</td>
<td>0.0060</td>
</tr>
<tr>
<td>Optimal</td>
<td>0.0144</td>
<td>0.0002</td>
<td>0.0322</td>
<td>0.0018</td>
</tr>
</tbody>
</table>

The control mission is divided into set-point regulation which set the system dynamics to a set of fixed values, and arbitrary output tracking, which comes from part of the data collection in previous section. Considering the robustness performance, 30dB SNR noise is added to the dynamics of the system as modeling uncertainty. The comparison of control performance are shown in figure 5–25, 5–26, 5–27, and 5–28, the detail is shown in table 5–7. It is clear that in the noise-free case, both controllers have similar transient and long-term performance. In the first five seconds, we find strong coupling is taking effect such that two parameters are converging at about the same time. In the case of noise, however, the optimal robust controller has obvious advantage over second order inverse controller. As shown in the comparison figures 5–27 and 5–28, control result from optimal robust controller obviously has smaller error variance when 30 dB noise present to both dimensions as modeling uncertainties.
Figure 5–25. Control performance on the LoFLYTE system (set-point, noise free). (a) Inverse controller performance; (b) Optimal robust controller performance. (track of $p$ (top-left), track of $r$ (top-right), control signal $\delta_a$ (bottom-left), and control signal $\delta_r$ (bottom-right))
Figure 5–26. Control performance on the LoFLYTE system (tracking, noise free).
(a) Inverse controller performance; (b) Optimal robust controller performance. (track of $p$ (top-left), track of $r$ (top-right), control signal $\delta_a$ (bottom-left), and control signal $\delta_r$ (bottom-right))
Figure 5–27. Robust control performance on the LoFLYTE system (set-point, 30 dB SNR noise).
(a) Inverse controller performance; (b) Optimal robust controller performance. (track of $p$ (top-left), track of $r$ (top-right), control signal $\delta_a$ (bottom-left), and control signal $\delta_r$ (bottom-right))
Figure 5–28. Robust control performance on the LoFLYTE system (tracking, 30 dB SNR noise).
(a) Inverse controller performance; (b) Optimal robust controller performance. (track of $p$ (top-left), track of $r$ (top-right), control signal $\delta_a$ (bottom-left), and control signal $\delta_r$ (bottom-right))
Local modeling approaches have attracted large interest because they have the ability to explain local dynamics of a nonlinear system, which is difficult especially in the case when nonlinear dynamical system characteristics vary fast throughout the state space. In a number of cases, especially under the *black-box* approach, local modeling has been proven to be a more effective approximation methodology than global approaches.

### 6.1 Summary

This dissertation aims at experiencing dynamical behavior of Gaussian mixture model for modeling dynamics in un-supervised way as well as implementing such methodology as system identification and control infrastructure in supervised way. Through the mixture of Gaussian, MSE based local models are generated so as to simplify the system identification problem and later ease the design mission of system controller with acceptable robust control performance. The first motivation to apply GMM is the distinct characteristics of this model compared with other local modeling methodology. Comparing with SOM-based local modeling approach, which is indeed a *look-up table* modeling, GMM-based modeling method can decrease the computational burden dramatically. The second advantage is that GMM decreases the modeling error from quantization error through *soft-combination*, which marks a fundamental difference between this research with others.
Another important factor of this scheme is the usage of Growing-SOM similar algorithm for the initial distribution calculation of GMM’s weights. With the help of Growing-SOM, the first benefit is that the converge time and speed are globally optimized. And secondly, since the Growing-SOM’s weights and GMM’s mean holding the same physical meaning, GMM is provided a flexible cluster growing framework which is extremely useful when adding more Gaussian clusters is necessary.

Among several LLM-based controllers, optimal robust controller is one of the best results we can obtain. The problem for reference model based prediction control is that when reference model has error, the control will have errors too. Especially when modeling uncertainty and measurement noise present, the control performance will decrease dramatically. Optimal robust controller considers the disturbance effects in a long-term sense, thus decrease the noise effect in the largest way.

6.2 Future Work

From the many applications described in chapter 5, the proposed GMM based local linear modeling methodology appears feasible and easy to realize for a wide range of real nonlinear dynamical system. During this research, however, several questions concerning performance details arose:

1. Besides the conclusion on precision and robustness analysis for controller based on GMM-LLM, adaptive disturbance cancelation approaches should be further explored as model precision is always the prior concern in our modeling-control framework.

2. ESN modeling does not need embedding at input. Yet ESN controller training is still not a straight forward solution for many problems. Stable convergence of ESN controller training will be an interesting question to ask.

3. Optimal local modeling. GMM can decrease quantization error, yet LLM still need to make a tradeoff between the effective region and the curse of dimension.
Nonlinear modeling need more complex training effort and it has mode flexible modeling capability.

4. Other controller design methodology. The accomplished controller design follows the same soft-combination criterion from Gaussian local modeling, which is trained through input space. An innovative attempt could be made to design the controller through input-output space.

6.3 Conclusion

We have presented Gaussian mixture model based robust control strategy for tracking control of general unknown nonlinear dynamical systems. The encouraging results showed that our approaches are promising and easy to be applied in real system identification and control practices. Along with a series of linear controllers, LLM and ESN can be used for a large range of nonlinear dynamical applications where local short term prediction can be simple and successful solutions for complex problems.
APPENDIX A
OPTIMIZING PARAMETERS OF ESN

The current parameter design procedure for ESN follows Jaeger’s stability enhanced with Ozturk’s entropy criterion, which sets spectral radius less than one and maximize entropy for the internal weights. The advantage of the above method is its simplicity, the only major factors we can control are: the number of internal weights and the maximum spectral radius. Based on Ozturk’s approach, the eigenvalues of ESN will be evenly distributed within the circle which is defined by maximum spectral radius. Since we do not have any further control on the internal weights, the optimal ESN performance is largely limited to its size and distribution. Considering the computational cost on readout weights and possible memory limit on certain experiment environment, the ESN’s size cannot be increased beyond reasonable limits. Under the condition of stability, the maximum spectral radius also cannot be set to value larger than “1”. Thus there is one possible modification can be made to the ESN design, the distribution of its spectral radius.

According to the definition, the maximum spectral radius of ESN corresponds to its memory length. When the embedding length requirement is larger than the ESN memory length, it starts to “forget” the input. What we want to improve is to increase the average memory length of ESN and not to increase the related training cost.

From figure 2–2, the weights with largest spectral radius within ESN only account for a small portion of the whole set. We can also understand those locations as poles
of the linearized system. Since the poles of the target system are unknown, we found in experiments that the modeling performance will decrease when there is not “reasonable” amount of ESN poles located near the system poles. We propose here a “dominant pole” approximation to ESN design, i.e. we will place most of the poles near the largest spectral radius. Considering the target system might have several poles (eigenvalue) distributed along the maximum value, and the maximum poles dominate the internal embedding length, larger amount of poles close to maximum eigenvalue could be a good answer for system identification. Also, we found out that when a large portion of ESN poles are close to the maximum spectral radius, the weights entropy will not decrease appreciably compared with the maximum entropy from evenly distributed poles. Meanwhile, in case where the dominant pole is a reasonable approximation, ESN modeling performance is also improved. Comparing
figure 2–2 and figure A–1, we can find that there are more than 80% of poles in figure A–1 equal to the maximum spectral radius.

The realization of such optimized ESN weights matrix could be easily achieved through multiple root design. First, we design a regular ESN, then calculate the polynomial of its eigenvalues.

\[ s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n = (s - r_1) \cdots (s - r_n) = 0 \]

Setting \( a_i = 0 \), for a set of \( i \) \((i > d)\) in order to make the polynomial have multiple complex roots with identical magnitude and different phase. And the new polynomial \( s^n + a_1 s^{n-1} + \cdots + a_d s^{n-d} = 0 \) will determine how many poles are complex and their magnitudes are close to maximum eigenvalue. In order to keep the poles of the new polynomial as the poles of ESN, we re-arrange the weights matrix as

\[
W = \begin{bmatrix}
a_1 & a_2 & \cdots & a_n \\
1 & 0 & \cdots & 0 \\
0 & \ddots & \ddots & \vdots \\
0 & \cdots & 1 & 0
\end{bmatrix}
\]

Finally, we normalize the new matrix with the desired maximum spectral radius. The new ESN will follow the distribution as shown in figure A–1. The new matrix will hold the same eigenvalues from the polynomial, and since we can design the roots of polynomial, we “design” the ESN’s internal weights. One possible problem for the modified ESN design is that it may not be appropriate for all dynamic system modeling problems. In cases where the dominant pole approximation is valid should yield better performance. For instance, when the task is limited by long term internal memory, the new criterion is appropriate. In cases where the design requires modeling
of different time constants, the proposed ESN’s modeling capacity might decrease versus the old approach.
APPENDIX B
ESN-BASED DAC CONTROLLER

We have discussed the modeling property of ESN in chapter 2. Considering its RNN-related modeling capability, ESN can act as both model approximator and controller as shown in figure B–1. When ESN is designed for system identification, as we will show in the following sections, it can make short term prediction easier than GMM-LLM framework. And ESN’s modeling performance will not be worse than those of Gaussian modeling because of its structural advantages. In terms of model description simplicity, ESN is not an optimal choice for system identification. Under the local modeling control framework, LLM gives the simplest linear description of system dynamics. Through ESN modeling, however, it will take much more effort to back-propagate the model information from ESN internal weights. In some cases, back-propagated information may diverge from the actual system dynamics, and the reference model loses its local modeling capability. Most of the time, lower order estimation of the nonlinear model will not satisfy the precision requirement. Consequently, we will focus on ESN-based controller design in this section.

Currently there are limited papers discussing applications of ESN to control [32, 33]. Structurally, ESN control can be grouped to a set of neural control. In both papers, the ESN controller is trained when desire control signal is available, thereby reducing the control problem setting to that of the standard modeling problem. In other words, ESN is trying to model other controller instead of control the target
system. Though in some applications inference of desire control signal might be done, in many others such inference is either infeasible or unreliable [34].

Model based control is a real challenge for ESN. The main difficulty comes from the requirement that desire controls be known as used for ESN training. The desire control values can sometimes be inferred from the desire outputs if a transformation from the controls to the outputs has a single-valued inverse and the relative degree one. However, the plant model might not be inverted analytically. Further more, calculating desire control signal numerically might be time consuming, brittle and finally converge to local minima. Generally speaking, determining desire control signals should be replaced by the use of back-propagation through time (BPTT) to compute the sensitivities of plant model outputs to the changes of its control inputs. The incurred additional computational complexity may all but nullify the main advantage of the ESN.

The alternative scheme of indirect control with ESN would place ESN under the method of derivative adaptive critics (DAC) such as dual heuristic programming

Figure B–1. Demonstration system with ESN reference model and controller system.
DAC is derived from adaptive critic designs (ACD) which includes heuristic dynamic programming (HDP), DHP, and globalized dual heuristic programming (GDHP), etc. Prokhorov in [3] showed that there exists similarity between a particular form of BPTT and the popular DAC formulation for the general neural networks case. The similarities enable us to create a hybrid BPTT and ESN based DAC in which DAC provides derivatives of estimation from the future time steps.

We sketch an alternative scheme of indirect control with ESN. Without linearizing the system function, many control systems can be expressed in controls $u_n$:

$$
x_{n+1} = f(x_n, u_n) = H(x_n) + G(x_n)u_n \tag{B–1}
$$

where $H$ is the vector function, $G$ is the control matrix. Setting the instantaneous performance measurement as $(4–19)$ with $Q > 0, R > 0$, then implementable parameterization of the controller maybe expressed using ESN with output vector $\lambda$ as the following product

$$
u_n = -R^{-1}G^T(x_n) \cdot \lambda(x_n, W_{out}) \tag{B–2}
$$

The discrete close-loop system will run forward, in the BPTT way, with the sequence of control signals $\{u_n\}_{n=n_0}^{n_{max}}$ provided by the ESN controller with constant initial vector $W_{out}$. In order to train the ESN’s readout vector, we employ the dual equations as desire ESN output

$$
\lambda_{desire,n} = Q \cdot x_n + ((\partial f)/\partial x_n)^T \lambda_{desire,n+1} \tag{B–3}
$$
by going backwards from \( n = n_{\text{max}} \), where we set the end point \( \lambda_{\text{desire}, n+1} \), to the starting point \( n = n_0 \). The ESN readout vector training errors can be derived from ESN output as
\[
e_{\lambda} = \lambda_{\text{desire}, n} - \lambda(x_n, W_{\text{out}}) \tag{B-4}
\]

The error is feed into MSE training criterion to update the ESN readout weights. The whole procedure is set as: running the simulation forward to collect data (system outputs and ESN outputs); going backward to update \( W_{\text{out}} \) with errors (B-3) and (B-4), which is repeated till the converge of \( W_{\text{out}} \). Preliminary experiments indicate that the ESN based derivative adaptive critic is viable, especially for control of distributed parameter systems.

As for the model information of \( H \) and \( G \), ESN based approach has two options. One is the way like other controllers we discussed in previous sections in which GMM-LLM supplies piece-wise local linear estimation. The GMM-LLM part will bring additional computation requirement, yet the whole framework will be designed based on a more traditional style as show in figure B-2. Another option is using one ESN for both system identification and control. In this way the ESN will have single column of input and two independent readout weight matrix corresponding to state prediction \( \tilde{y}_{n+1} \) and control cost prediction \( \lambda_{n+1} \). For the state prediction part, ESN model function can be re-written from (2–10)
\[
\tilde{y}_{n+1} = W_{\text{out}}^{T} \cdot f[W_{\text{in}} \cdot x(n) + W \cdot s(n)] \tag{B-5}
\]
Figure B–2. Schematic diagram of control system with ESN as controller based on Gaussian models

Considering the small random values of input weights \( W_{in} \), the state prediction could be locally modified as first order Taylor-expansion

\[
\tilde{y}_{n+1} = W_{out}^T \cdot [f(Ws(n)) + W_{in}(n)\hat{f}(W \cdot s(n))X(n)] \quad (B-6)
\]

Considering the \( X(n) = [x(n), u(n)]^T \), and

\[
f(x) = tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})
\]

\[
\hat{f}(x) = \frac{1}{cosh^2(x)}
\]

The local model information of \( H \) and \( G \) can be derived from (B–6) as

\[
[H, G]^T = W_{out}^T \cdot W_{in}/cosh^2(W \cdot s(n))
\]

We notice that the calculation for the local model information \([H, G]\) is mainly based on the ESN parameters. First order estimation can not guarantee the precision
of the reference morel yet that kind of calculation will not bring too much computational cost. Considering the high modeling efficiency of GMM-LLM framework, we will make use of model information from GMM-LLM for ESN controller in the following chapter.
REFERENCES


BIOGRAPHICAL SKETCH

Jing Lan was born in Beijing, People’s Republic of China, on June 16, 1973. He received his B.S. degree from Beijing University of Aeronautics and Astronautics, majoring in automatic control, in 1996. He received his M.S. degree at Tsinghua University, majoring in automation, in 1999. Since 2001, he has been with the Computational NeuroEngineering Laboratory at the University of Florida to pursue his Ph.D. degree. His research interests include non-linear system identification, Gaussian mixture model prediction, and neural networks and adaptive filters on signal processing.