A Data-Driven Approach for Modeling, Analysis and Control of Stochastic Hybrid Systems using Gaussian Processes

By

Hamzah Abdelaziz

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical Engineering

January 31, 2018

Nashville, Tennessee

ACKNOWLEDGMENTS

All praises are due to God alone, the most gracious, the most merciful, for providing me with the strength, and the patience to complete this work.

I would like to express my gratitude to my parents and family for their endless support from the very beginning and their wonderful care and encouragement.

I also would like to express my deepest gratitude to professor Xenofon Koutsoukos, my advisor, for his excellent guidance, advice, and encouragement towards successful completion of this work. I am thankful for his trust, time and continuous support. My research would not have been possible without his help.

I am also sincerely grateful to professors Aniruddha Gokhale, Gautam Biswas, Alan Peters and Mitchell Wilkes for serving as my Ph.D. committee and for their valuable comments on my work.

Last but not least, I would like to thank many members of the Institute for Software Integrated Systems and Vanderbilt community, specifically, Shashank Shekhar, Shweta Khare, Anirban Bhattacharjee, Faruk Caglar, Zhenkai Zhang, Siyuan Dai, Bradley Potteiger, Amin Ghafouri, Nasser Alshammary and others, for the great professional, collaborative, enjoyable and friendly work environment they created during my time at Vanderbilt.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Motivation

The growing advances in information technologies have led to a new generation of systems known as cyber-physical systems (CPS). Cyber-physical systems are multi-discipline engineering systems which integrate computational and physical processes [10]. Advances in CPS change the way people interact with the physical systems through smart computation, communication and control algorithms such as self-driving urban vehicles and smart buildings. Great opportunities and research challenges arise to enable the development of new CPS technologies that can incorporate complex computational algorithms (e.g., learning and control) with complex physical systems [82]. Moreover, the developments in sensor technologies enable the utilization of data to build robust models of complex physical systems. These robust models are needed to achieve intelligent CPS and to automate their design and control processes.

Model-based design of CPS abstracts the system physical processes using dynamical models. These dynamical models support the integration of the cyber and the physical parts with model-based analysis and control methods. Typically, model-based design of CPS includes a computing system, a physical system and the environment as shown in Figure 1.1. The computing system observes sensory data from the physical system and generates control signals through computational processes to drive the physical system behavior. Additionally, the computing system uses the observed data to identify a system model (i.e., model learning) that represents the system behavior. Thus, the identified model can be used to generate, optimize or verify the control signals in effective and efficient manner. The environment also affects the physical system behavior; however, its effect cannot be controlled. Typically, this effect can be represented as a disturbance in order to consider the impact of the environment on the physical system. Identifying a model of this disturbance is very useful in order to increase the model accuracy.

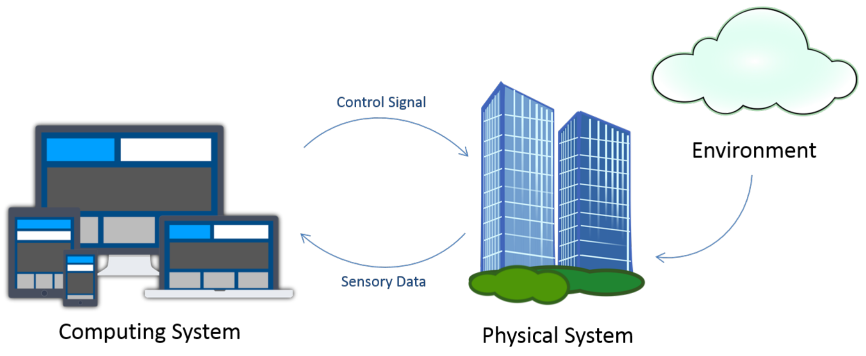


Figure 1.1: Model-based design and control

There are different types of modeling frameworks for representing CPS. One end of the spectrum is white-box models, also known as parametric models. The other end is black-box models which are also known as non-parametric models. White-box models are mathematical models where the system behavior is described through formalisms such as ordinary differential equations (ODEs). In this case, the model structure is often assumed to be fully known and the modeling problem lies on identifying the model parameters. There exist many identification algorithms and techniques for such models which are well-established and mature especially for linear systems [92]. However, many modern systems are very complex and their behavior is characterized by nonlinearity, time-variability, and uncertainty. Identification of such systems is more challenging since their model structure is usually hard to obtain especially when a detailed model is required. As a result, black-box models have drawn considerable attention as an alternative where the system behavior is learned from observed data without the need of domain knowledge about the system structure and parameters. Various methods based on machine learning exist for modeling such complex systems [12]. Machine learning techniques offer attractive properties for learning stochastic models based on data and use them to control complex systems autonomously. Furthermore, these techniques are data-driven, and therefore, they enable us to automatically learn models of nonlinear systems in the present uncertainty from the observed data [46].

Stochastic hybrid systems (SHS) are models that include coupled continuous and discrete dynamics and can be used to represent complex stochastic systems [41]. SHS have several modes of operation such that the continuous dynamics behave differently in each mode. This multi-modal behavior adds another level of complexity, especially when both continuous and discrete dynamics exhibit stochastic behaviors. These complexities increase the difficulty of identifying parametric models for such systems. Therefore, data-driven methods based on machine learning techniques are a promising alternative that requires investigation.

Modern smart buildings can be considered as an example of complex CPS where the integration of data sensing and control systems is used to achieve user comfort and to promote energy efficiency. Model-based design of smart buildings requires accurate models of the buildings thermal dynamics. However, buildings have complex stochastic nonlinear thermal dynamics which are affected externally by the environment and internally by interactions between buildings zones. Additionally, the thermal dynamics of buildings behave differently based on the thermal load, e.g., occupancy in buildings. Buildings parameters may change over time as the buildings age or may change abruptly due to events such as opening/closing windows. Such challenges can be addressed using nonparametric modeling approaches based on online model learning because accurate parametric models of smart buildings are hard to obtain.

1.2 Research Challenges

Stochastic hybrid systems (SHS) can model many CPS with complex behaviors in order to analyze and control them efficiently. Accurate modeling of these complex behaviors using parametric models is a major challenges and may not be feasible. Data availability can potentially support the use of machine learning techniques to learn complex SHS. These challenges necessitate the use of machine learning techniques because of their ability to extract information about systems and the environment from sensory data. However, there are many research challenges that arise for using machine learning techniques to learn data-driven SHS models. Additionally, different research opportunities and challenges exist to utilize data-driven SHS models to develop efficient reachability analysis and control methods. In this work, we focus on mitigating various of these challenges which are related to model learning, reachability analysis, and control of data-driven SHS.

Model Learning of SHS

Model learning of SHS aims to identify the continuous and the discrete dynamics from sensory data. However, the sensory data in many CPS do not necessary include explicit information about discrete states of the system (i.e., the discrete state are latent). In this case, identifying the latent discrete dynamics from data becomes a major challenge that needs to be addressed. Also, identifying the latent discrete dynamics is required in order to segment the data for each corresponding discrete state, so that, each data segment can be used to identify the corresponding continuous dynamics. Model learning of SHS faces another challenge when the parameters change over time. The model learning algorithm should adapt to this variability in the system while satisfying timing constraints.

As mentioned earlier, the physical processes of many CPS are usually affected by the environment. Modeling the effects caused by the environment is essential to achieve better accuracy of the system model. Therefore, an appropriate environment model should be learned and integrated with the SHS modeling framework.

Reachability Analysis

The objective of reachability analysis is to predict the reachable states of SHS [15]. Reachability analysis can be used to compute the probability that the system states will stay within a certain safe region. In SHS, such prediction presents a major challenge because of the interleaved stochastic discrete/continuous dynamics, so that the prediction may result in different trajectories with different likelihoods. Typical reachability analysis algorithms are based on Monte Carlo simulation and can address this challenge by predicting the system trajectories [15]. However, algorithms based on Monte Carlo simulation are not feasible for many systems because they may require running large numbers of simulations. Therefore, another reachability analysis methodology is required to predict the SHS reachable states and to express their probability by a suitable distribution in an efficient online fashion.

Decision Making and Control of Stochastic Dynamical Systems

A controller of a stochastic system should deal with the system uncertainty in order to achieve optimal and robust performance. Typically, the controller should optimize a cost function subject to deterministic and stochastic constraints. Stochastic model predictive control (SMPC) is a popular control methodology that can achieve desired performance under constraints. SMPC is based on optimizing the cost function for a receding finite horizon without violating the constraints [33]. In this work, we develop a stochastic model predictive control (SMPC) methodology for data-driven SHS models. The stochasticity exhibited in both discrete/continuous dynamics of SHS and the system constraints cause the main technical challenges. The SMPC optimization problem is very difficult, and therefore, approximation algorithms are required. Many approximation algorithms have been established for continuous stochastic systems modeled using parametric models, however, new algorithms are needed to address systems with data-driven models and interleaved stochastic discrete/continuous dynamics as the case in SHS.

1.3 Summary of Contributions

In view of these challenges, this dissertation presents the following contributions.

- Model Learning of SHS

    1. We present a nonparametric SHS model based on Gaussian processes and pe-

riodic Markov chain. Gaussian processes capture the stochastic nonlinear continuous dynamics for different discrete states and the periodic Markov chain captures the periodic transitions of the discrete states.

2. We present an online clustering-based learning methodology for SHS when the discrete dynamics cannot be measured excitability. We use the clustering algorithm (e.g., K-means) to identify the discrete states of the system and to label the training data. Thus, we can learn the transition probabilities of the Markov chain and segment the data for each corresponding discrete state. Each segment is then used to learn the continuous dynamics using GPs.

3. We evaluate the efficiency of the online learning methodology, so the model adapts to time-varying changes in the physical system parameters.

4. We utilize the data-driven modeling approach to learn efficiently an accurate thermal model of multi-zone buildings when the buildings thermal load is latent (e.g., occupancy) using data generated by Energy-Plus (high-fidelity building simulator) and a stochastic occupancy simulator.

- Reachability Analysis of SHS

  1. We present a finite-horizon reachability analysis algorithm to estimate statistically the probability distribution of the reachable states based on mixtures of Gaussian processes.

  2. We provide in depth study of the performance and the efficiency of the reachability analysis algorithm for smart buildings applications.

  3. We also consider a special use case of SHS when the discrete dynamics of the system are deterministic and known.

- Stochastic Model Predictive Control of SHS

1. We present a scenario-based stochastic model predictive control algorithm to provide an optimal control algorithm of complex systems modeled by data-driven SHS. The presented algorithm is based on the analytic calculations of the model gradients which enhance the accuracy and the efficiency of the optimization routines.
2. We evaluate the performance and the efficiency of the proposed SMPC for smart buildings. The proposed approach is used to control the HVAC unit to minimize the energy consumption without violating user comfort.

1.4 Organization

The rest of this document is organized as follows:

- Chapter 2 summarizes the necessary background in model learning and prediction using Gaussian processes and also illustrates related work in the research of stochastic systems using GPs and SHS.
- Chapter 3 presents the data-driven SHS modeling paradigm. A case study of modeling the thermal models of buildings is presented to demonstrate the performance of the proposed approach.
- Chapter 4 presents the proposed reachability analysis algorithm of stochastic hybrid systems based on mixtures of Gaussian processes and demonstrates the performance of its multi-step prediction algorithm in smart buildings application.
- Chapter 5 presents the scenario-based stochastic model predictive control of data-driven SHS and illustrates the performance of the approach using smart buildings application.
- Chapter 6 concludes this dissertation with a summary of the dissertation overall content and some suggestions of potential future work.

Chapter 2

Background and Related Work

There are different technical areas related to data-driven SHS for modeling CPS. In this chapter, we summarize the necessary theoretical background and related work that are relevant to our research in developing model learning, reachability analysis and model predictive control of SHS using Gaussian processes. The organization of this chapter is as follows. Section 2.1 summarizes the theory of Gaussian processes for machine learning and Markov chain models. Afterward, Section 2.2 presents a review of model learning approaches developed for stochastic systems using Gaussian processes. It also presents the model learning approaches developed for SHS. Section 2.3 discusses reachability analysis techniques developed for SHS and illustrates the related work for state prediction of stochastic systems for a finite horizon. Section 2.4 provides an overview of existing methods used to solve stochastic model predictive control of systems with uncertainty. Finally, Section 2.5 reviews related work for model-based design and control of smart buildings.

2.1 Background

2.1.1 Gaussian Processes

A Gaussian process (GP) is a nonparametric probabilistic model that requires only high-level knowledge about the system behavior and uses the observed data to model the behavior of the underlying system [84]. Generally, a GP builds a Gaussian distribution over functions, by which it maps a function index variable to an infinite-dimensional function space.

Definition 1. *(Gaussian process). A Gaussian process is a collection of random variables, any finite number of which have a consistent joint Gaussian distribution [84].*

A GP is identified by its mean and covariance functions. The mean function represents the expected value before observing any data and the covariance function (also called kernel) identifies the expected correlation between the observed data. For a function $y = f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^D$, the mean function $m(\mathbf{x})$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$ are defined as:

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \end{aligned} \tag{2.1}$$

The function modeled by the GP can be written as:

$$f(\mathbf{x}) \sim \mathscr{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')).$$

We typically use a zero mean function for simplicity and squared exponential (SE) covariance kernel for its expressiveness combined with a noise kernel. Therefore, the mean and covariance functions can be expressed as:

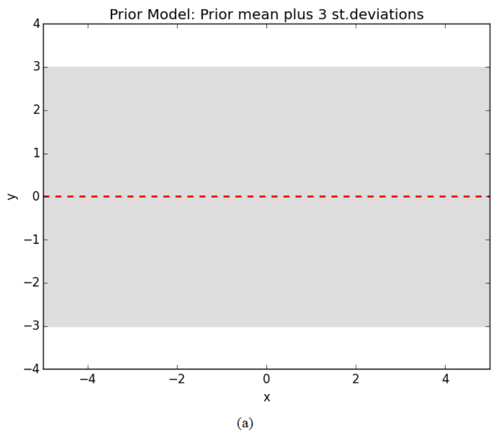
$$\begin{aligned} m(\mathbf{x}) &= 0, \\ k(\mathbf{x}, \mathbf{x}') &= \sigma_f^2 exp[-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T \Lambda^{-1} (\mathbf{x} - \mathbf{x}')] + \delta_{\mathbf{x},\mathbf{x}'} \sigma_\omega^2 \end{aligned} \tag{2.2}$$

where σ_f is the kernel signal variance, $\Lambda := diag([l_1^2, \cdots, l_D^2])$ is the characteristic length-scales matrix, δ is the Kronecker delta, and σ_ω is the noise variance. The above GP model builds a probability distribution over the functions $p(f(\mathbf{x}))$ by mapping n-samples $\mathbf{X}$ of a continuous variable $\mathbf{x}$ to a vector of random variable $\mathbf{f}$ with a Gaussian joint distribution, such that:

$$p(\mathbf{y}) \sim \mathscr{N}(0, \mathbf{K}(\mathbf{X}, \mathbf{X})) \tag{2.3}$$

where $\mathbf{K}$ is nD-by-nD covariance matrix generated by (2.2). Figure 2.1 shows an example of the prior distribution of the GP in (2.3).

We are interested in the GP posterior distribution given some test inputs and observations (training data). We define the set of test inputs where we want to predict the function



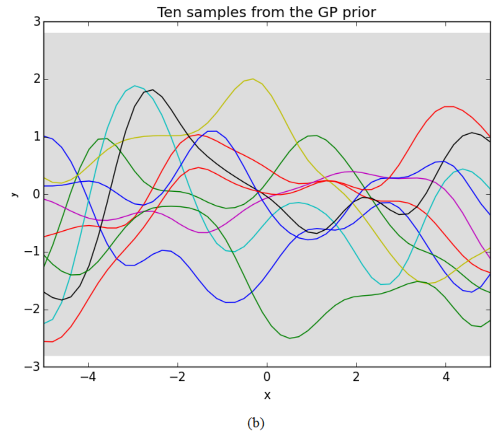


Figure 2.1: Gaussian process prior model of the underlying function before observing any data (a) prior distribution which is constant along the function dimension (b) samples function drawn from the prior distribution

value as $\mathbf{X}_*$. After observing data $\mathscr{D}$, and according to (2.3), the joint distribution of the known $\mathbf{y}$ and the unknown $\mathbf{y}_*$ function values is:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}_* \end{bmatrix} \sim \mathscr{N}\left(0, \begin{bmatrix} K(\mathbf{X},\mathbf{X}) & K(\mathbf{X},\mathbf{X}_*) \\ K(\mathbf{X}_*,\mathbf{X}) & K(\mathbf{X}_*,\mathbf{X}_*) \end{bmatrix}\right).$$

Therefore, the posterior distribution $p(\mathbf{y}_*|\mathbf{X}_*,\mathbf{X},\mathbf{y})$ is also a conditional Gaussian distribution with a mean and a covariance given by:

$$\begin{aligned} \mathbb{E}[\mathbf{y}_*|\mathbf{y},\mathbf{X},\mathbf{X}_*] &= \mathbf{K}_*^T\boldsymbol{\beta} \\ Var[\mathbf{y}_*|\mathbf{y},\mathbf{X},\mathbf{X}_*] &= \mathbf{K}_{**} - \mathbf{K}_*^T(\mathbf{K}+\sigma_\omega^2\boldsymbol{I})^{-1}\mathbf{K}_* \end{aligned} \tag{2.4}$$

where $\mathbf{K}_* := k(\mathbf{X},\mathbf{X}_*)$, $\mathbf{K}_{**} := k(\mathbf{X}_*,\mathbf{X}_*)$, $\mathbf{K} := k(\mathbf{X},\mathbf{X})$ and $\boldsymbol{\beta} := (\mathbf{K}+\sigma_\omega^2\boldsymbol{I})^{-1}\mathbf{y}$.

Figure 2.2 depicts an example of the GP posterior distribution in (2.4).

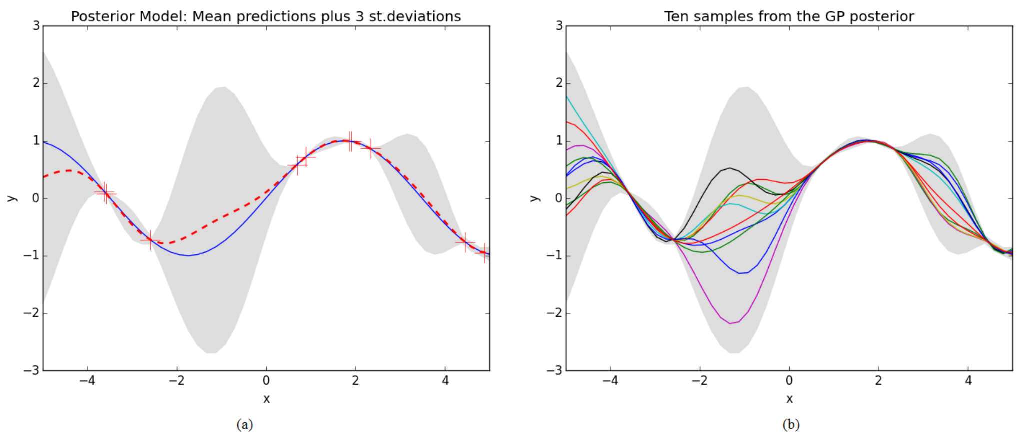


Figure 2.2: Gaussian process posterior model of the underlying function after observing data (a) posterior distribution along the function dimension (b) samples function drawn from the posterior distribution

2.1.1.1 Model Learning

In previous section, we determine the GP prior/posterior distribution given the set of hyperparameters defined as $\Theta := (\sigma_f, l_1^2, \cdots, l_D^2, \sigma_\omega)$. However, the hyperparameters are not usually known a prior. Figure 2.3 shows the effect of varying the hyperparameters on the GP posterior distribution and its variance represented by the gray region. For instance, the GP posterior becomes smoother as the lengthscale parameter in (2.2) increases. Therefore, we need to learn the model hyperparameters that best represent the training data. To that end, our objective is to learn the hyperparameters vector for a given set of observations $\mathscr{D} = \{(\mathbf{x}_i, y_i) | i = 1, ..., n\}$ as the training data. The learning process can be seen as an optimization problem, where the optimal hyperparameters ($\hat{\Theta}$) maximize the marginal likelihood given by.

$$\begin{aligned}\hat{\Theta} &= \arg\max_{\Theta} \log p(\mathbf{y}|\Theta, \mathscr{D}) \\ \log p(\mathbf{y}|\Theta, \mathscr{D}) &= -\frac{1}{2}\mathbf{y}^T\mathbf{K}\mathbf{y} - \frac{1}{2}log(|\mathbf{K}|) - \frac{n}{2}log(2\pi).\end{aligned} \tag{2.5}$$

An effective algorithm based-on conjugate gradients has been developed to optimize GPs hyberparamters [84, 58]. This algorithm optimizes the hyperparameters by using conju-

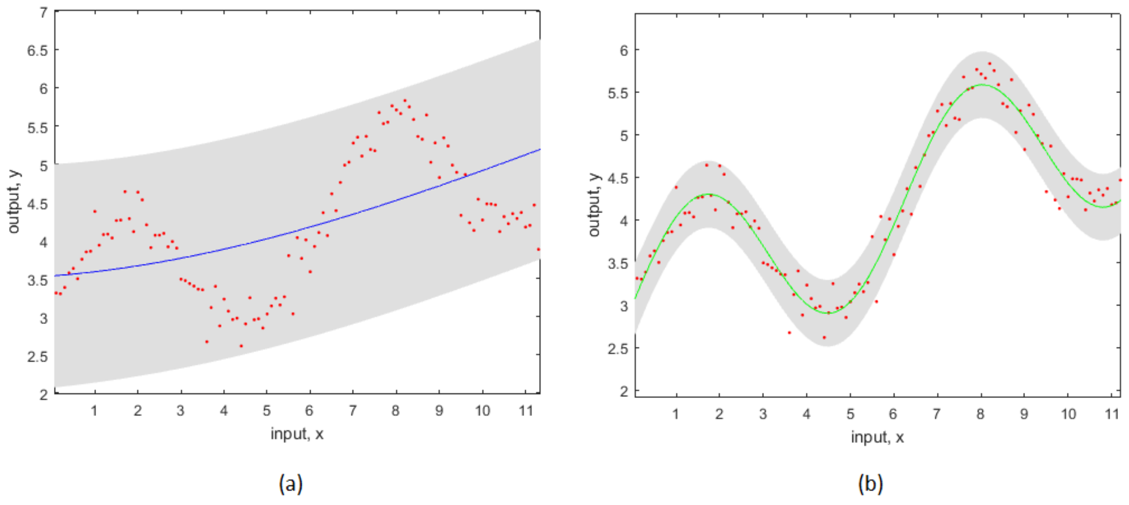


Figure 2.3: The effect of varying GP hyperparameters (a) a GP model with hyperparameters $\Theta := (18.3, 3.1, 0.71)$, (b) a GP model with hyperparameters $\Theta := (2.9, 3.9, 0.19)$

gate gradients and approximating linesearches based on polynomial interpolation. Also, the popular quasi-Newton optimization method has been used to learn the GPs hyperparameters effectively.

2.1.1.2 Prediction at Certain Input

The posterior distribution shown in (2.4) is a prediction model of the model output $\mathbf{y}_* \in \mathbb{R}$ at a given certain test input(s) $\mathbf{X}_* \in \mathbb{R}^D$. For a system with multivariate targets $\mathbf{y}_* \in \mathbb{R}^E$, we model each target $\mathbf{y}_*^i \in \mathbb{R} : i = [1, \cdots, E]$ with a GP independently from the other targets given the test input. Thus, the system is modeled with E independent GPs.

2.1.1.3 Prediction at Uncertain Input

The posterior distribution shown in (2.4) is a prediction model for a given test input $\mathbf{X}_*$. However, this equation is not valid if $\mathbf{X}_*$ is defined by a probability distribution. For instance, if the test input is defined as a Gaussian joint distribution (i.e. $p(\mathbf{X}_*) \sim \mathcal{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$), the GP posterior distribution is calculated by:

$$p(\mathbf{y}_*) = \int\int p((\mathbf{y}_*|\mathbf{X}_*)p(\mathbf{X}_*)d\mathbf{y}_* d\mathbf{X}_*. \tag{2.6}$$

The prediction distribution shown in (2.6) is analytically intractable [37] but it can approximated as Gaussian (i.e $p(\mathbf{y}_*) \sim \mathcal{N}(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)$) using several approximation methodologies based on the law of iterated expectations and conditional variance such as the linearizion approximation introduced in [37] and moment matching approximation introduced in [24] [19]. In this dissertation, we use the linearization approximation, which approximates the mean and variance of the predictive distribution as:

$$\begin{aligned}
\boldsymbol{\mu}_y &= \mathbb{E}[\mathbf{y}_*|\boldsymbol{\mu}_*] \\
\boldsymbol{\Sigma}_y &= Var[\mathbf{y}_*|\boldsymbol{\mu}_*] + \mathbf{V}\boldsymbol{\Sigma}_*\mathbf{V}^T + cov[\mathbf{y}_*, \mathbf{X}_*] + cov[\mathbf{X}_*, \mathbf{y}_*]
\end{aligned} \tag{2.7}$$

where $\mathbb{E}[\mathbf{y}_*|\boldsymbol{\mu}_*]$ and $var[\mathbf{y}_*|\boldsymbol{\mu}_*]$ is the mean and covariance of the GP posterior calculated at the mean $\boldsymbol{\mu}_*$ of the input distribution as in (2.4) and $cov[\mathbf{X}_*, \mathbf{y}_*]$ is the cross-covariance between the input and output and it is given by $\boldsymbol{\Sigma}_*\mathbf{V}$ where $\mathbf{V}$ is defined by:

$$\mathbf{V} = \frac{\partial \boldsymbol{\mu}_y}{\partial \boldsymbol{\mu}_*} = \beta^T \frac{\partial k(\mathbf{X}, \boldsymbol{\mu}_*)}{\partial \boldsymbol{\mu}_*}$$

2.1.2 Markov Chains

Markov chains is a random sequence with a set of discrete states, such that the probability of each state depends only in the previous state, (i.e., $p(z_{k+1}|z_k, z_{k-1}, \cdots, z_{k-n}) = p(z_{k+1}|z_k)$) [94]. A state-transition diagram of a Markov chain model with two discrete states is shown in Figure 2.4.

Typically, a Markov chain has a finite discrete state space, and therefore, the transition

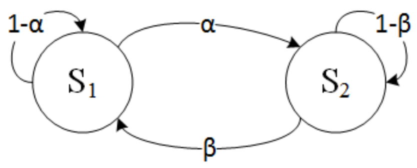


Figure 2.4: A State-transitions diagram of a 2-state Markov chain model with states $\{S_1, S_2\}$ and transition probabilities $(\alpha, 1-\alpha, \beta, 1-\beta)$

probabilities can be represented with a matrix $\mathbf{A} = \{a_{ij}\}$:

$$a_{ij} = P(z_{k+1} = S_i | z_k = S_j)$$

For example, the transition matrix for the example depicted in Figure 2.4 is:

$$\mathbf{A} = \begin{bmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{bmatrix}$$

Markov chain model provides the probability of a sequence of the discrete states, such that the probability of the n^{th} element from p_0 is:

$$\begin{aligned} p_1 =& p_0 A \\ p_2 =& p_1 A \\ =& p_0 A^2 \end{aligned} \tag{2.8}$$

Therefore,

$$p_n = p_0 A^n$$

where p_k denotes $p(z_k | z_{k-1})$. The objective of learning a Markov Chain is to infer the value of the transition matrix entries (i.e., a_{ij}) from a sequence of n observations ($\mathscr{O} = O_0, O_1, O_2, \cdots, O_n$). In a typical case, the transition matrix entries can be learned according to the following formula:

$$a_{ij} = \frac{\text{total number of } O_i O_j \text{ occurrence}}{\text{total number of } O_i \text{ occurrence}} \tag{2.9}$$

2.2 Model Learning

The objective of model learning is to identify a system from observed data. Typically, model learning algorithms use experimental data-sets to build a function that maps the system inputs to outputs. This function can be learned through a wide variety of regression algorithms such as Bayesian locally weighted regression [9] and time-dependent linear models [56] [54]. However, the main drawback of using typical regression methods is the lack of expressing the learned model quality and the uncertainty, especially when there are only few training data available. Therefore, probabilistic regression techniques are used to mitigate the limitation of typical regression methods. Probabilistic regression techniques provide a probabilistic approximation over the learned function to express the model confidence and uncertainty. Gaussian processes are a popular probabilistic regression technique with attractive features and they have been used widely for systems identification [46]. In this section, we provide a review of model learning methodologies for different types of systems. First, we describe related work that has been done for continuous systems focusing on Gaussian processes based techniques. Second, we present model learning related methods that have been used for stochastic hybrid systems.

2.2.1 Model Learning of Continuous Systems using Gaussian Processes

Systems with continuous dynamics can be modeled in different ways based on the system type and the required level of abstraction, for example, time-series models or state-space models. This section presents related work for modeling continuous systems using time-series models, state-space models and multi-modal models based on Gaussian processes. Also, online model learning methodologies using Gaussian processes are discussed.

2.2.1.1 Time-Series Models

Let's define a time-series model for an observed time-dependent variable y_t at time t as:

$$y_t = f(x_t) + \omega_t$$

where f is an unknown function, and $\omega \sim \mathcal{N}(0, \sigma_\omega)$ is typically white noise. The structure of the time-series input (x_t) can be determined based on one of the following approaches: *function mapping* or *curve fitting* [87]

The *function mapping* approach maps observed time-dependent quantities to the time-series output y without reference to time explicitly. For example, the input of a time-series model can be chosen as a previous observation of y_t (e.g., $x_t = y_{t-1}$), thus the model can be expressed as $y_t = f(y_{t-1}) + \omega_t$. This model structure is useful in many applications to build a simple short-term forecasting model. For instance, one-step ahead and two-step ahead short-term traffic volume forecasting models are proposed in [101], where Gaussian processes are used to identify f. Both proposed models have the same inputs which are the n previous traffic volume observations. Formally, the one-step and two-step forecasting models can be expressed as $v_{t+1} = f(v_t, v_{t-1}, \cdots v_{t-n})$ and $v_{t+2} = f(v_t, v_{t-1}, \cdots v_{t-n})$ respectively, (where v_t is the observed traffic volume at time t). Time-series models based on *function mapping* can also incorporate additional knowledge from different dependent time-series variables by including the dependent variables in the model input x. For instance, a short-term load forecasting model for power systems using Gaussian processes is proposed in [64]. The load forecasting model inputs are chosen to include the current observation of the power system load along with other dependent variables such as ambient temperature.

Despite the simplicity of the *function mapping* approach, there are two main drawbacks that limit its usage. First, the *function mapping* approach cannot model time-domain features in the model (e.g., time-series variables with a periodic pattern over time) because of the implicit modeling of time dependency. Second, the *function mapping* approach requires the data to be sampled at a fixed rate and it cannot model time-series variables with missing samples or varying sampling rate.

As an alternative, the *curve fitting* approach assumes that y is ordered by x which typically represents time, (i.e., $y_t = f(t)$). This approach has many applications such as filtering, smoothing, and prediction because of the explicit representation of time dependency. For example, a time-series model of tide height as a function of time is developed in [72] using Gaussian processes with periodic and smoothing components represented by Matern kernel [84]. This model is then used to predict and smooth the missing data along the time axis successfully.

Multi-Output Time-Series Models

Another attractive feature of *curve fitting* is the ability to model multiple time-series with weighted correlations between them. *Multi-output Gaussian processes*, proposed in [72], are used to model multiple correlated time-series variables. This model represents the cross-correlation between the time-series variables with additional hyperparameters in the GP kernel function such that:

$$k([l,t],[l',t']) \triangleq k_L(l,l')k_T(t,t')$$

where t is time index, k_T is a typical GP kernel as illustrated in Equations (2.1) and (2.2), and k_L determines the cross-correlation weight between two different time-series with label l and l', such that it equals unity if both inputs from the same time series (i.e., $l = l'$) and α otherwise.

Multi-output Gaussian processes models are very useful in many applications where there are correlated time-series data. For instance, multi-output Gaussian processes have been used to model a wireless sensor network in order to predict the missing data for a failed sensor [72]. Also, the study in [87] shows that the model quality and performance of multi-output Gaussian processes time-series models with missing data are similar to single independent time-series models with complete data. Another application that uses multi-output Gaussian processes to model a building ambient temperature has been proposed in [88]. In this case, there are two correlated time series: The building ambient temperature, and the weather forecast of the building's city. The correlation between the two time-series can also be approximated alternatively by modeling the difference between both series using a single Gaussian process model only, (i.e., $y = y'_l - y_l = f(t)$).

2.2.1.2 *State-Space Models*

State-space models provide a useful modeling framework for many dynamical systems. Typically, the successor system state $\mathbf{x}_{t+1}$ depends on the predecessor system state ($\mathbf{x}_t$) and an applied control input ($\mathbf{u}_t$). A graphical representation of such systems is shown in Figure 2.5.

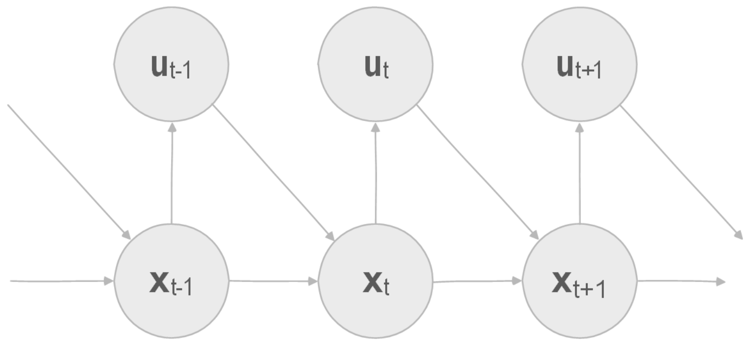


Figure 2.5: A graphical representation of dynamical systems with control input

Formally, a state-space model of a stochastic system can be defined as:

$$\boldsymbol{x}_{t+1} = f(\boldsymbol{x}_t, \boldsymbol{u}_t) + \boldsymbol{\omega}_t, \qquad \boldsymbol{\omega}_t \sim \mathcal{N}(0, \boldsymbol{\Sigma}_\omega)$$

where $\boldsymbol{x} \in \mathbb{R}^D$ is the continuous state, $\boldsymbol{u} \in \mathbb{R}^E$ is the control input and $\boldsymbol{\omega}_k$ is the disturbance modeled by an i.i.d Gaussian distribution. The system's transition function f is an unknown nonlinear function which maps a tuple of system state and controller input $(\boldsymbol{x}_t, \boldsymbol{u}_t)$ to a successor state $(\boldsymbol{x}_{t+1})$. Gaussian processes have been used successfully to identify f from sensory data when a parametric model is very hard to obtain [98]. Also, Gaussian processes have been used to identify f when the state $\boldsymbol{x}$ is not directly observed [31]. Another advantages of nonparametric modeling based on Gaussian processes is the flexibility of including other dependent variables to the state variables such as environmental states [51].

State-space models have been used widely to model continuous dynamics. We propose to utilize these models to represent the continuous dynamics of SHS. This allows us to integrate the state-space modeling techniques for stochastic continuous systems into the proposed data-driven SHS modeling framework.

2.2.1.3 Model Learning of Large-Scale Systems

Despite the simplicity and the flexibility advantages of GP models, a typical GP model suffers from scaling limitations. As the number of data and model dimension increase, the required computation increases exponentially. In GPs, the major computation operation is the inverse calculation of the $n \times n$ covariance matrix K. This inversion is computationally expensive with a complexity of $O(n^3)$, (where n is the number of data points). Therefore, many studies have been conducted to solve this problem with different approaches. Mainly, there are three major approaches that have been developed: Sparse Gaussian processes, local weighted models, and product of Gaussian processes. Sparse Gaussian processes are based on approximating the inverse of the covariance matrix K with a low rank matrix approximation of dimension $m \times m$, (where $m << n$) [81] [93]. Local weighted models partition the function space into smaller weighted models [67]. The partitioning process can be achieved by clustering the training data [68]. Then, the mixtures of these weighted local models are used to approximate the overall model output. The product of Gaussian processes, also known as distributed Gaussian process, is an orthogonal approximation where the data is partitioned into subsets, and each subset is partitioned into another group of subsets until the desired level of approximation is reached [26]. Then, each subset is used to calculate a separate GP in a distributed fashion, (i.e., separate covariance matrix and GP mean function). Later, these separate GPs are combined to calculate the overall covariance matrix and the mean function of the final GP model. This approximation also allows the data partitions to share some data points in order to smooth out the approximation drawback caused by partitioning [66].

2.2.1.4 Model Learning of Multi-Modal Systems

Many modern systems possess multi-modal behavior where they behave differently on each mode. Therefore, inference using a typical GP model for such systems would behave poorly because its covariance function is stationary and cannot represent multiple behaviors with the same hyperparameters. Hence, an alternative GP-based model known as mixtures of Gaussian processes (MGP) is developed to solve this problem [97]. MGP is inspired by the well-known mixture of experts (ME) model [43] where the model experts are Gaussian processes (GP) models. Thus, an MGP consists of a latent discrete variable, typically called gating network, and a set of GP functions called the experts. The state of the discrete variable specifies the GP function which is used to calculate the system output at a given input. Therefore, identifying a system behavior using an MGP allows the model to recognize different behaviors by selecting the appropriate GP function through the gating network. Gating networks play a major role since it identifies the discrete mode of operation in a probabilistic manner. A typical choice of gating network is a GP classifier [97]. This typical choice is limited to systems where the discrete mode of operations are known a priory. An alternative MGP model known as infinite mixture of Gaussian processes (iMGP) is developed to mitigate this limitation [83]. The iMGP model assumes that the data can be modeled by an infinite mixture of Gaussian processes and uses an input-dependent Dirichlet process to sample the gating network.

Another approach that has been developed to model systems with multiple modes is changepoint detection models. Changepoint detection (CPD) models are very useful in time-series modeling where there is a need to detect and locate a significant change in data behavior (known as changepoints) in an online fashion. CPD models build a probabilistic distribution for the run length of each mode and model the data within each mode. Since a typical GP is stationary and cannot model such time-series models, a new Bayesian framework based on GP is developed to build a CPD model that can update the posterior distribution of the run length and allow GP to support this mode changes [32] [90].

2.2.1.5 Online Model Learning

Modern complex systems may possess a dynamical behavior in which the parameters of the underline system changes either abruptly and/or slowly over time. The typical offline model learning methodologies use the available data during the system design to learn the system model. Therefore, the offline learned models usually fail to adapt to the dynamical changes that happen in the runtime. In contrast to offline learning, online model learning methodologies provide a platform which continuously learns the system model during the system runtime. GPs have attractive features which make them good candidates to be used in online learning. The main challenge in online learning is how to deal with the old and the new data. Several methodologies have been developed to tackle this challenge by using either windowing or sampling weighting techniques. Windowing techniques are based on maintaining a fixed size of the training data [70]. Windowing techniques usually exclude old data and include the new measurement to the training data set using either first-in-first-out (FIFO) or by removing the data point with the highest error [5]. On the other hand, sampling weighting techniques are based on giving the new measurement a higher weight than the old data [69] [89].

Another problem which requires an online model learning framework is model-based reinforcement learning for autonomous and optimal control. In robotics application for example, the robot objective is to learn the system dynamics and its control policy efficiently while it interacts with the world through try and error. To achieve this objective, GPs have been used to build a model-based data-efficient learning framework for control policy search, known as PILCO [25] [27]. PILCO starts by applying random control signals and records their corresponding data to learn a probabilistic GP model of the system dynamics given the available initial data. The system model is then used to learn an optimal control policy by optimizing the control policy parameters that minimize a cost function under constraints. Finally, a new control signal is generated using the learned control policy and applied to the system, which in turn will reveal a new data element. These steps are executed iteratively after updating the model using the new revealed data until the system achieves its control objective.

2.2.2 Model Learning of Stochastic Hybrid Systems

Stochastic hybrid systems (SHS) are dynamical systems that integrate continuous and discrete dynamics. Moreover, the continuous and/or the discrete dynamics may exhibit stochastic behavior. Several stochastic hybrid modeling paradigms have been proposed in the literature. For instance, some models use deterministic discrete dynamics with stochastic continuous dynamics while other models use probabilistic discrete dynamics. A typical SHS model has been developed to extend a typical deterministic hybrid system (HS) by modeling the continuous dynamics using stochastic differential equation (SDE) instead of ordinary differential equations (ODEs) [41]. General stochastic hybrid systems (GSHS) extend the (SHS) model by introducing stochastic behaviors in the discrete transitions instead of using deterministic transitions [17]. Both SHS and GSHS are continuous-time stochastic hybrid systems (CTSHS) and they lack a control input. A discrete-time stochastic hybrid system (DTSHS) model has been developed to represent the theoretical and computational aspects of SHS from a discrete-time point of view [4]. In addition to the discrete-time representation, DTSHS allows modeling systems influenced by control inputs.

System identification of hybrid systems (HS) has been investigated in the literature substantially to develop system identification methods for various classes of HS. Typically, these methods aim to estimate the system model parameters for a given model complexity [74]. For HS with unknown model complexity, a kernel-based approach is developed to identify a popular class of HS, known as piecewise affine systems, where GPs are used to model the impulse response of each submodel of the HS [76]. Model learning of SHS has an additional level of complexity because of the presence of uncertainty in the model behavior along with the coupled continuous/discrete dynamics. Many methods have been developed to learn the model parameters (i.e., parameter identification) for a given model structure [49]. The following subsection discusses two main model learning approaches: simulation-based model learning approach and likelihood maximization model learning approach.

2.2.2.1 Simulation-Based Model Learning

The simulation-based model learning approach uses simulated trajectories for parameters identification based on randomized optimization techniques. The goal of this approach is to find the best fit of candidate values of the model parameters. The fit of each candidate solution is determined by evaluating the simulated model trajectories generated using the candidate values against the measured data. Randomized optimization techniques such as genetic algorithm (GA) and Markov chain Monte Carlo (MCMC) can be used to achieve this gaol [49]. Genetic algorithms (GA) are an iterative algorithm inspired by natural evolution and are used typically to solve optimization problems. The main steps of SHS model learning algorithm based on GA are:

1. **Initialization:** Randomly, generate an initial population of candidate values (called individuals) of the model parameters in a binary encoding form.

2. **Evaluation:** Compute the fitness for each individual by evaluating the simulation trajectory generated using it.

3. **Selection:** Select a set of individuals based on their selection probability (a probability proportional to their fitness).

4. **Evolution:** Generate offspring population by applying genetic operators (i.e., crossover and mutation) to the selected individuals, then generate the next population by replacing a fixed percentage of the worst fitting with the new offspring.

5. Repeat **Evaluation**, **Selection** and **Evolution** steps till obtaining a satisfactory solution.

2.2.2.2 *Likelihood Maximization Model Learning*

Learning SHS with likelihood maximization approach is based on learning a maximum-likelihood (ML) model of the underlying system parameters using an expectation-maximization (EM) algorithm [13] [91]. EM is an iterative algorithm which aims to find a local maximum of a target function. Formally, EM aims to maximize the likelihood probability of observed data $\mathbf{Y}$ given a vector of parameter θ, (e.g., $g(\theta) = \log p(\mathbf{Y}|\theta)$). EM is typically used when the likelihood function $p(\mathbf{Y}|\theta)$ cannot be easily evaluated. For instance, when the likelihood function depends on a hidden random variable. In this case, the likelihood can be evaluated through a marginalization over the hidden parameters variables. Calculating this marginalization contains a logarithm operation over an integral (or a large summation in discrete systems). This logarithm operation makes the calculation intractable. Therefore, EM overcomes this problem by evaluating a tractable lower bound $h(\theta|\theta^k)$ of the likelihood function $g(\theta)$ given the current parameters guess θ^k. Then, it maximizes the bound function to get the next guess until the algorithm converges. The typical EM algorithm proceeds as follows:

1. **Initialization**: Initialize the first parameter guess θ^k.

2. **Expectation step**: Calculate the bound function $h(\theta|\theta^k)$.

3. **Maximization step**: Calculate θ^{k+1} that maximize the bound function $h(\theta|\theta^k)$.

4. **Convergence check**: Evaluate $g(\theta^{k+1})$, Repeat the Expectation and Maximization steps until $g(\theta)$ converge.

Applying the typical EM algorithm in hybrid systems is not a straightforward task because the latent variable consists of hybrid states (i.e., discrete and continuous) along with their dependence (i.e., transition guards). Therefore, marginalization over this hybrid latent variables is another major challenge in SHS model learning. Since it requires the estimation of the hybrid state for the given observation. The hybrid state estimation problem requires an enumeration of every possible sequence of the discrete latent variable associated with the continuous variable. This enumeration process is exponential in time, and therefore, the marginalization process would be intractable. Thus, a proper assumption or approximation is needed. An approximation based on a forward-backward Kalman filter recursion on a restricted set of discrete state sequence instead of all possible sequence was proposed in [13]. The restricted set is computed using a heuristic algorithm for N-best enumeration, then the probability of each sequence in the set is normalized by a factor so that their summation is equal to one. This normalization process ensures a valid probability distribution of the restricted set, such that any other sequence, not in the set, has a zero probability.

2.3 Reachability Analysis

Reachability analysis is a typical problem in hybrid systems where for given initial states of a system, it is necessary to predict the reachable states for some finite time horizon T (see Figure 2.6). In many systems, the motivation is to verify the system safety and stability. Often, reachability analysis is used to predict the system behavior given a control policy in order to optimize the control signal over a finite horizon.

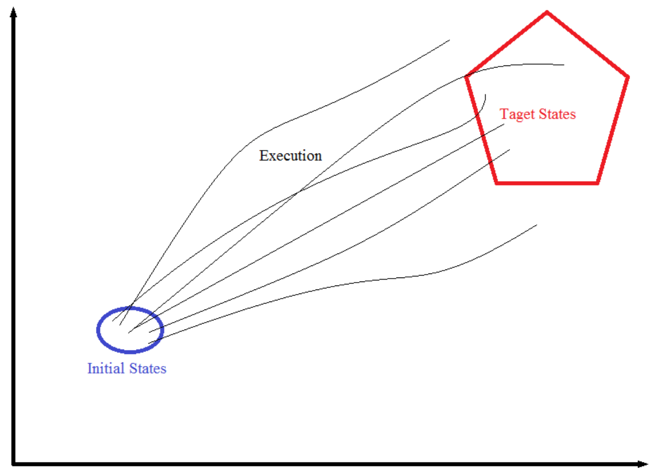


Figure 2.6: Reachability Analysis Problem

2.3.1 Prediction of Nonlinear Stochastic Systems

The major challenge with the prediction of nonlinear stochastic systems for a finite horizon is the need of uncertainty propagation. The simple one-step prediction is a typical regression problem where the system state (or model output) is tested at the current certain input. In contrast, multi-step prediction is more challenging and requires the uncertainty of the predicted state at each time step to be propagated. This problem is also known as multi-step prediction or prediction at uncertain input represented by a probability distribution. For a given Gaussian process model and an uncertain input $\mathbf{X}_*$ represented by a Gaussian distribution, the GP posterior (i.e., predicted state) can be calculated by marginalizing the model output over the input distribution, such that:

$$p(\mathbf{y}_*) = \int\int p((\mathbf{y}_*|\mathbf{X}_*)p(\mathbf{X}_*)d\mathbf{y}_*d\mathbf{X}_*. \tag{2.10}$$

The distribution shown in (2.10) is non-Gaussian and it is analytically intractable [37], and therefore, analytic or numerical approximations are required to overcome this challenge.

To that end, there are two main approaches that have been developed to predict the system behavior in the present of uncertainty in the input. The first approach is based on Monte-Carlo simulation where the predicted system trajectories are described through a set of particle simulations (i.e., trajectory samples) [51]. Approaches based on Monte-Carlo simulation are limited to particle-based control methodologies and optimization techniques. Moreover, these approaches are computationally demanding, especially when a large amount of particles are needed to obtain good accuracy.

2.3.1.1 Analytical Approximation

Analytical approximation approaches approximate the predicted non-Gaussian distribution in (2.10) by a Gaussian distribution. There are two main Gaussian approximations that have been introduced in the literature known as moment-matching and linearization of the predictive distribution [19][36] [27]. Moment-matching approximation is based on computing the first two moments of the predictive distribution by applying the law of iterated expectations. On the other hand, linearization approximation linearizes the GP predictive mean function. The linearization approximation is computationally advantageous over moment-matching but has lower accuracy. These approximations are not applicable in all cases because they depend on the GP kernel function. For example, moment-matching cannot approximate a periodic GP kernel. In [34], moment-matching approximation has been extended to allow long-term forecasting of periodic processes by re-parametrize the periodic kernel.

Another drawback of these approximation is the poor Gaussian approximation of the predictive distribution. Therefore, another algorithm, known as GP-aGMM, is developed to mitigate the moment-matching limitations [40]. This algorithm is based on utilizing adaptive Gaussian mixture model (aGMM) to perform multi-step prediction. GP-aGMM starts by approximating the output distribution by a single Gaussian distribution, then it evaluates the quality of the Gaussian approximation of the output distribution using a kurtosis metric. If the algorithm detects a poor Gaussian approximation, it splits the input distribution into three weighted Gaussian components (i.e., GMM), and then, it approximates the output at each Gaussian component with a Gaussian approximation. The evaluation and splitting steps are then applied recursively for each of the three components until the desired approximation quality is achieved (i.e., the kurtosis metric does not exceed a certain threshold). This algorithm uses multiple Gaussian components to approximate the predictive distribution of the model output, hence, the predictive distribution is approximated by GMM.

2.3.2 Reachability Analysis of Stochastic Hybrid Systems

For hybrid systems, different approximation methods have been proposed to estimate the reachable states such as polygonal flow-pipe approximation [21] and ellipsoidal approximation [52]. Optimization techniques such as face lifting [23] have been also used. On the other hand, reachability analysis of stochastic hybrid systems is more complex since the prediction of the reachable states along with the probability of reaching these states are required. This problem has been investigated extensively in the literature through different estimation methods. In this section, we provide an overview of estimation methods related to our work.

2.3.2.1 Analytical Approximation

Methods based on analytical estimation have been used to solve the reachability problem [15]. For instance, the probabilistic reachability problem has been addressed in [16] using a quadratic form. Methods based on numerical estimations are also used to solve the reachability problem such as Markov chain approximations [53, 77] [3] and numerical solutions of partial differential equation (PDF) [62]. A Markov chain approximation is used to address the reachability problem by computing the probability of reaching some assigned set, and then propagate this probability through the approximated Markov chain transition kernel. Moreover, a numerical solution of Hamilton-Jacobi-Isaacs (HJI) PDE's can be used to tackle the reachability problem in a general game theoretical framework [63]. Typical methods based on HJI equation result in a conservative bound of safely reachable states. Therefore, combining a machine learning technique with HIJ equation is a useful approach to reduce this conservativeness [6]. Another method approximates the reachability problem for a discrete-time SHS with a control input as a stochastic optimal control problem and solves it using dynamic programing [4] [7].

2.3.2.2 Simulation-Based Approximation

Despite the accuracy of the above methods, they may be limited to a certain class of SHS (e.g., SHS with linear continuous dynamics) and more importantly their computational complexity typically explodes with the dimension of the state space. Therefore, probabilistic estimation methods based on randomized algorithms such as Monte-Carlo methods [50][100] [78], multilevel splitting (MLS) variance reduction [85], and simulation-based methods [86] are considered for reachability analysis. Methods based on Monte-Carlo simulation are promising because of their simplicity. Moreover, these methods can be applied to multiple classes of SHS model and are not limited to a certain model representation. These methods rely on simulating a large number of model trajectories to approximate the reachable states by analyzing the simulated samples simultaneously. The number of samples affects the accuracy of the model prediction which introduces a trade-off between computation efficiency and approximation accuracy. This trade-off gives Monte-Carlo methods more flexibility. For instance, a Monte-Carlo based method known as importance sampling is used to increase the prediction accuracy in critical regions while using fewer samples in uncritical regions.

Finally, statistical methods are an area of active research which aim to leverage available data to approximate the reachable state [15]. In this context, a data-driven Bayesian framework is developed to learn and to verify complex physical systems via reachability analysis [38].

2.4 Stochastic Control of Stochastic Dynamical Systems

In model-based system design, decision and control algorithms utilize the system models to drive the system behavior in the desired way. The choice of these algorithms depends on the nature of the modeled system and the desired response. Model predictive control (MPC) is one of the important model-based control algorithms. MPC is very popular because of the ability to optimize the desired system variables subject to constraints on system inputs and states. Additionally, MPC has a flexible formulation in the time domain. The fundamental idea behind MPC is based on manipulating the control inputs of the underline system to obtain an optimal future response of the physical system for a finite receding horizon. This optimization process results in an optimal sequence of the control signals for the desired finite receding horizon. Only the first control signal in this sequence is applied to the physical system then the whole optimization process is repeated again iteratively after receiving new measurements. MPC for linear systems is well-established [80]. In contrast, nonlinear MPC (NMPC) is still an active research topic, especially when the model of the system is hard to obtain. Several methods of NMPC based on data-driven black-box models have been proposed such as NMPC for neural network [73] and NMPC for fuzzy logic [45]. The main drawback of these approaches is the model bias problem. In the model bias problem, MPC inherently considers that the learned model resembles the system behavior accurately regardless of the quality of the learned models and/or the uncertainty in the system dynamics. As an alternative, probabilistic models such as GPs have the advantage of expressing the model confidence using predictive variance. This probabilistic representation allows MPC to consider the quality of the learned models and to take the system uncertainty into account. This section provides a review of MPC algorithms for nonlinear systems modeled by GPs, and MPC algorithms for stochastic systems known as stochastic model predictive control (SMPC).

2.4.1 Gaussian Process Model Predictive Control

Model predictive control using GP models follows the same general principles as typical MPC. However, the process model used to represent the physical system behavior is identified using GPs. Thus, a typical Gaussian process model predictive control problem can be defined as follows.

$$\min_{\mathbf{U}} \sum_{k=0}^{T-1} h(\hat{x}(k+1), u(k)) \tag{2.11}$$

subject to:

$$\begin{aligned} var\,\hat{x}(k) &\leq \delta_v,\; k = \{1, \cdots, T\} \\ u(k) &\in \mathscr{U},\; k = \{1, \cdots, T-1\} \\ \bar{x}(k) &\in \mathscr{X},\; k = \{1, \cdots, T\} \end{aligned} \tag{2.12}$$

where $h(.)$ is the cost function defined over the system state x and the control inputs u, $\bar{x}$ is the estimated system state, $\mathbf{U}$ is the sequence of control signals $[u(1), \cdots, u(T-1)]$, δ_v represents the variance constraints limit, $\mathscr{U}$ represents the input constraints set, $\mathscr{X}$ represents the state constraints set, and T is the finite receding horizon to optimize over.

The goal of Gaussian process MPC is to minimize a given cost function for a finite receding horizon T. The optimization process is based on predicting the system state trajectory for each time step k in the horizon $[1\ T]$. These prediction steps depend on the sequence of control signals $\mathbf{U}$. Therefore, the optimization process results in a control sequence $\mathbf{U}$ that minimizes the cost function of the predicted trajectory and does not violate the constraints on input and state shown in Equation (2.12). Then, only the first element in the control sequence $\mathbf{U}$, i.e. $u(1)$, is applied to the physical system and the entire algorithm is repeated again when a new measurement is received. The optimization can be done using several algorithms. The choice of these algorithms depends on the form of the cost function. For nonlinear systems, the optimization process can be achieved by using dynamic programming, or nonlinear programming.

Degradation in the performance and the stability is a major challenge in MPC because of mismatches between the predicted trajectories using the system model and the actual system response. The reason of these mismatches is due to the unmodeled behaviors and/or poor model quality due to incomplete training data. Luckily, GP MPC can overcome many of these challenges because of its ability to capture nonlinear behaviors and to express the model confidence by the predictive variance. As shown in Equation (2.12), incorporating the model confidence represented by variance constraints δ_v results in an MPC controller with a high level of robustness and stability [48] [47]. The feasibility and realization of using GP MPC has been introduced recently in the literature for many application. For example, GP MPC for nonlinear systems has been used to control unmanned quadrotors in [20]. In industrial practice, GP MPC is used to control a gasliquid separation plant in [55]. Moreover, a nonlinear adaptive control based on GP MPC is proposed in [65], where the uncertainty of the GP models is propagated for the MPC receding horizon to gain more accuracy on the prediction variance.

Despite the ability of GP MPC to handle model uncertainty and enable a robust control scheme, its robustness is limited to system uncertainty that can be modeled by a Gaussian distribution. This limitation restricts the efficacy of GP MPC. Therefore, a stochastic system with non-Gaussian uncertainty represents another control problem that needs to be addressed.

2.4.2 Stochastic Model Predictive Control

Formally, stochastic model predictive control (SMPC) is a stochastic optimization problem defined as:

$$\min_{\mathbf{U}} \sum_{k=1}^{T-1} \mathbb{E}[h(\hat{x}(k+1), u(k))] \tag{2.13}$$

subject to:

$$\begin{aligned} p(\hat{x}(k) \notin F) &\leq \delta_x,\ k = \{1, \cdots, T\} \\ u(k) &\in \mathscr{U},\ k = \{1, \cdots, T-1\} \\ \mathbb{E}[x(k)] &\in \mathscr{X},\ k = \{1, \cdots, T\} \\ x(k+1) &\sim f(x(k), u(k), \theta(k)) \end{aligned} \tag{2.14}$$

where $h(.)$ is the cost function defined over the system state x and the control inputs u, $\mathbf{U}$ is the sequence of control signals $[u(1), u(2), \cdots, u(T-1)]$, F is the unsafe region for the state trajectory, $\mathscr{U}$ represents the input constraints set, $\mathscr{X}$ represents the state constraints set for the expected state trajectory, $\theta(k) \in \mathscr{R}^{n_\theta}$ is a random variable representing the model parameter and T is the finite receding horizon to optimize over. The goal of this stochastic optimization problem is to determine an optimum finite sequence of the control signals $\mathbf{U}$ that minimizes the system cost function $h(.)$ without violating the constraints on the system state and inputs. Therefore, the controller ensures that the system trajectory will not reach an unsafe region F with probability at most δ_x and will be within a certain set $\mathscr{X}$.

The above stochastic optimization problem is intractable, especially when there is no assumption about the form of the probability distribution of the system uncertainty. Hence, an approximation methodology is required in order to transform the probabilistic constraints into deterministic constraints. For instance, deterministic second-order cone constraints have been used to equivalently approximate stochastic constraints in SMPC to control buildings efficiently [71]. In the next subsection, we discuss a feasible approximation known as scenario-based MPC.

2.4.2.1 Scenario-Based Model Predictive Control

Scenario-Based MPC is based on enumerating or sampling the uncertainty in the SMPC problem in order to transform the stochastic optimization problem into a deterministic optimization problem. The enumerated scenarios construct a deterministic MPC problem and collectively approximate the original SMPC. Typically, scenario-based SMPC can be used for systems with any form of uncertainty, however, the random variables representing the uncertainty should be independent of the control signals. The scenario-based optimization approach for SMPC can be represented graphically as an optimization tree (see Figure 2.7), where its nodes represent a possible system state with a calculated weight. Each path in the tree represents a possible system trajectory. The root node corresponds to the current system state (i.e. $x(1)$). The node weight π_i is calculated as the probability of reaching node i from the root node given the probability distribution of the uncertainty in the system. Generating the optimization tree is based on sampling the random variable in the system (e.g., θ) such that the probability of the sampled trajectories is higher than a desired threshold.

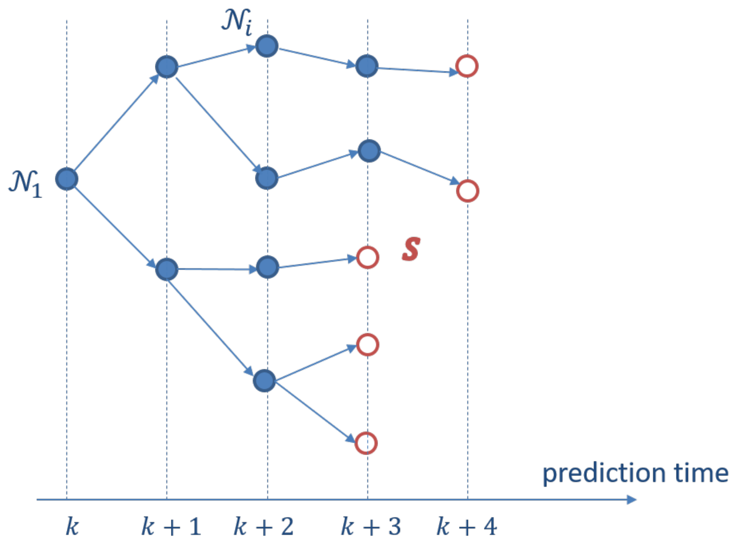


Figure 2.7: A graphical representation of a scenario-based optimization tree, where $\mathcal{N}_i$ is the tree nodes, and S is the set of leaf nodes.

To that end, the SMPC problem defined in Equation (2.13) and (2.14) can be approximated with an optimization tree $\mathcal{T} = \{\mathcal{N}_1, \mathcal{N}_2, \cdots \mathcal{N}_n\}$ with n nodes. This approximation is based on a deterministic optimization problem and can be defined as the following:

$$\min_{\mathbf{U}} \sum_{i \in \mathcal{T} \setminus \{\mathcal{N}_1\}} \pi_i h(x_i, u_{pre(i)}) \tag{2.15}$$

subject to:

$$\begin{aligned} & u_i \in \mathscr{U}, \forall i \in \mathscr{T} \setminus \{\mathscr{N}_1\} \\ & x_i \in \mathscr{X}, \forall i \in \mathscr{T} \setminus S \\ & x_i = f(x_{pre(i)}, u_{pre(i)}, \theta_i) \end{aligned} \tag{2.16}$$

Solving the optimization problem in Equation (2.15) and (2.16) depends on the type of the cost function h and the system model f. Scenario-based MPC approach for stochastic constrained linear systems has been introduced in [11], such that, the scenario-based approximation converts the stochastic optimization problem into quadratically constraint quadratic problem (QCQP). The QCQP can be solved using linear programming. Another form of sampling the uncertainty presented in the estimated system state, modeling error, disturbance, and stochastic mode change is based on a particle-based approach [14]. In this approach, the sampled particles approximate the probability distribution of every random variable in the system. Mixed linear programming algorithm can be used to solve the approximated deterministic optimization problem. The scenario-based SMPC approach has been applied to many systems. For instance, the performance of scenario-based SMPC is evaluated for energy management of hybrid electric vehicles given a stochastic model of the driver behaviors modeled by Markov chain model [28]. Also, scenario-based SMPC has been used to control advanced HVAC systems in energy-efficient fashion while considering the uncertainty in weather and occupancy patterns [75] [102].

2.5 Modeling and Control of Smart Buildings

In this section, we discuss the related work of smart buildings applications we use to evaluate the proposed research approaches. Further, we discuss the challenges presented in these applications and illustrate the contribution of our research to mitigate these challenges.

Heating, ventilation and air conditioning (HVAC) in buildings is a major source of energy consumption. Annual reports show that it is the highest cause of energy consumption in residential buildings. Moreover, HVAC along with miscellaneous electric loads accounts for the highest two energy consumption sources in commercial buildings [2]. Therefore, there is a high demand for developing advanced HVAC control methodologies that can reduce the energy consumption of HVAC unit without compromising users comfort. Many of these advanced control methodologies are model-based and require accurate thermal models. These models should represent the thermal dynamics of buildings accurately and consider the effect of the thermal load in buildings such as occupancy.

Developing accurate thermal models is a challenging task because of the complex behavior of buildings. Thermal dynamics of buildings are a stochastic nonlinear process which are affected externally by the environment (e.g., ambient temperature) and internally by the adjacent zones/rooms. Moreover, buildings dynamics typically differ from one building to another since each building has different construction materials, size, layout, and location. As a result, extensive research efforts have been made in this area to mitigate these challenges.

There are two cases for modeling buildings: Single-zone and multi-zone cases. In the single-zone case, buildings are modeled and approximated as a single thermal zone with an average temperature to reduce model complexity [57] [39]. These models are helpful to provide approximated behaviors of buildings. In the multi-zone case, building models represent the thermal dynamics of each zone in the building to increase the model accuracy [42] [95] [96]. These models are amenable for a better control design to minimized power consumption and therefore the system cost. For instance, an SMPC control method that utilizes a multi-zone model is proposed in [71] in order to achieve energy efficient building climate control.

Most of the above-mentioned work relies on parametric models to represent the thermal behaviors of buildings. However, parametric models require the building detailed structure and/or equations to be known a priori. Also, they might fall to represent the thermal behaviors accurately due to the linearity assumptions in many parametric models. Therefore, nonparametric models can be used to construct a detailed nonlinear model of buildings from sensory data. A nonparametric thermal model based on recurrent neural network (RNN) architecture has been developed to learn a compact thermal modeling of buildings from sensory data [103] . Models based on RNNs architecture are very useful to represent the nonlinearity of thermal dynamics, however, they typically do not consider the uncertainty and time variability.

The thermal behavior of buildings depends on the thermal load such as occupancy. Modeling the effect of the thermal load is another challenge because estimating the heat gained from the thermal load cannot be measured in many cases. A gray box parametric model is used to build a thermal model and combine it with a latent force model based on Gaussian processes [35]. The latent force model is used to model the disturbance caused by the variability in the thermal load. Another parametric model is developed to estimate the heat gained from occupancy, equipment, and solar heating using temperature measurements [8].

In this dissertation, we introduce a data-driven nonparametric modeling framework to learn thermal models of multi-zone buildings. The proposed model can learn the thermal behavior of buildings from sensory data in an online fashion. The nonparametric nature of the proposed model supports creating a unified data-driven modeling framework for buildings. Additionally, we don not assume the buildings thermal load to be measurable, and therefore, we estimate them as a latent discrete variable. This estimation allows the proposed model to capture the effects of the thermal load and to model and predict its pattern (e.g., occupancy pattern).

2.6 Comparison with Proposed Work

In summary, we showed that many recent studies developed data-driven approaches (e.g., Gaussian processes) for stochastic systems to learn nonparametric models. However, these studies are limited to continuous stochastic systems and, they typically do not consider systems with coupled discrete/continuous dynamics (e.g., SHS). Thus, the goal of our research is to address the research challenges and opportunities in developing data-driven approaches for model learning, analysis, and control of SHS.

Model Learning

Most work in SHS assumes a parametric model where the model structure is known a priori. However, it is not always feasible to build a parametric model for many complex systems. Therefore, in our research, we present a non-parametric modeling framework which utilizes sensory data to learn the system dynamics. Additionally, the proposed modeling framework uses an online learning algorithm to adapt the model to variable changes that occur during the system operation.

Existing nonparametric modeling approaches for buildings assume that the thermal load can be measured and used as a model input. This assumption may limit the use of these approaches in many systems, therefore we present nonparametric SHS model for multi-zone buildings when the thermal load is latent. The proposed model estimate the level of the thermal load and learn a distinct model for each level in order to improve the model efficiency and accuracy.

Reachability Analysis

Reachability analysis has been investigated extensively using many approximation methods. However, these methods are based on parametric SHS modeled. Alternatively, many approaches based on Monte-Carlo simulation have been developed to mitigate the limitations of other approximation methods. However, these approaches are computationally expensive. In this dissertation, we present an online data-driven approximation approach which leverages data to approximate a statistical distribution of the reachable states of SHS.

Control/Decision Making of Stochastic Dynamical Systems

Several methods based on stochastic model predictive control have been developed to control systems with stochastic behavior. These methods are limited to systems whose behavior can be represented using a parametric model. Alternatively, recent studies introduced model predictive control methods based on nonparametric models such as Gaussian processes, however, these studies consider systems with continuous dynamics only. In this dissertation, we present a stochastic model predictive control approach based on scenario-based approximation of SHS represented by Gaussian processes and Markov chains.

Chapter 3

Online Model Learning of Stochastic Hybrid Systems

Stochastic hybrid systems (SHS) can model modern cyber-physical systems (CPS) with coupled discrete/continuous stochastic dynamics. Many of these CPS exhibit complex behavior so it may not be feasible to develop accurate parametric models. Data-driven approaches based on machine learning provide alternative methods to identify nonparametric models for analysis and control of CPS. However, model learning of nonparametric SHS faces many research challenges because of the complexity of the coupled discrete/continuous dynamics. In this chapter, we introduce a data-driven SHS modeling paradigm based on Gaussian processes (GPs) and describe a novel clustering-based online learning methodology for the proposed model. Moreover, we demonstrate the feasibility of the proposed approach to create thermal models for multi-zone buildings when the buildings thermal load cannot be measured.

This chapter is organized as follows. Section 3.1 defines the proposed nonparametric SHS based on Gaussian processes. Section 3.2 presents the model learning problem and the challenges associated with it when the discrete dynamics are latent (i.e., the discrete dynamics cannot be measured explicitly). Section 3.3 introduces the proposed data-driven model learning of SHS. Finally, Section 3.4 demonstrates the advantages of the proposed SHS model for smart buildings and evaluates the performance of the model learning approach.

3.1 Model Definition and Execution

We introduce in this chapter a novel nonparametric SHS based on Gaussian processes (GPs). To formalize the SHS model, we define $\mathscr{Q}$ as the set of discrete states and denote the continuous state space by $\mathbb{R}^D$ for each discrete state $q \in \mathscr{Q}$ with dimension D. The hybrid state space is defined as $\mathscr{S} := \mathscr{Q} \times \mathbb{R}^D$. For each discrete mode $q \in \mathscr{Q}$, its corresponding continuous dynamics evolves according to a stochastic process modeled by a GP model. The discrete state may change based on a stochastic process too. Furthermore, we consider systems with two types of inputs: (1) control input and (2) external uncontrolled input (disturbance) from the environment. The control input usually affects the system dynamics based on a control policy ($\pi(\mathscr{S}) : \mathscr{S} \rightarrow \mathscr{U}$) which maps the hybrid state space ($\mathscr{S}$) into the control input space ($\mathscr{U}$). On the other hand, the external uncontrolled input ($v \in \mathscr{V}$) affects the system dynamics and represents the interaction with the environment. Therefore, we propose to model the external input as a time-series disturbance model ($E : \mathbb{N} \rightarrow \mathscr{V}$). The model is formalized by the following definition.

Definition 2. *(Nonparametric SHS). A nonparametric SHS model is defined as a tuple* $\mathscr{H} = (\mathscr{Q}, X, Init, \mathscr{U}, \mathscr{V}, \mathrm{A}, \delta)$*:*

- $\mathscr{Q} := \{q_1, q_2, \cdots, q_m\}$*, for some* $m \in \mathbb{N}$*, represents the discrete state space.*
- X *is a set of continuous variables in the Euclidean space* $\mathbb{R}^D$*.*
- *Init:* $\mathscr{B}(\mathscr{S}) \rightarrow [0,1]$ *is an initial probability measure on the Borel space* $\mathscr{B}(\mathscr{S})$ *where* $\mathscr{S} := \mathscr{Q} \times \mathbb{R}^D$*.*
- $\mathscr{U} \subset \mathbb{R}^E$*, for some* $E \in \mathbb{N}$*, represents the control input space.*
- $\mathscr{V} \subset \mathbb{R}^F$*, for some* $F \in \mathbb{N}$*, represents the external uncontrolled input space.*
- $\mathscr{A}$ *assigns to each discrete state* $q \in \mathscr{Q}$ *a function* $(x_{k+1} = f_q(x_k, u_k, v_k))$ *modeled by a GP which represents the evolution of the continuous state given the predecessor continuous state* $x_k \in \mathbb{R}^D$*, a control input* $u_k \in \mathscr{U}$ *and an external uncontrolled input* $v_k \in \mathscr{V}$*.*
- $\delta : \mathscr{S} \times \mathscr{Q} \rightarrow [0,1]$ *is a stochastic process which assigns a probability distribution over the discrete state given* $\mathscr{S}$*.*

A graphical representation of the model is shown in Figure 3.1.

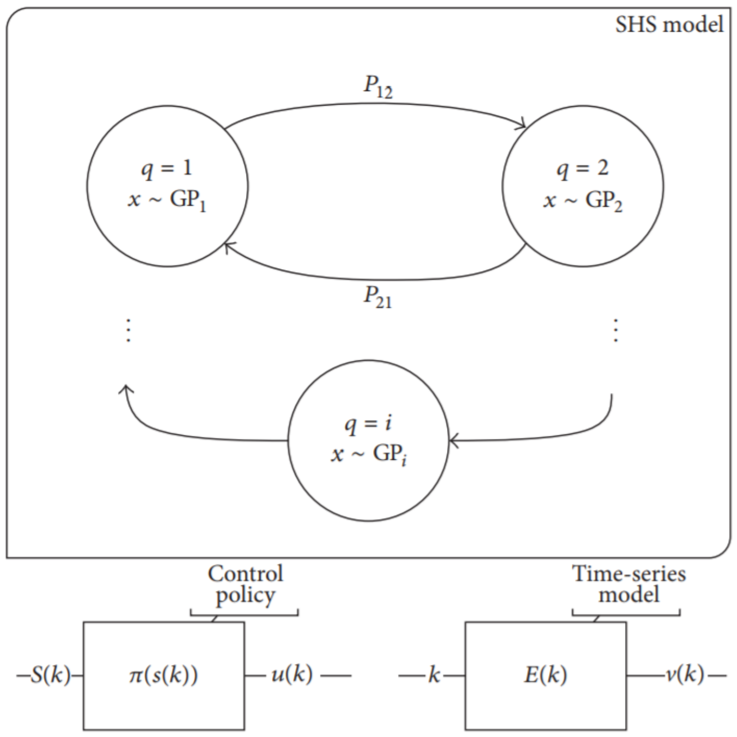


Figure 3.1: Nonparametric SHS components and interconnections

Model Execution

For a finite time horizon $[0,N]$, a system trajectory is denoted by $\{s_k = (q_k, x_k), k \in [0,N]\}$, and it is an execution of $\mathscr{H}$ with a given control policy $\pi(\mathscr{S})$ and a time-series disturbance model $E(k)$. The system trajectory can be obtained using the discrete-time algorithm described below.

For example, Figure 3.2 depicts a sampled trajectory of a given SHS model with a finite time horizon $[0,100]$. The executed SHS model has two discrete modes and the transition between these modes are obtained based on the following probabilistic kernel.

$$q_{k+1} \leftarrow \delta(s_k) = \begin{cases} \sigma(x), & q_{k+1} = 0 \\ 1 - \sigma(x), & q_{k+1} = 1 \end{cases} \tag{3.1}$$

Algorithm 1 Discrete-time SHS execution algorithm

```
State Initialization: $s_0 = (q_0, x_0) \in Init$
k $\leftarrow$ 0
**while** $k < N$ **do**
    $\triangleright$ Calculate the control input $u_k$:
    $u_k \leftarrow \pi(s_k)$
    $\triangleright$ Forecast the external input $v_k$:
    $v_k \leftarrow E_k$
    $\triangleright$ Update the discrete mode $q_{k+1}$:
    $q_{k+1} \leftarrow \delta(s_k)$
    $\triangleright$ Update the continuous state $x_{k+1}$:
    $x_{k+1} \leftarrow f_{q_{k+1}}(x_k, u_k, v_k) \sim \mathscr{G}\mathscr{P}_{q_{k+1}}$
    k $\leftarrow$ k +1
**end while**
```

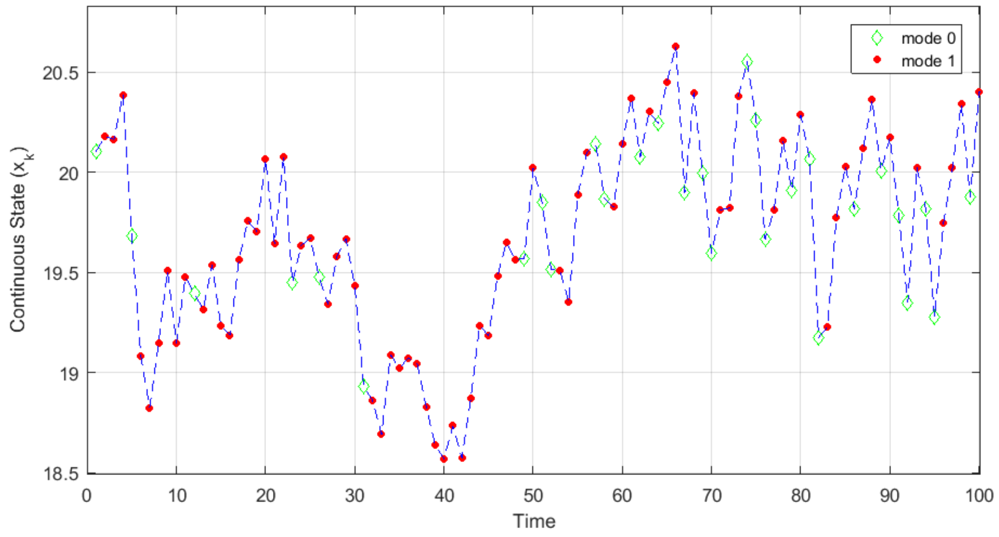


Figure 3.2: Sampled trajectory of a given SHS for the finite time horizon $[0, 100]$

where x is the continuous state, and $\sigma : \mathbb{R} \rightarrow [0,1]$ is a sigmoidal function given by

$$\sigma(x) = \frac{x^d}{\alpha^d + x^d}$$

α and d are the control policy parameters which determine the transition threshold and steepness respectively. In this example, the value of these parameters are as the follows: $\alpha = 20$ and $d = 100$.

3.2 Model Learning of Stochastic Hybrid Systems with Latent Discrete State

The goal of model learning is to identify the discrete state space (i.e., $\mathscr{Q} := \{q_1, q_2, \cdots, q_m\}$ and the number of the discrete states m), the discrete transition function δ, and the continuous dynamics (i.e., $\mathscr{GP}_q(.)$ for all $q \in \mathscr{Q}$) from a given dataset ($\mathscr{D}$). In this chapter, we focus in learning the discrete state space and the continuous dynamics only. However, we extend the learning methodology in the next chapter to include learning the discrete transition function δ as well.

The training data consists of the continuous state, the control input, and the external uncontrolled input. Formally, we can define the training data as:

$$\mathscr{D} := \{(\hat{\mathbf{x}}_k, \mathbf{y}_k) : k = T_s, \cdots, T_e\}$$

where $\mathbf{y}_k$ is the successor continuous state (i.e., $\mathbf{y}_k = \mathbf{x}_{k+1}$), $\hat{\mathbf{x}}_k$ is defined as $(\mathbf{x}_k, u_k, v_k)$, and $[T_s, T_e]$ is the time period at which the data have been collected. For many complex systems such as buildings, achieving the model learning objective is a challenging task, because the sensory data do not necessary include information about the discrete state explicitly. For instance, learning the level of the thermal load in buildings is a major challenge because the thermal load cannot be measured in many real scenarios. Moreover, the system dynamics vary over time due to the variability of the system parameters.

In summary, model learning of SHS encompasses the following challenges:

1. Identifying the discrete state space from data causes a major challenge, because the training data may not have explicit information about the discrete state.

2. The hybrid dynamics add a level of complexity to the learning algorithm because identifying the continuous dynamics requires segmenting the data for each corresponding discrete state, so that, a distinct GP model can be learned using each data segment.

3. As mentioned earlier, the system dynamics depend on an external uncontrolled input v. Therefore, an appropriate time-series model $E(k)$ should be learned and integrated into the SHS modeling framework.

4. Because of the variability of the system parameters, the learning algorithm must adapt the model to these changes, and therefore, the model learning needs to be performed in an online fashion.

3.3 Online Clustering-Based Model Learning of Stochastic Hybrid Systems

In this section, we present a novel online learning approach which can be used to learn a nonparametric SHS model of complex systems with latent discrete dynamics. Moreover, the approach is used for online learning in order to adapt to system changes. The proposed learning approach consists of two phases: Offline phase and online phase as depicted in Figure 3.3. In the offline phase, we initialize the SHS model by identifying its discrete space first. We determine the number of the discrete states heuristically using Silhouette analysis method, and then, we identify the discrete state of each data point by clustering the data. The clustering is also used to label and segment the training data to the corresponding discrete states. This allows us to learn the continuous dynamics for each discrete state using a distinct GP model. In the online phase, we estimate the discrete state in order to determine the GP model used to predict the continuous dynamics. Then, we update the model accordingly. This section provides a detailed discussion of the major steps in the proposed approach.

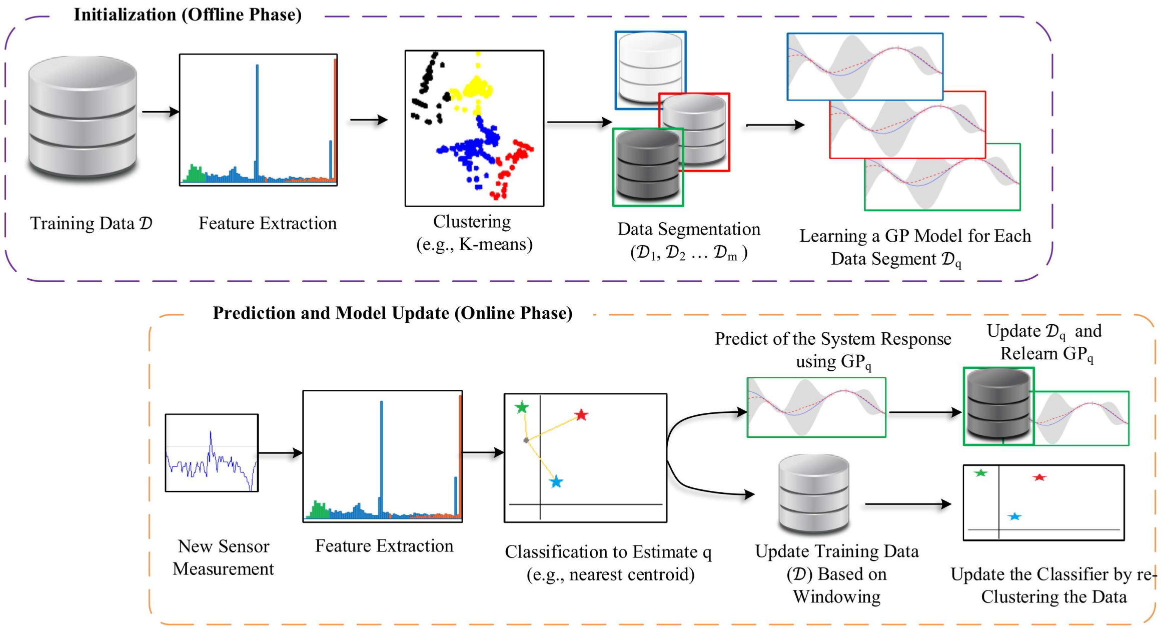

Initialization (Offline Phase)
Training Data $\mathcal{D}$
Feature Extraction
Clustering (e.g., K-means)
Data Segmentation ($\mathcal{D}_1, \mathcal{D}_2 \ldots \mathcal{D}_m$)
Learning a GP Model for Each Data Segment $\mathcal{D}_q$
Prediction and Model Update (Online Phase)
New Sensor Measurement
Feature Extraction
Classification to Estimate q (e.g., nearest centroid)
Predict of the System Response using GP_q
Update $\mathcal{D}_q$ and Relearn GP_q
Update Training Data ($\mathcal{D}$) Based on Windowing
Update the Classifier by re-Clustering the Data

Figure 3.3: Clustering-based online model learning of SHS

3.3.1 Initialization (Offline Phase)

Feature Extraction

Feature extraction is a technique used to transform the training data into a set of features. A set of features (usually referred to as a feature vector) contains the useful data needed for the clustering stage where the irrelevant information is discarded. Feature extraction is needed to distinctively filter the training data. Generally, the feature vector can be computed based on time domain features (e.g., mean, root-mean-square) or frequency domain analysis (e.g., transfer function, Fourier transforms). In this work, the feature vector is computed based on time-domain features, such as mean and rate of change, because of the simplicity and the efficiency of these features.

Data Clustering for Discrete Space Identification

Data clustering is used to estimate the discrete state of each data point. Various clustering algorithms can be used to learn the discrete state such as K-means, Gaussian Mixture Model (GMM), and Hierarchical clustering algorithms. The choice of the appropriate algorithm depends on the nature of the application and the collected data . In this section, we describe the K-means clustering algorithm which calculates the optimal centroids ($\hat{\mathscr{C}}$) for K clusters, so that $\hat{\mathscr{C}}$ minimizes the following potential function:

$$\hat{\mathscr{C}} = \arg\min_{\mathscr{C}} \sum_{g(x)\in\chi} \min_{c\in\mathscr{C}} \parallel g(\hat{\mathbf{x}}) - c \parallel^2 \tag{3.2}$$

where $\chi \in \mathbb{R}^{n\times d}$ is the feature matrix extracted from the training data ($\mathscr{D}$) with size n and feature dimension d, $g(\hat{\mathbf{x}}) \in \mathbb{R}^d$ is the feature vector for the data point $\hat{\mathbf{x}} \in \mathscr{D}$, and $\hat{\mathscr{C}} \in \mathbb{R}^{K\times d}$ represents the optimal K-centroids. In the evaluation section, we consider the study of additional clustering algorithms such as Gaussian Mixture Model (GMM) and Hierarchical clustering.

The clustering-based learning approach considers the data that lie close to each other to probably belong to the same discrete state. However, it requires the number of clusters (i.e., the number of discrete states $m \in \mathbb{N}$) to be known a priori. We identify the number of discrete states m using a heuristic algorithm known as Silhouette analysis method [44]. The Silhouette analysis method determines the best number of clusters ($\hat{K}$) which results in the best clustering consistency. Silhouette analysis evaluates the clustering consistency for a given K by calculating a scoring coefficient for each clustered data point. The Silhouette scoring coefficient has a range of [-1,1] where scores near +1 are assigned to data points that lie far from the neighboring clusters. On the other hand, scores near 0 are assigned to data points that lie very close to the boundary between their cluster and a neighboring one. Negative Silhouette scores are assigned to data points which might have been allocated to the wrong cluster. Therefore, clusters with a higher average Silhouette score have a better consistency than clusters with a lower average Silhouette score. Formally, the Silhouette score $s(i)$ for a given data point i can be obtained using the following formula:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{3.3}$$

where $a(i)$ is the average distance from the data point i and the other points in its cluster, and $b(i)$ is the minimum, minimized over clusters, average distance between the data point i and other points in a different cluster.

We use Silhouette analysis method to identify the number of discrete states, such that the average Silhouette score is maximized:

$$m = \hat{K} = \arg\max_{K} \bar{s}(i),\ K \in G \tag{3.4}$$

where G is a finite set of potential value of m.

Data Segmentation and Learning GP Models for the Continuous Dynamics

As shown in the previous subsection, data clustering enables us to identify the discrete states ($\{q_1, q_2, \cdots, q_m\}$) and to label each data point in $\mathscr{D}$ with the corresponding discrete state. The labeled data is used to segment the training data into m datasets:

$$\begin{aligned} &\forall q \in \mathscr{Q}, \mathscr{D}_q := \\ &\{(\hat{\mathbf{x}}, \mathbf{y})_i : q = \arg\min_{q' \in \mathscr{Q}} \parallel g(\hat{\mathbf{x}}_k) - c_{q'} \parallel^2, \forall (\hat{\mathbf{x}}, \mathbf{y})_i \in \mathscr{D}\} \end{aligned} \tag{3.5}$$

where $c_{q'}$ is the cluster centroid of the discrete mode q' and $g(\hat{\mathbf{x}})$ is the feature vector of the data point $\hat{\mathbf{x}}$. We use each data segment $\mathscr{D}_q$ to learn a distinct GP model in order to learn the continuous dynamics for the discrete state $q \in \mathscr{Q}$. As discussed in Section 2.1, learning a GP model is an optimization process that calculates the optimal hyperparameters $\hat{\Theta}_q$ of the GP in order to maximize the log likelihood function:

$$\hat{\Theta}_q = \underset{\Theta_q}{\arg\max} \log p(\mathbf{y}|\Theta_q, \mathscr{D}_q) \tag{3.6}$$

The offline phase learns the SHS model using the initial training dataset, then the model online is updated online whenever new measurement is received in order to improve the model quality and to adapt to the variation in the underline system.

3.3.2 Prediction and Model Update (Online Phase)

Classification for State Estimation/Prediction

During the system operation at each time step k, we classify the new measured data point ($\hat{\mathbf{x}}_k$) to estimate the current discrete state q_k. The new data point is classified using a nearest centroid classifier. This classifier assigns to the new data point the label of the class with the nearest centroid. Hence, the discrete state can be estimated as:

$$q_k = \underset{l}{\arg\min} \parallel g(\hat{\mathbf{x}}_k) - c_l \parallel^2$$

where c_l is the centroid of class l.

Online Learning of SHS with Windowing

Online learning is very useful in order to adapt the model to the variability of the system parameters; however, it requires a proper data selection method to update the training dataset. We use a moving window method to update the model dataset (i.e., $\mathscr{D}_q$ for all $q \in \mathscr{Q}$) based on first-in-first-out (FIFO) policy where the new data point is inserted and the earliest one is dequeued. The dataset ($\mathscr{D}_q$) for each discrete state q is updated independently, and then it is used to update its corresponding ($\mathscr{GP}_q$) model and to re-optimize its hyperparameters ($\hat{\Theta}_q$). We also use the moving window method to update the training dataset for the clustering (i.e., $\mathscr{D}$), and then we update the classifier centroids accordingly.

Both the classifier centroids and the GP associated with the current discrete mode are updated by repeating the learning process of those models using the updated datasets.

We maintain a fixed window size for all the m datasets of the GP models and use a different window size of the training dataset for the clustering algorithm. The sizes of the training datasets affect the model learning running complexity and determine the forgetting weight of the model. Typically, there is a trade-off between selecting large and small dataset size. Online algorithms with large dataset size suffer from high running complexity and they adapt the updated models to the system variability in a slow pace (i.e., low forgetting weight of old information). However, these algorithms can learn models with high quality. On the other hand, online algorithms with small dataset size have efficient running complexity and the updated models adapt to the system variability faster. However, these algorithms may miss useful information from the discarded old data especially in models with large input space. Therefore, we consider the datasets sizes as configuration parameters because they depend on the nature of the modeled system and the requirements and the constraints of the system application. The updated models adapt to the variation in the systems, so that the model accuracy is improved and reflects the behavior of the underline system.

Predict the System Response

We predict the continuous state of the underline system using the GP model corresponding to the estimated discrete state (i.e., $\mathscr{GP}_{q_k}$). Hence, The predictive distribution of the continuous state can be formalized as:

$$p(\mathbf{x}_{k+1}|\hat{\mathbf{x}}_k) = f_{q_k}(\hat{\mathbf{x}}_k) \sim \mathscr{GP}_{q_k}(m(\hat{\mathbf{x}}), k(\hat{\mathbf{x}}, \hat{\mathbf{x}})) \tag{3.7}$$

where $\hat{\mathbf{x}}_k$ is the tuples $(\mathbf{x}_k, u_k, v_k)$. The prediction and the learning steps in the online phase are repeated iteratively each time when new data measurements arrive.

The quality of the learned model is evaluated by measuring the prediction performance. The prediction performance represents the agreement between the model and the physical process output. This comparison can be measured quantitatively using the root mean square error (RMSE) and the mean relative square error (MRSE) metrics. The RMSE and MRSE are defined as.

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{N} e_k^2},\ MRSE = \sqrt{\frac{\sum_{k=1}^{N} e_k^2}{\sum_{k=1}^{N} y_k^2}} \tag{3.8}$$

where y_k is the process output and $e_k = \hat{y}_k - y_k$ is the prediction error of the k^{th} time-step.

Multi-Step Prediction within a Discrete Mode

Multi-step prediction of the system response is required in many verification and optimization algorithms. For the proposed SHS model, multi-step prediction can be achieved by propagating the predictive distribution in (3.7) using the GP model of the current estimated mode (i.e., $\mathscr{GP}_q$) . However, this propagation faces a major challenge where it requires to predict the system response at an uncertain input defined by a Gaussian distribution (i.e., $p(\mathbf{X}_*) \sim \mathscr{N}(\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$). The GP posterior of this predictive distribution is analytically intractable and can be approximated with Gaussian distribution as discussed in section 2.1.1.3.

3.3.3 Online Learning of Time-Series Disturbance Model

As mentioned earlier, the proposed approach identifies a time-series model $E(k)$ for the uncontrolled input v using single Gaussian processes model. The time-series model is learned online and independent from the SHS model learning algorithm. Therefore, the time-series model $E(k)$ of an observed time-dependent variable v_k is modeled as:

$$v_k = f(k) \sim \mathscr{GP}(m(k), K(k,k'))$$

We also use the moving window method to learn the above time-series model online. The above model is very generic and simple, however other complex time-series models can be used based on the modeled system and the available data, (please refer to section 2.2.1.1 for a detailed discussion).

3.4 Modeling Thermal Dynamics of Buildings with Latent Thermal Load

Thermal models are essential for model-based control methodologies which allow buildings to be operated autonomously in energy and cost efficient manner [79]. However, buildings have complex stochastic nonlinear thermal dynamics which are affected externally by the environment. Moreover, the thermal dynamics of buildings depend on internal thermal loads such as occupancy. In this section, we evaluate the performance of the proposed framework and its efficacy to learn thermal models for buildings when the applied thermal load (e.g., occupancy) cannot be measured. In this case, the zone air temperature represents the model continuous state, the HVAC heating/cooling rate represents the control input, the ambient temperature represents the uncontolled input, and the thermal load represents the latent discrete state.

We evaluate the thermal model learning for: (1) A two-zones data center building and (2) a five-zones office building. The actual behaviors for both buildings are generated using simulations based on the EnergyPlus software [22]. EnergyPlus is an open-source cross-platform building energy simulator engine funded by the U.S. Department of Energys (DOE), and Building Technologies Office (BTO), and managed by the National Renewable Energy Laboratory (NREL). EnergyPlus is used by engineers, architects, and researchers for high fidelity simulation of buildings. EnergyPlus requires two inputs: (1) The ambient temperature and the environment data and (2) The building description. The building description defines its structure and layout, the construction materials, the thermal zones with their dimensions and area, the HVAC system, the control strategies and more. It also defines the building thermal loads with their schedules such as occupancy, lights, and electrical equipment. These detailed descriptions are used to construct several models (e.g., airflow network model, pollution model, on-site power model) by which EnergyPlus simulates the building behavior.

The main purpose of this experiment is to learn the thermal dynamics of buildings when their thermal load cannot be measured. We use EnergyPlus to represent the system response, and use the data collected from EnergyPlus simulation (the building state and the control input values) to learn and to evaluate the proposed model learning framework. We also use the weather data used by EnergyPlus to learn a time-series model in order to forecast the uncontrolled input. At each time step k, we collect the system state (the zone air temperatures) and the control input (the applied heating/cooling rate from the HVAC unit). The collected data are then used to learn/update the model online as described early. Moreover, we compare the model prediction against the system response (i.e., zone air temperatures for the next time step $k+1$) in order to evaluate the model learning performance. A general block diagram of the experiment setup is shown in Figure 3.4.

The proposed framework has been implemented using MATLAB® and statistics and machine Learning Toolbox Release 2016a [59]. Gaussian processes models are learned using quasi-Newton optimization algorithm in order to optimize the model hyperparameters. The following subsections discuss the experiment results for a two-zones data center building and a five-zones office building.

3.4.1 Two-Zones Data Center Building

In this experiment, we use a dataset of a two-zones data center building to learn and to evaluate the building thermal model using the proposed SHS modeling framework. This dataset is a synthetic data generated by EnergypPlus and used to evaluate buildings modeling in [103]. The data center building consists of two zones: The west zone and the east zone as shown in Figure 3.5. The dataset contains hourly data for one-year simulation. Each data point consists of zone air temperature, ambient temperature, thermal load heat-

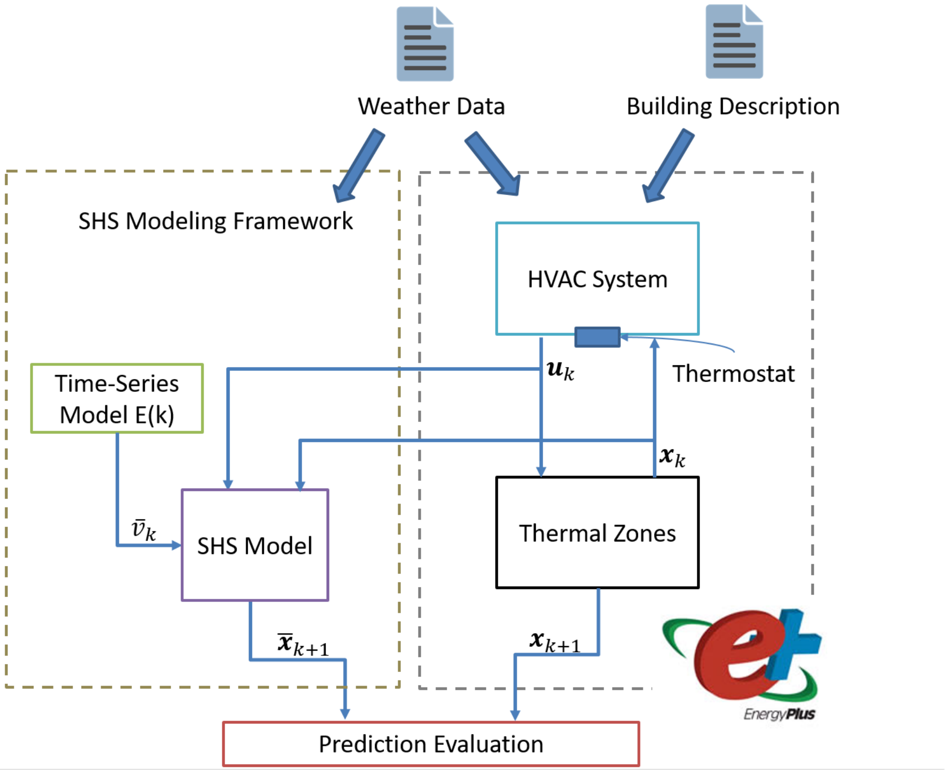


Figure 3.4: Block diagram of the experiment setup

ing rate from IT equipment, lights, and electrical equipment, and HVAC unit cooling rate. The heat rate from the building thermal load depends on the building activities (e.g., how utilized or idle is the IT equipment). Many models in the literature assume that the thermal load can be measured and therefore can be used as a model input (e.g., [103]), however, this is very expensive in real data centers. Therefore, we consider the thermal load as a latent discrete state of the system and we use our proposed approach to estimate it from the measurable thermal data (i.e., zone temperature and HVAC unit cooling rate).

To formalize the model, we define the SHS model as follows: The continuous state ($\mathbf{x} \in \mathbb{R}^2$) represents the air temperature for both zones, the discrete state q represents the thermal load level, the uncontrolled input ($v \in \mathbb{R}$) represents the ambient temperature, and the control input ($u \in \mathbb{R}$) represents the HVAC cooling rate. Therefore, the predictive

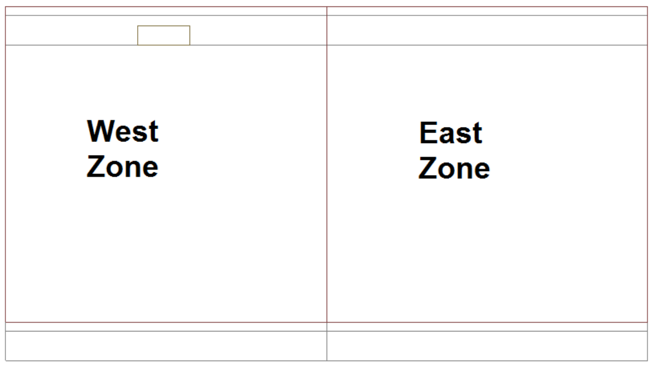


Figure 3.5: Two-zone data center building

distribution of the zone air temperature can be represented as

$$\mathbf{x}_{k+1} \sim f_{q_k}(\mathbf{x}_k, u_k, v_k) : q_k \in \mathcal{Q}$$

In the offline phase of the model learning approach, we used the first four weeks of data (i.e., 672 data points) to initialize the model and to learn its discrete states using the K-means clustering algorithm. Data clustering starts by extracting the time-domain features from the data. In this experiment, the features are the average cooling rate (i.e., $u(k)$) and the zone air temperature difference (i.e., $\Delta x = x_k - x_{k-1}$)(note, the values of these features are normalized in order to unify their scale). Then, the number of discrete states is estimated using the Silhouette analysis method. The calculated average Silhouette scores of this experiment are shown in Table 3.1. Based on this analysis, the number of the discrete states is three (i.e., $m = 3$). Since the discrete state represents the thermal load level, we identified four discrete states of the thermal loads which are corresponding to low, medium and high heat gained from the thermal loads. Figure 3.6 depicts the three clusters of the training data for each zone. Finally, we segment the data into three datasets and use them to learn three distinct GP models for each zone.

As explained earlier, the ambient temperature is considered as an uncontrolled input, thus a time-series GP model is used to forecast its value for the next hour. We also update

Table 3.1: The average Silhouette scores for different number of clusters

m	Average Silhouette Score	
	West Zone	East Zone
2	0.70	0.55
3	0.71	0.65
4	0.62	0.60
5	0.65	0.65

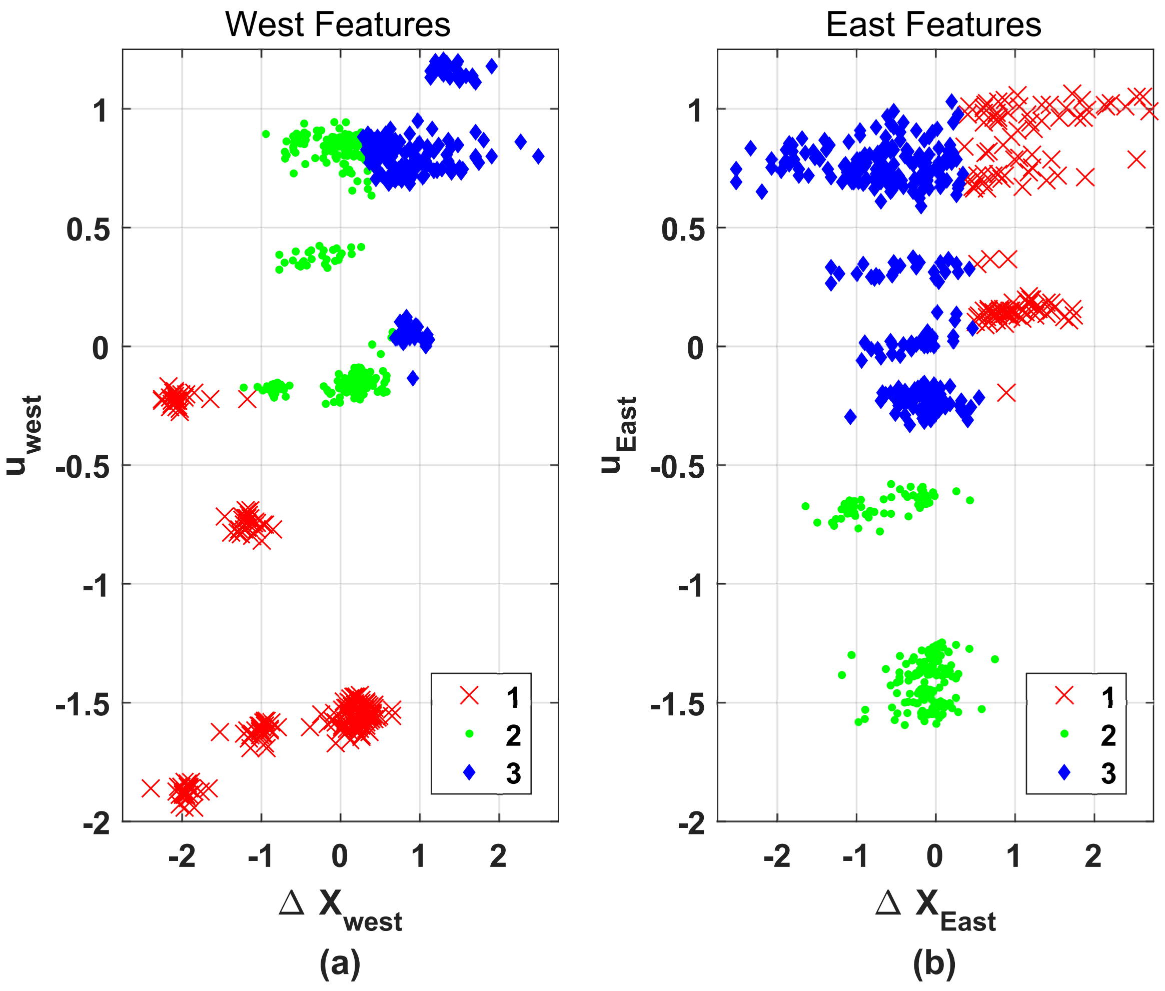


Figure 3.6: Clustering of the training data using *K*-means algorithm

the forecast model every hour when we receive a new measurement. Figure 3.7 shows the forecasting results of the ambient temperature for three days.

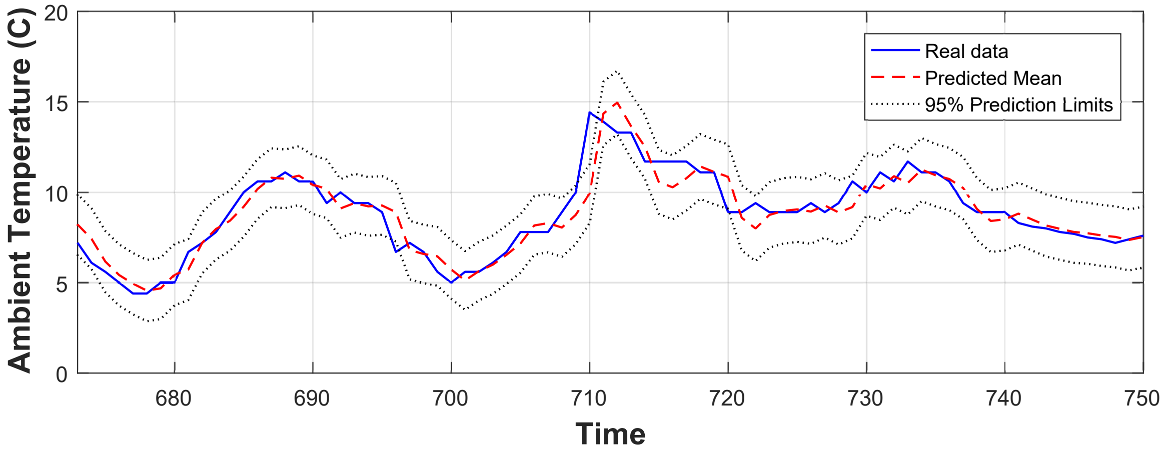


Figure 3.7: Prediction of ambient temperature using a time-series GP model

In the online phase, we predict the zone temperature for the next hour using the learned SHS model and the predicted ambient temperature. Further, we update the training datasets every hour when we receive new data point using the moving window method with a size of one week data (i.e., 168 points). The updated datasets are then used to relearn the corresponding models. We run the prediction/learning steps iteratively for almost 11 months and use the results to evaluate the proposed SHS model learning approach. The results for the first three days of the discrete mode estimation (i.e., thermal load level) and the west zone temperature prediction are shown in Figure 3.8 and Figure 3.9; respectively. The depicted results present the performance of the online model learning approach to predict the continuous state accurately and to estimate the discrete state successfully. As shown in Figure 3.9, we compare the model prediction with a full GP model. The full GP model assumes that the thermal load can be measured and the thermal model is defined as:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, u_k, v_k, l_k)$$

where l_k is the total heat rate from the thermal load. As shown in the results, the proposed SHS model and the full GP model have a similar performance. However, the learning of the full GP is computationally expensive because of the large dimension of the input data. On the other hand, SHS model segment the data into smaller distinct models for each estimated level of the thermal load.

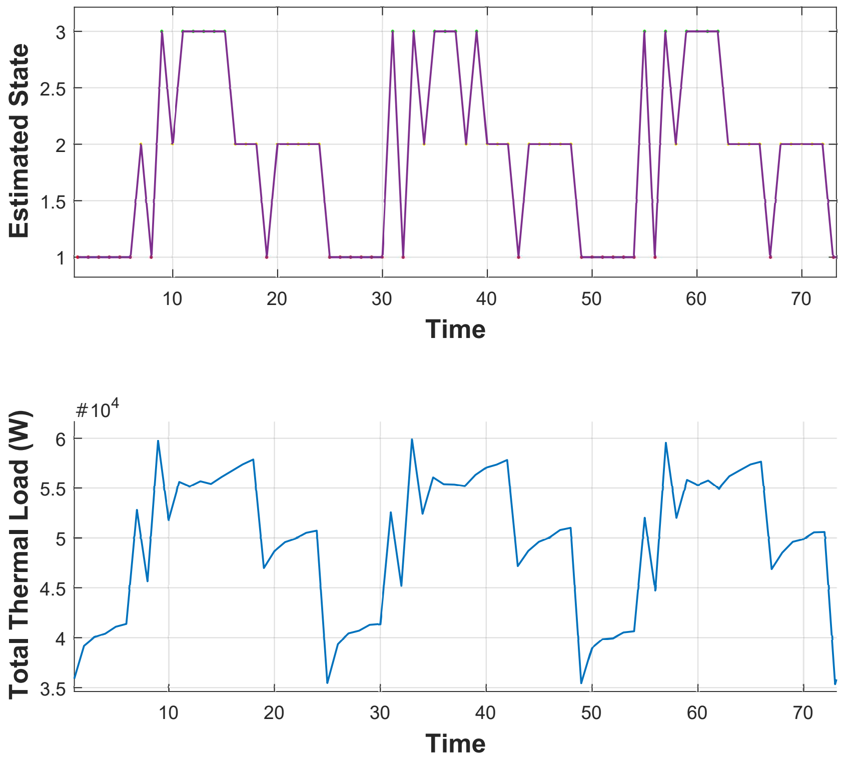
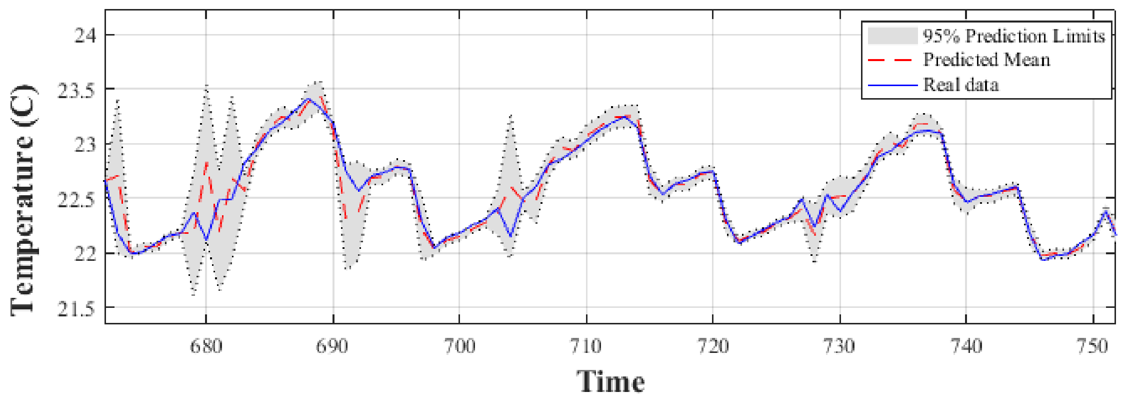


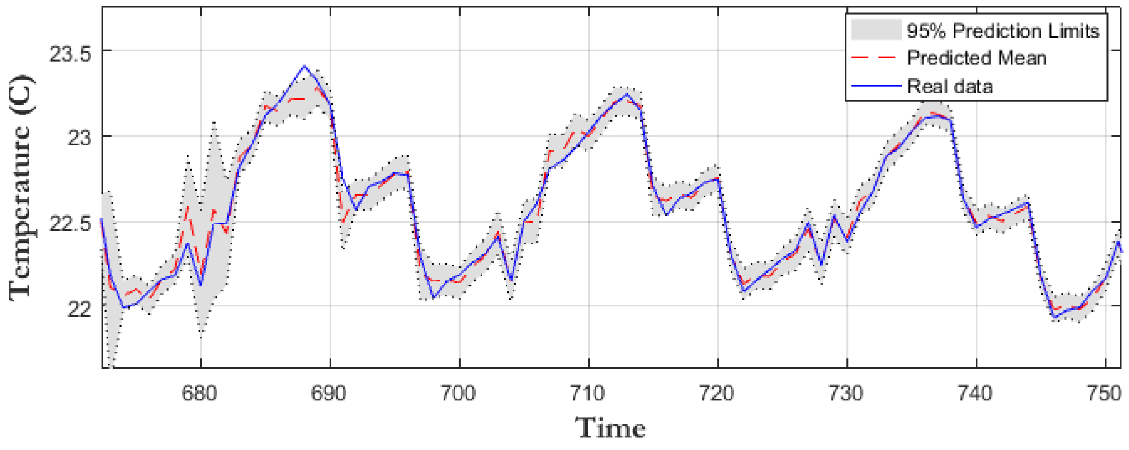
Figure 3.8: Discrete state estimation of the west zone versus the actual total the thermal load

We evaluated the prediction performance of the continuous state using the root mean square error (RMSE) and the mean relative square error (MRSE) error metrics, defined in Equation (3.8). The error metrics statistics for the SHS model and the full GP model are shown in Table 3.2. The results show that our SHS model predicted the zones temperature with a good performance. Also, the performance of the SHS model is similar to the full GP model which assumes that the thermal load is known and can be measured.

To evaluate the improvement of the online learning approach, we compared the RMSE metric for the proposed onlined learned model against another offline model, which is learned one time offline only. In this experiment, we consider the changing of the building climate due to seasons changing as an example to represent the variability in the system.



(a)



(b)

Figure 3.9: The predicted temperature of the west zone using: (a) the proposed SHS model (b) the full GP model

Table 3.2: Performance metrics for the prediction of the zone air temperature, with a comparison between the SHS model and the full GP model

	West Zone		**East Zone**	
	Full GP	**SHS**	**Full GP**	**SHS**
RMSE	0.05	0.06	0.07	0.06
MRSE	0.008	0.009	0.01	0.009
Max	0.6	1.03	0.6	0.8

Figure 3.10 shows the RMSE statistics for both models calculated for each week of the year. The results indicate that online learning allows our model to adapt to the variation in the system with a good performance. On the other hand, the offline model fail to adapt to these variation.

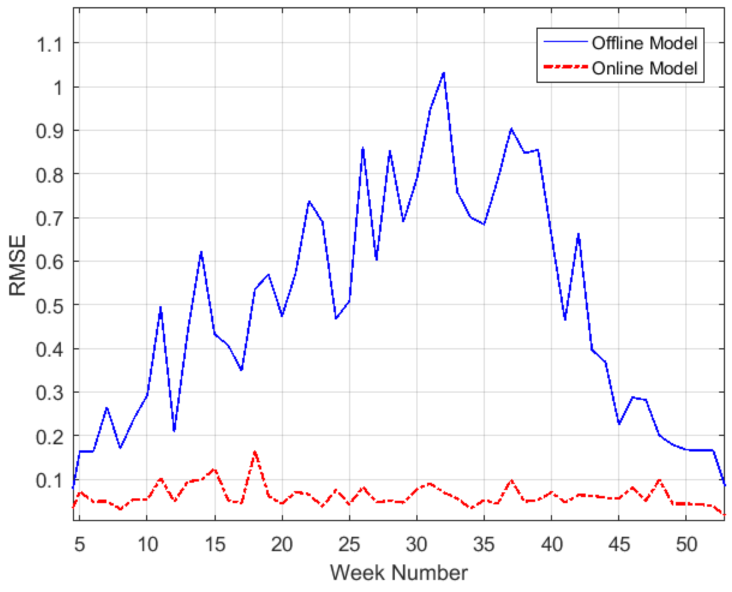


Figure 3.10: Performance metric (RMSE) of the zone air temperature prediction for both the online and the offline learned SHS model

Evaluating Different Clustering Algorithms

There are many clustering algorithms which can be used to estimate the mode of the system. The choice of such algorithm depends on the nature of the data and the application. In this experiment, we studied three different clustering algorithms which are K-means, Gaussian Mixture Model (GMM), and Hierarchical clustering algorithms. Figure 3.11 shows the clustering of the training data using each algorithm. Both K-means and GMM algorithms require the number of the clusters to be given a priori, where we used Silhouette analysis method to estimate it. Hierarchical algorithm, on the other hand, has the advantage of determining the number of the clusters based on its stopping criteria (e.g., cluster inconsistency), but it has many design parameters and suffers from run time complexity $O(n^2 \log(n))$ for large-scale data as indicated in Figure 3.12.

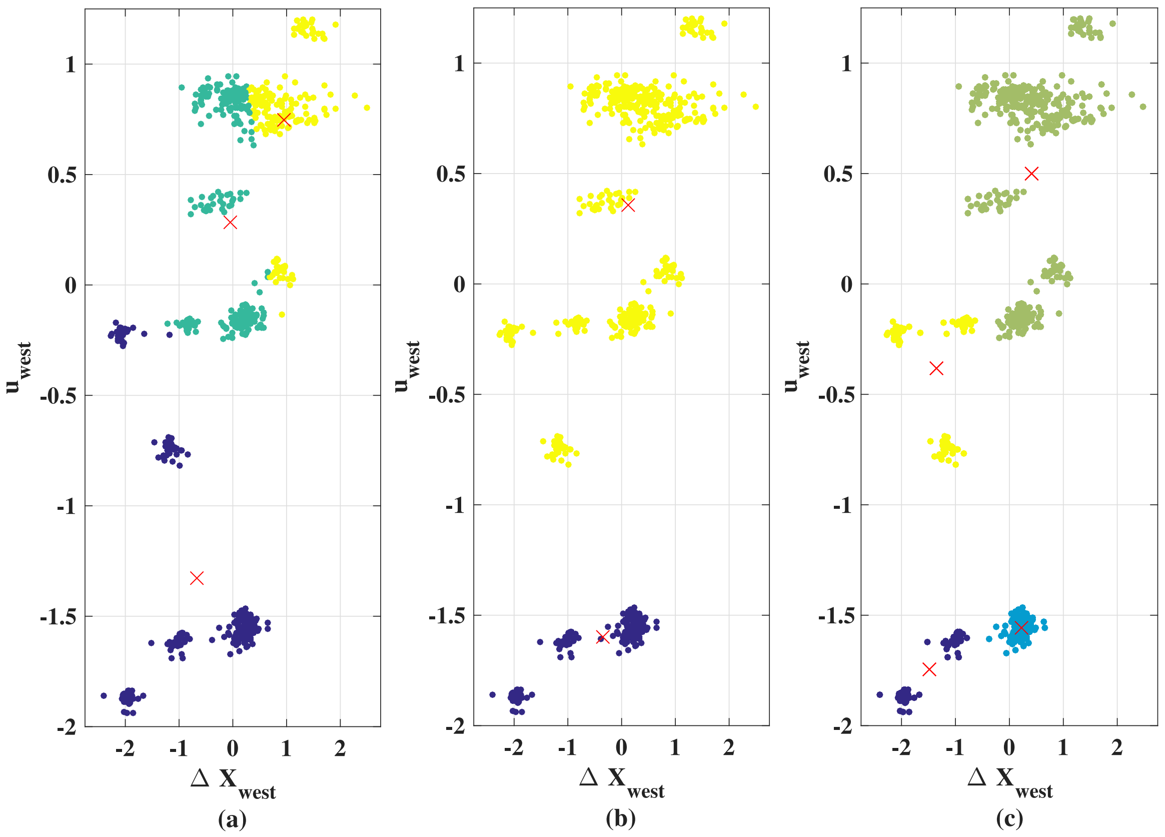


Figure 3.11: Clustering of the training data for the West zone using: (a) *K*-means algorithm, (b) GMM algorithm, (c) Hierarchical algorithm

Figure 3.13 shows the boxplot of the heat rate caused by the thermal load (i.e., light, occupancy and equipment) for each estimated discrete mode (i.e., cluster). Despite the differences between these algorithms, the results show that they identify the discrete state with distinctive levels of the thermal load. GMM estimated the thermal load levels with the lowest accuracy (i.e., Low, High). Hierarchical algorithm has the highest accuracy (i.e., low, medium-low, medium-high, high) of the thermal load levels.

Mutli-step prediction within the same mode

As described earlier, multi-step prediction requires propagating the uncertainty of the predictive distribution for each time step. In addition, the control policy of the HVAC is

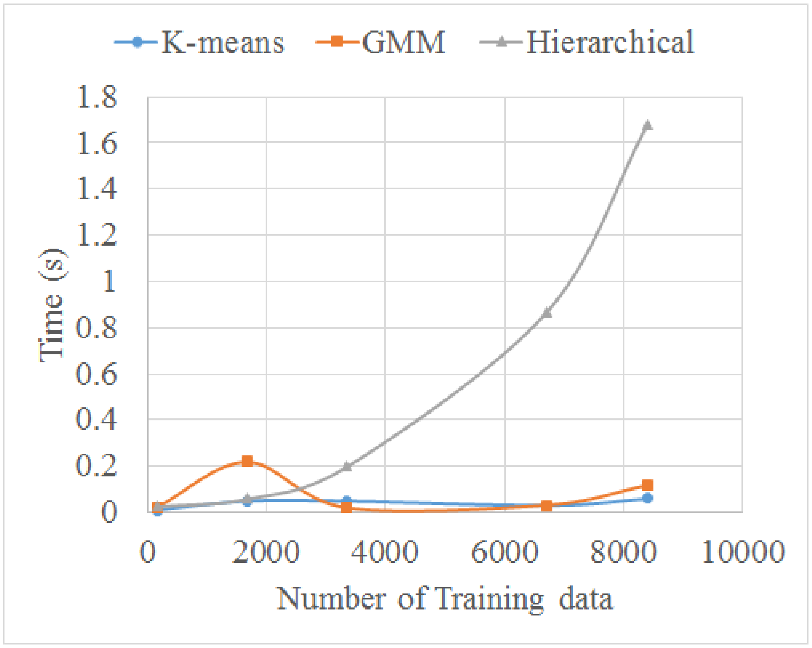


Figure 3.12: Clustering time for different data size of *K*-means algorithm, GMM , and Hierarchical clustering algorithms

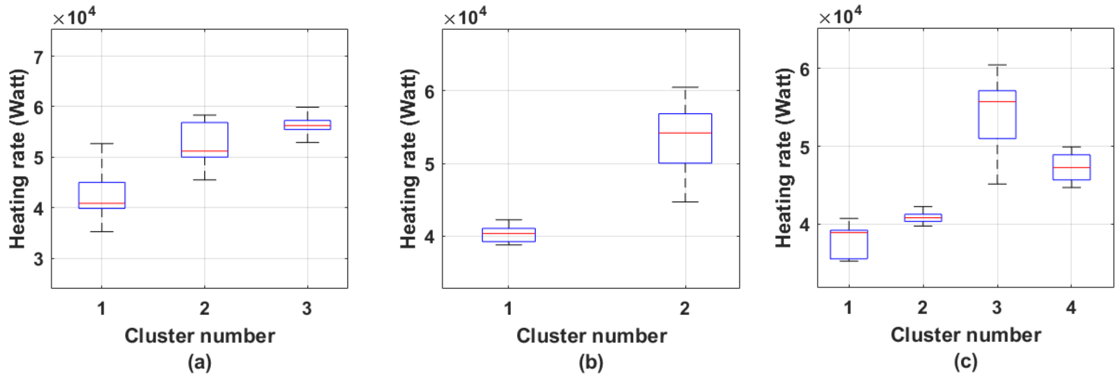


Figure 3.13: Average heating rate caused by west zone's thermal load for each discrete mode (in Watt) estimated by: (a) *K*-means algorithm, (b) GMM algorithm, (c) Hierarchical algorithm

Table 3.3: Performance metrics for multi-step prediction of both SHS model and unimodal GP model

	SHS model		**Unimodal GP**	
	West	**East**	**West**	**East**
RMSE	0.23	0.3	0.56	0.3
MRSE	0.009	0.013	0.023	0.013
LD	51	6	27	257

required to predict the control sequence of the HVAC unit (i.e., cooling rate). To do so, we used GP to learn the control policy of the HVAC as a function of the hybrid system state (i.e., zone air temperature and discrete mode) and the ambient temperature, i.e.,:

$$\mathbf{u}_{k+1} \sim f(\mathbf{s}_k, v_k) : \mathbf{s}_k = (q_k, \mathbf{x}_k)$$

Furthermore, we use the RMSE and MRSE metrics as a performance measure in order to evaluate the predicted mean. We also use another performance metric known as log predictive density (LD) to evaluate the prediction performance of the predicted mean and the predicted variance as well [46]. LD is defined as:

$$LD = \frac{1}{2}\log(2\pi) + \frac{1}{2N}\sum_{i=1}^{N}\log(\sigma_i^2) + \frac{e_i^2}{\sigma_i^2}$$

where σ_i is the prediction variance in i^{th} step and e_i is the error between the system output and the predicted mean. Figure 3.14 shows the multi-step prediction of the zone air temperature for both west and east zones using the proposed SHS model. We also implemented the multi-step using a typical single GP model which does not estimate the discrete mode of the buildings (i.e., thermal load level). Figure 3.15 shows the multi-step prediction of the zone air temperature for both west and east zones using a typical unimodal GP model. Table 3.3 shows the performance metrics statistics for the multi-step prediction within the same mode (i.e., [1 10]) for both the SHS and the unimodal GP.

The performance metric shows that, the SHS model provide more accurate prediction

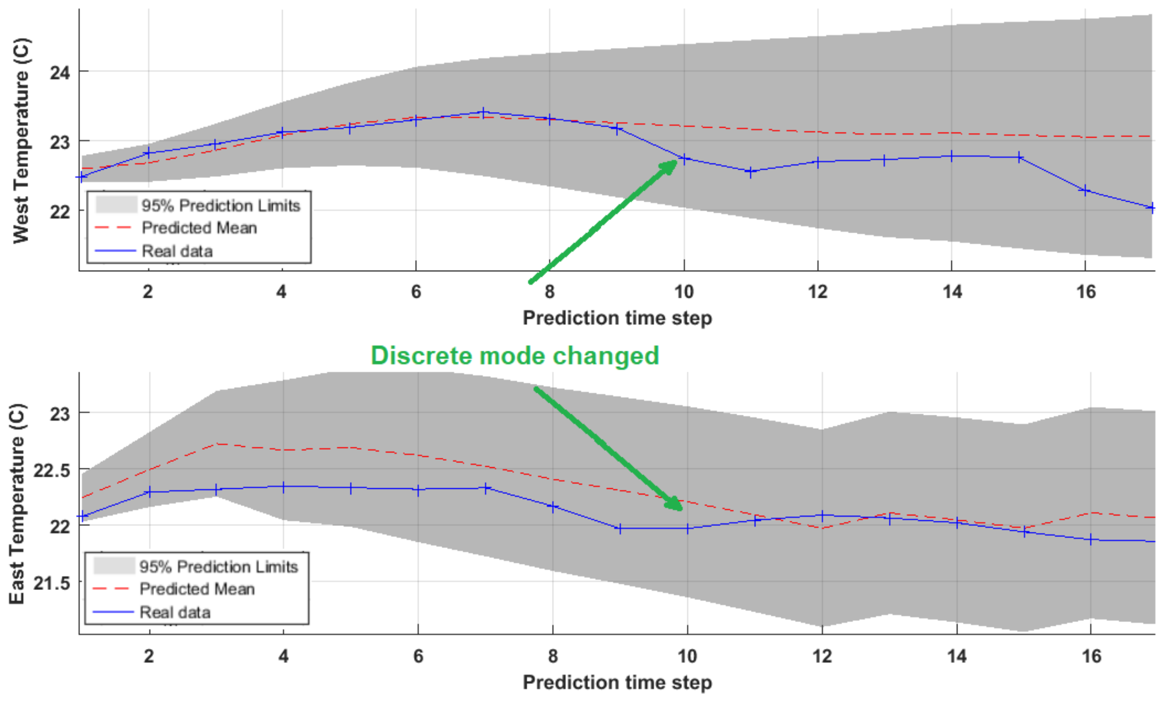


Figure 3.14: Multi-step prediction of zone air temperature for both zones using the proposed SHS model

than the unimodal GP as far as the system does not change its discrete mode. Moreover, the variance of the unimodal GP model is falsely optimistic. The multi-step prediction, shown in this section, is limited to the continuous state in the same discrete mode. In the next chapter, we present a reachability analysis approach of SHS, which provides a multi-step prediction for both the continuous and the discrete states.

3.4.2 Five-Zones Office Building

In this experiment, we evaluate the scalability of the framework using a five-zones office building dataset. The dataset is a synthetic dataset generated by EnergyPlus for a single story office building. The office building is a rectangular building with five zones. It has four windows in each facade, and there are two interior glazed doors between the west and the core zone, and between the east and the core zone, as shown in Figure 3.16.

The main thermal sources for all the five zones are the HVAC unit heating and cooling

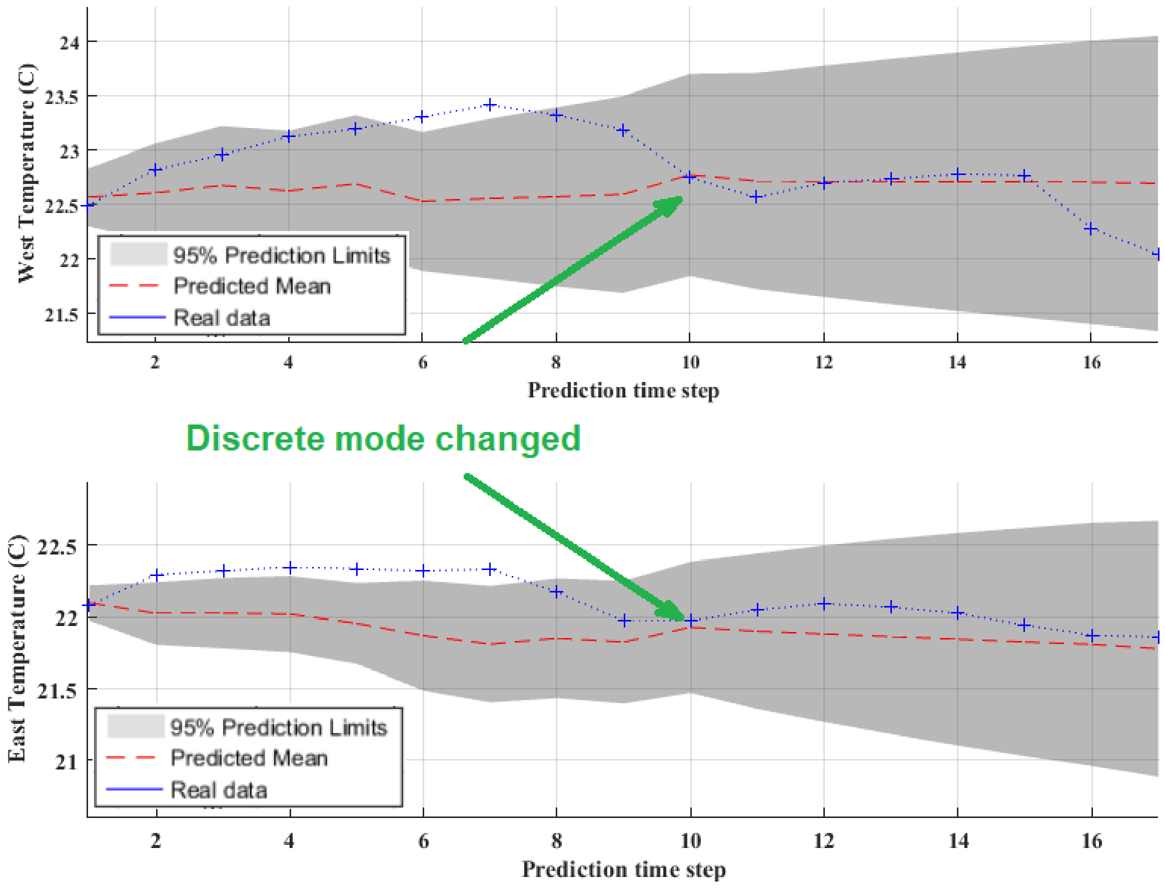


Figure 3.15: Multi-step prediction of zone air temperature for both zones using a typical unimodal GP model

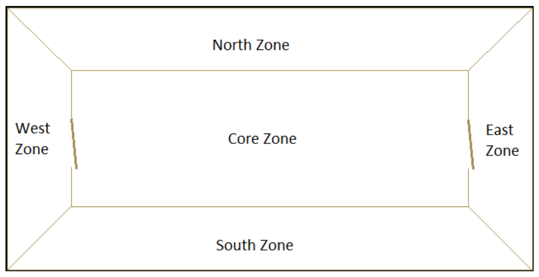


Figure 3.16: Five-zones office building layout

supply air, the office lights and equipment, and the office occupancy. The dataset measures the building thermal behavior hourly for one year. The measurements consist of the ambient temperature, zone air temperatures, cooling/heating rate from the HVAC unit, and heating rate from the thermal load (lights, occupancy, and office equipment) aggregated and averaged for every hour.

The thermal model of the building is represented by the following SHS model: The continuous state ($\mathbf{x} \in \mathbb{R}^5$) represents the zone air temperatures. The discrete state q represents the latent thermal load. The uncontrolled input ($v \in \mathbb{R}$) represents the ambient temperature. Lastly, the control input ($u \in \mathbb{R}$) represents the HVAC heating/cooling rate. Therefore, the predictive distribution of the zone air temperatures is given by

$$\mathbf{x}_{k+1} \sim f_{q_k}(\mathbf{x}_k, u_k, v_k) : q_k \in \mathscr{Q}$$

Like the two-zones building, we also use the first four weeks of data (i.e., 672 data points) to initialize the model and learn its discrete space (i.e., m) and use the heating/-cooling rate (i.e., $u(k)$) and the zone air temperature difference (i.e., $\Delta x = x_k - x_{k-1}$) as time-domain features for the K-mean clustering algorithm. We use the extracted feature to cluster the training data and to identify the discrete states based on the Silhouette analysis. The estimated number of the discrete states of the south, the east, the west and the core zones is two and for the north zone is three. Figure 3.17 shows the clustering of the training dataset for the south and the north zones and Figure 3.18 shows the clustering results of the west, the core, and the east zones.

In the online phase, we use the learned model to predict the zone air temperatures for the next hour. Also, we update the training data and its corresponding models every hour when we receive new data points. We run the prediction/learning steps iteratively for the rest of the data (about 11 months). The results for the discrete mode estimation (i.e., thermal load level) and zone air temperature prediction of the north zone are shown in

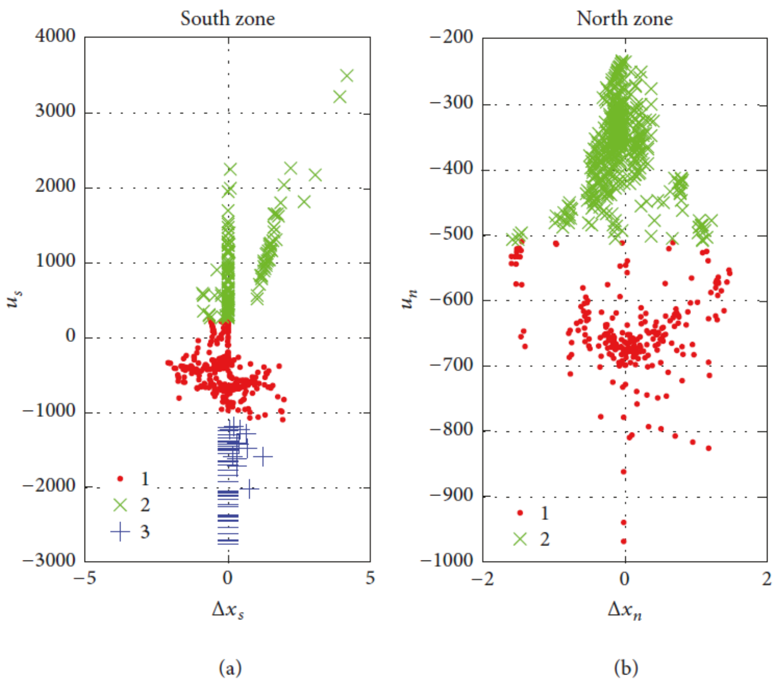


Figure 3.17: *K*-means clustering of the South and the North zones

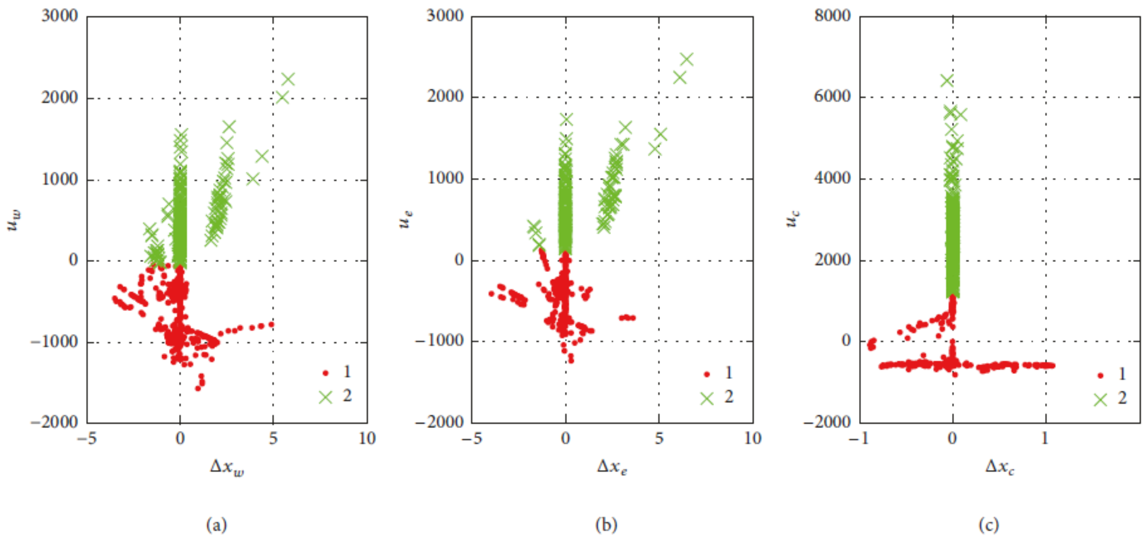


Figure 3.18: *K*-means clustering of the (a) West zone , (b) East zone, and (c) Core zone.

Figure 3.19 and Figure 3.20 respectively. Typically, office environments have two modes: busy and idle, because people tend to go to their offices during the business hours and then the building becomes almost idle during the nights and holidays. The model learning approach estimated this office behavior successfully as shown in the results.

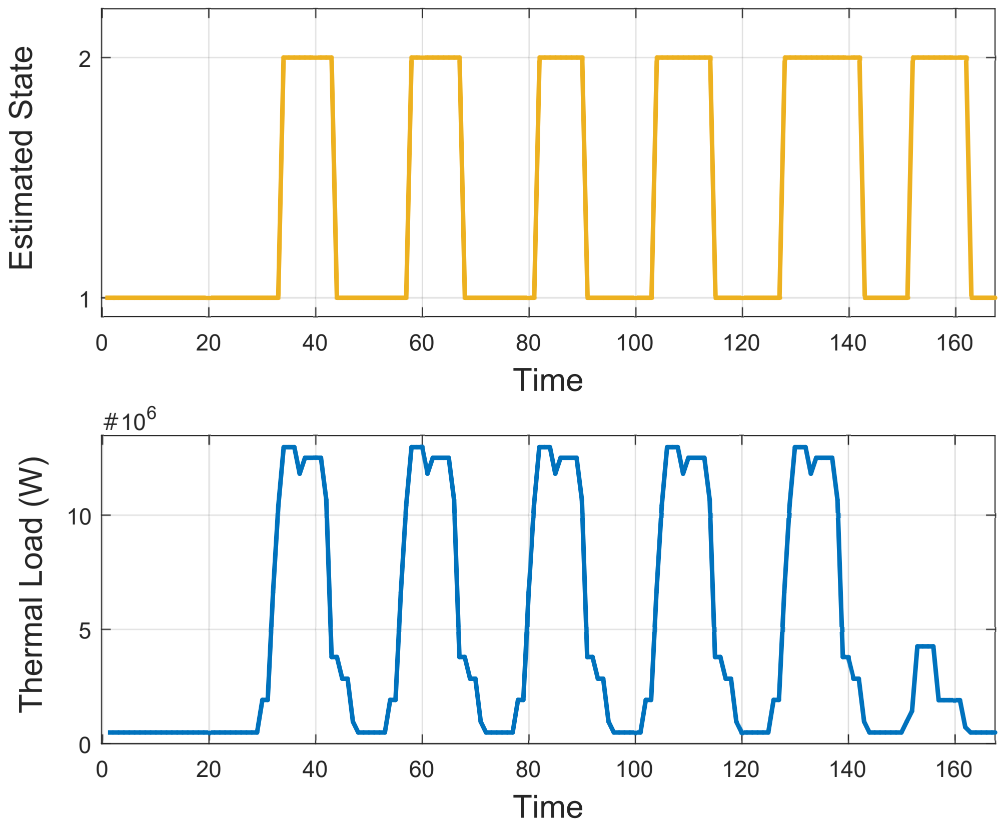


Figure 3.19: North zone discrete state estimation vs the actual total thermal load

To evaluate the prediction performance, we use the root mean square error (RMSE) and the mean relative square error (MRSE) error metrics defined in Equation (3.8). The prediction error evaluation statistics are shown in Table 3.4. These results show that our SHS model predicted the zone air temperatures with a good performance where the average prediction error is less than or equal 6%. Further, the proposed model shows a similar performance to the full GP model which assumes that the thermal load is known and can be measured.

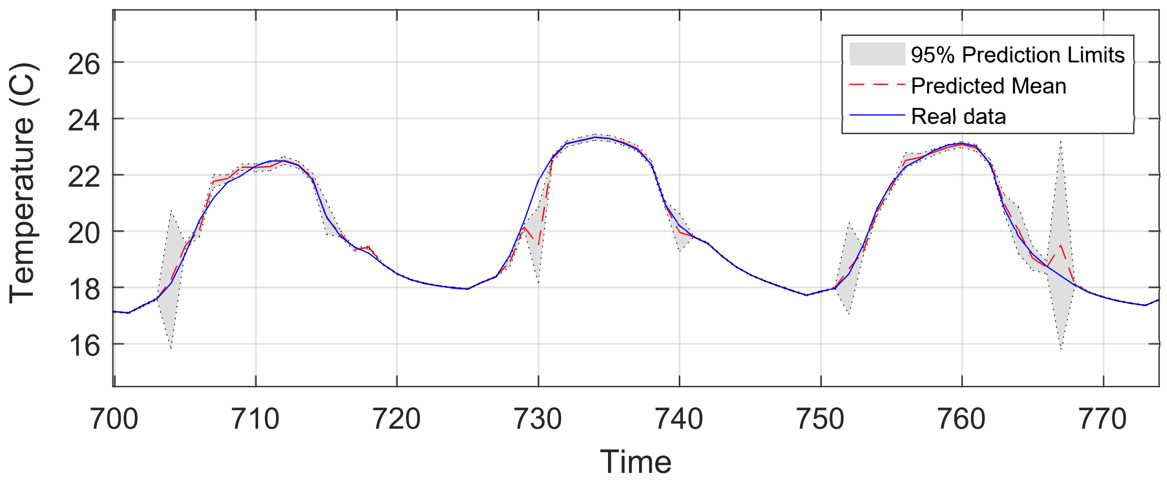


Figure 3.20: The predicted zone air temperature of the North zone

Table 3.4: Prediction error statistics per zone

	South Zone		**East Zone**		**North Zone**		**West Zone**		**Core Zone**	
	Full GP	**SHS**	**Full GP**	**SHS**	**Full GP**	**SHS**	**Full GP**	**SHS**	**Full GP**	**SHS**
RMSE	0.32	0.28	0.25	0.36	0.28	0.29	0.28	0.27	0.22	0.11
MRSE	0.11	0.05	0.08	0.06	0.10	0.05	0.1	0.05	0.6	0.02
Max	3.5	6.3	5.5	10.1	12.1	4.5	6.7	4.5	2.5	3.2

3.4.3 Efficiency and Scalability of Online Learning

Despite the attractive features and properties of GPs, the running time becomes a major factor in the GP performance when the dimension of the training data is large. Learning a GP model requires the inversion of the $n \times n$ covariance matrix where n is the size of the data. The matrix inversion has a complexity of $O(n^3)$. Therefore, GP learning becomes more computationally expensive when the dimension of the model (i.e., number of regressors) and/or the training dataset increase. To address this limitation, a typical solution is to divide or distribute the computation. We mitigate this limitation by approximating the thermal load as a discrete state. Therefore, we divide the training data to learn a distinct GP model independently for each mode. Further, in the context of buildings modeling, the model dimension can be decreased by using the adjacent zones only in the regressors regardless the total number of zones in the buildings, (i.e., $x^i_{k+1} = f_{q_k}(\bar{\mathbf{x}}^i_k, u_k, v_k) : q_k \in \mathcal{Q}$

where $\bar{\mathbf{x}}_k^i$ is the zone air temperature of the adjacent zones of zone i). This approximation is acceptable because the zone air temperature is conditionally independent for other zones given the adjacent zones air temperature.

To evaluate the model scalability, we used EnergyPlus to generate data for five different buildings with a different number of zones. For each building, we use the vector of zone air temperatures as the continuous state variable, therefore, the model dimension increases as the number of zone increases. We use a fixed training data size for all the buildings to learn a GP model of the thermal dynamics. Figure 3.21 shows the variation of the learning time of the GP model versus the number of zones in the building.

We also evaluated the performance and the efficiency of the model learning with respect to the training data size. To do so, we learn both the SHS and the full GP model for the north zone, in the five-zones building, online every hour for one week using different dataset sizes. Figure 3.22 compares the model learning time and the prediction error between the full GP model and the proposed SHS model. The results show that both models have a similar prediction performance, however, the proposed SHS model is faster than the full GP model. The SHS is faster because the training data are segmented into two smaller dataset for each level of the thermal load and then used to learn two distinct GP models. This segmentation distributes the computation of the GPs in the SHS model. In contrary, the full GP uses a single GP with all data at once. Further, the SHS model approximates the thermal load as a discrete state instead of an input which reduce the model dimension. Despite these approximations, the SHS prediction performance is almost similar to the full GP performance.

3.5 Conclusions

In this chapter, we propose a nonparametric SHS modeling framework with a clustering-based online learning approach. The proposed model can be used to build thermal model for buildings when the thermal load is latent (i.e., the thermal load can not be measured).

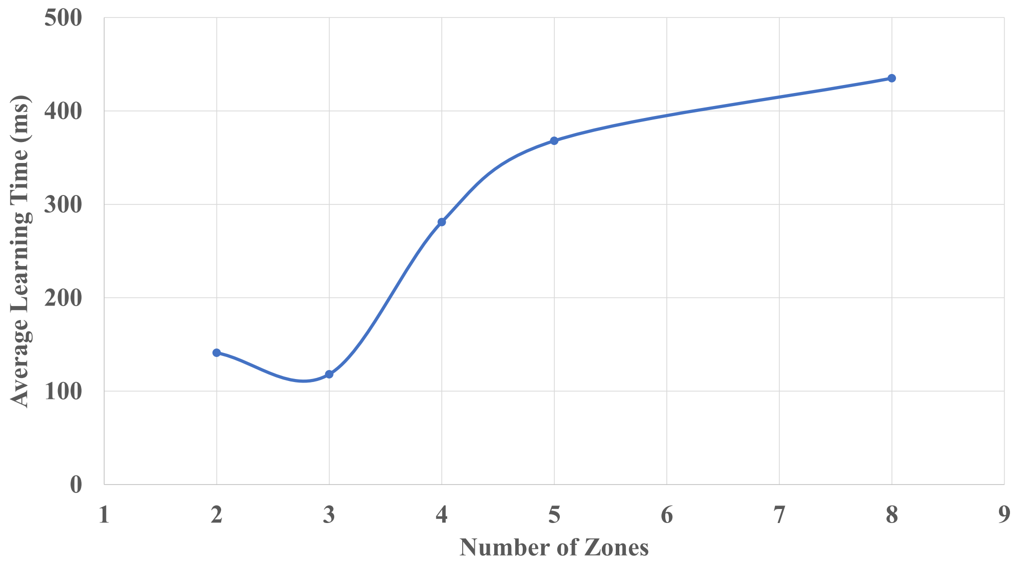
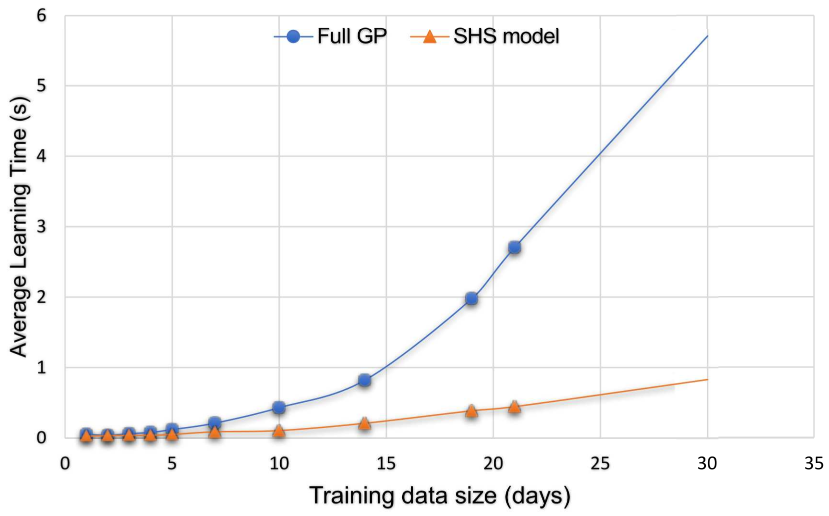
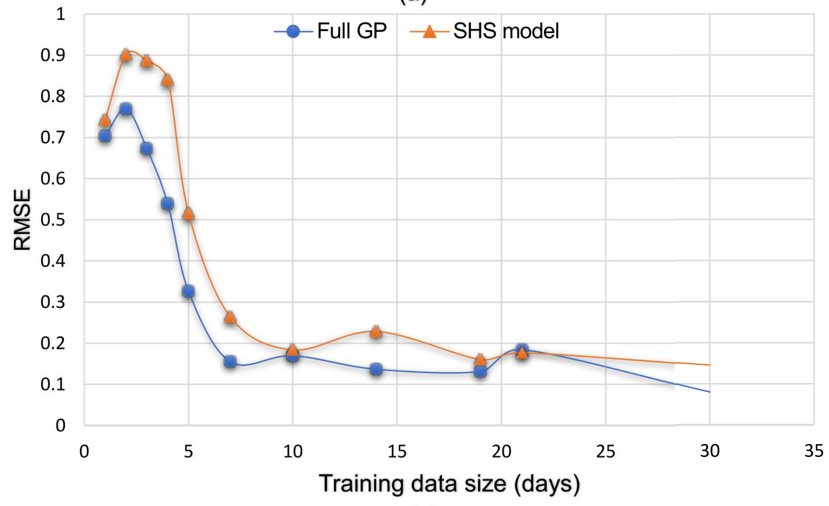


Figure 3.21: Average learning time of one GP model for different buildings with different number of zones

We evaluate the performance of our model by learning thermal models for buildings. Online learning is useful to adapt our model to time-variability behavior and to improve the model efficiency because the training data in the offline phase is not necessarily complete. Despite these advantages, online learning dictates real time constraints. Experimental results demonstrate the efficiency of the proposed model learning approach. The learning process runs online with an adequate computation time. The average learning time for one GP model is 140 ms and 368 ms for the two-zones building and the five-zones building; respectively. Further, the average prediction time is 1 ms for both buildings.



(a)



(b)

Figure 3.22: Evaluating model learning efficiency and performance of the North zone in the five-zones building using different training dataset sizes, with a comparison between the full GP model and the SHS model. (a) Average learning time (b) Prediction error statistics using RMSE

Chapter 4

Reachability Analysis of Stochastic Hybrid Systems

Robust and efficient modeling and reachability analysis of stochastic hybrid systems for control and decision is very demanding and challenging task. Reachability analysis is a typical problem in SHS to verify system safety and stability. Typically, reachability analysis is based on predicting the reachable states for a finite time horizon. Prediction of the system behavior of SHS is a difficult task because SHS have coupled continuous and discrete dynamics with uncertain behaviors. In this chapter, we present our novel methodology for reachability analysis of SHS using a data-driven approach based on mixtures of Gaussian processes (MGP). The proposed method uses the observed data to learn/update the SHS model in an online fashion. Consequently, the updated model is used to approximate the reachable states for a finite horizon by a mixture of Gaussian processes. Moreover, we demonstrate the proposed approach using prediction of the thermal behavior of smart buildings. The simulation results show that our model can adapt to system uncertainty and variability and predict its reachable states efficiently.

In contrast to the previous chapter, we learn nonparametric SHS based on coupled Gaussian processes and periodic Markov chains. These coupled models are used to capture the coupled continuous and discrete dynamics of SHS. In particular, Gaussian processes capture the stochastic nonlinear continuous dynamics for different discrete states and the periodic Markov chain captures the periodic transitions of the discrete states. Further, we solve the reachability analysis problem to estimate the reachable states for both the discrete and the continuous states instead of multi-step prediction of SHS within the same discrete mode.

This chapter is organized as follows: Section 4.1 formalizes the proposed nonparametric SHS. Section 4.2 discusses the model learning and the reachability analysis problem of SHS. Section 4.3 illustrates the proposed data-driven online learning and reachability analysis approach to approximate the reachable states for a finite horizon using an MGP. Section 4.4 illustrates the implementation and the evaluation of the proposed method in smart buildings application. Finally, Section 4.5 shows a use case of the reachability analysis problem of SHS when the discrete dynamics is deterministic and known.

4.1 Nonparametric Stochastic Hybrid Systems

Let's denote $\mathscr{Q}$ the set of discrete states and $\mathbb{R}^D$ the continuous state space. The system hybrid state space is defined as $\mathscr{S} = \mathscr{Q} \times \mathbb{R}^D$. The continuous dynamics evolves according to a stochastic process modeled by a GP which depends on the current discrete mode ($q \in \mathscr{Q}$). The discrete state also evolves based on a stochastic process $\delta : \mathscr{Q} \times \mathscr{Q} \to [0,1]$ represented by a periodic Markov chain (MC). Periodic MCs represent periodic system behavior (e.g., based on hour-of-day, day-of-week, or seasonal effects in buildings applications). We consider systems with two inputs: Control inputs and external uncontrolled inputs (disturbances) from the environment. The control inputs are typically determined by a control policy ($\pi(\mathscr{S}) : \mathscr{S} \to \mathscr{U}$) which maps the hybrid state space ($\mathscr{S}$) into the control input space ($\mathscr{U}$). The external inputs ($v \in \mathscr{V}$) affect the system dynamics and are modeled as a time-series disturbance model ($E : \mathbb{N} \to \mathscr{V}$). The SHS model is formalized as follows:

Definition 3. *(Nonparametric SHS). A nonparametric SHS model is defined as a tuple* $\mathscr{H} = (\mathscr{Q}, X, Init, \mathscr{U}, \mathscr{V}, \mathrm{A}, \delta)$*:*

- $\mathscr{Q} := \{q_1, q_2, \cdots, q_m\}$*, for some* $m \in \mathbb{N}$*, represents the discrete state space.*
- X *is a set of continuous variables in the Euclidean space* $\mathbb{R}^D$*.*
- *Init:* $\mathscr{B}(\mathscr{S}) \to [0,1]$ *is an initial probability measure on the Borel space* $\mathscr{B}(\mathscr{S})$ *where* $\mathscr{S} := \mathscr{Q} \times \mathbb{R}^D$*.*
- $\mathscr{U} \subset \mathbb{R}^E$*, for some* $E \in \mathbb{N}$*, represents the control input space.*

- *$\mathscr{V} \subset \mathbb{R}^F$, for some $F \in \mathbb{N}$, represents the external uncontrolled input space.*

- *$\mathscr{A}$ assigns to each discrete state $q \in \mathscr{Q}$ a function $(x_{k+1} = f_q(x_k, u_k, v_k))$ modeled by a GP which represents the evolution of the continuous state given the predecessor continuous state $x_k \in \mathbb{R}^D$, a control input $u_k \in \mathscr{U}$ and an external uncontrolled input $v_k \in \mathscr{V}$.*

- *δ is a stochastic process of the discrete state $\{q_k : k \geq 0, q_k \in \mathscr{Q}\}$ represented by a periodic MC such that $p(q_k|q_{k-1}, q_{k-2}, \cdots q_0) = p(q_k = i|q_{k-1} = j) = p_{ij}(k), \forall i, j \in \mathscr{Q}$.*

Model Execution

For a finite time horizon $[0,N]$, a system trajectory is denoted by $\{s(k) = (q(k), x(k)), k \in [0,N]\}$ which is an execution of $\mathscr{H}$, with a control policy $\pi(\mathscr{S})$ and a time-series disturbance model $E(k)$. A trajectory can be easily obtained by simulating the model (i.e., calculate the control input, forecast the external input, and evaluate the continuous/discrete states) for the required horizon. A discrete-time execution algorithm of $\mathscr{H}$ is shown in Algorithm 1.

Algorithm 2 Discrete-time SHS execution algorithm

```
State Initialization: $s_0 = (q_0, x_0) \in Init$
k ← 0
while $k < N$ do
    ▷ Evaluate the control input $u_k$:
    $u_k \leftarrow \pi(s_k)$
    ▷ Forecast the external input $v_k$:
    $v_k \leftarrow E_k$
    ▷ Update the discrete mode $q_{k+1}$:
    $q_{k+1} \leftarrow \delta(q_k)$
    ▷ Update the continuous state $x_{k+1}$:
    $x_{k+1} \leftarrow f_{q_{k+1}}(x_k, u_k, v_k) \sim \mathscr{GP}_{q_{k+1}}$
    k ← k +1
end while
```

4.2 Problem Formulation

Online learning and reachability analysis require updating the system model $\mathscr{H}$ and predicting the system states iteratively and efficiently after receiving new measurements as shown in Figure 4.1.

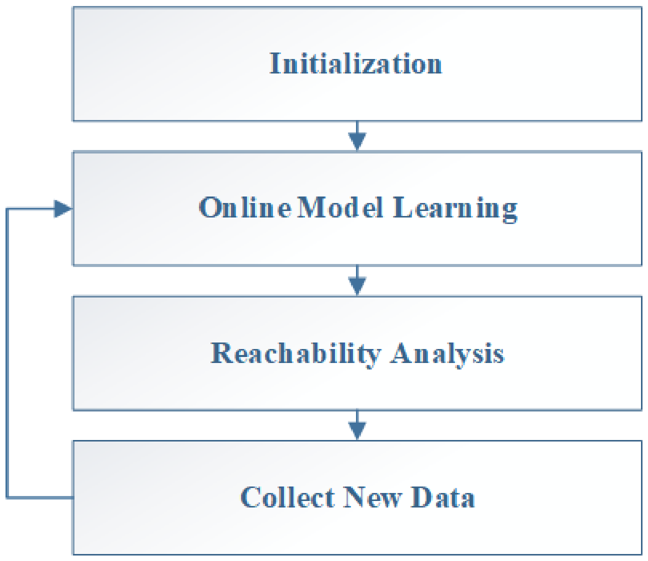


Figure 4.1: Online learning and reachability analysis of SHS

Learning the SHS model $\mathscr{H}$ requires identifying the discrete dynamics (i.e., the discrete state space $\mathscr{Q} := \{q_1, q_2, \cdots, q_m\}$ and the discrete transition function δ), and the continuous dynamics (i.e., $\mathscr{GP}_q(.)$ for all $q \in \mathscr{Q}$) from the observed data ($\mathscr{D}$) collected from the physical system and the environment. At each time step k, the observed data include measurements of the continuous state $\mathbf{x}_k$, the control inputs u_k, and the external disturbances v_k. Thus, the training dataset is defined as:

$$\mathscr{D} := \{(\hat{\mathbf{x}}_k, \mathbf{y}_k) : k = T_s, \cdots, T_e\}$$

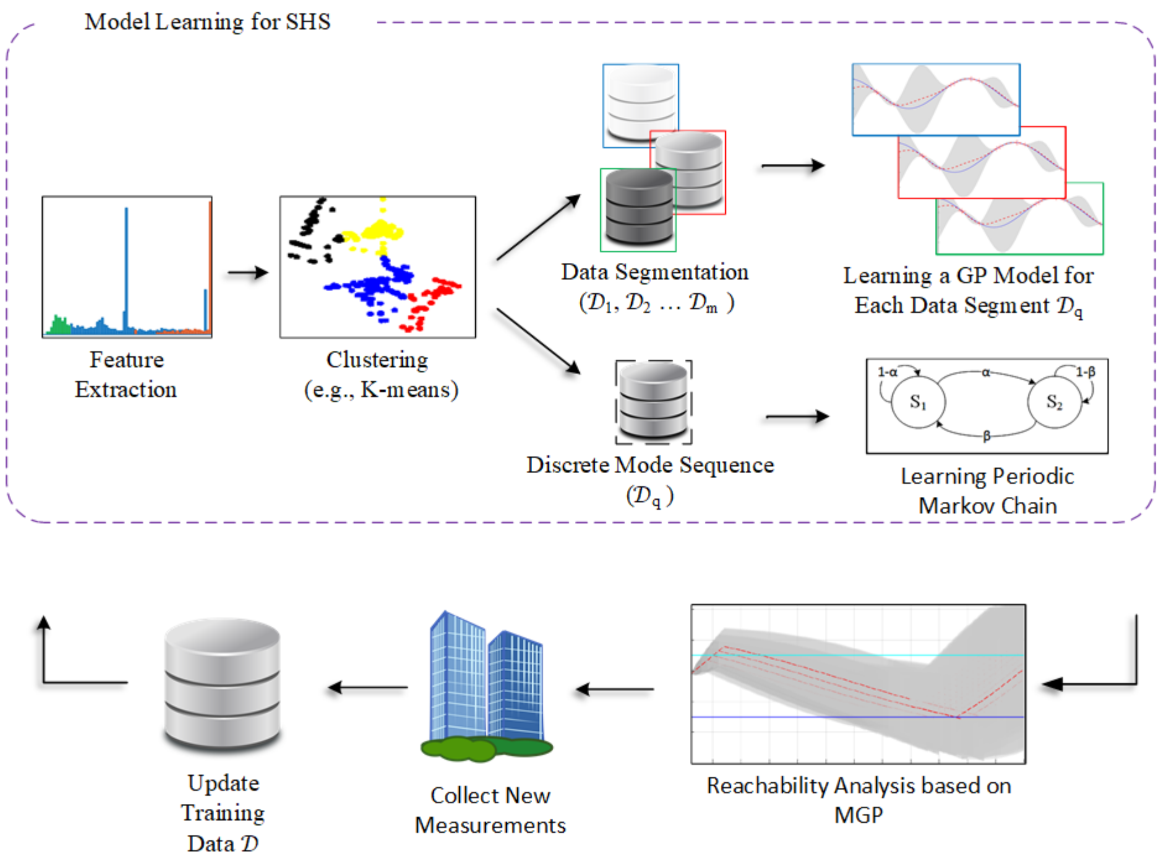
where $\mathbf{y}_k$ is the successor continuous state (i.e., $\mathbf{y}_k = \mathbf{x}_{k+1}$), $\hat{\mathbf{x}}_k$ is the tuple $(\mathbf{x}_k, u_k, v_k)$, and $[T_s, T_e]$ is the time period at which the data are collected. In general, learning $\mathscr{H}$ using $\mathscr{D}$ is a challenging task because the sensory data do not necessary include explicit measurements of the discrete state (e.g., thermal load in smart buildings applications). Moreover, the model must capture the uncertainty and the variability of the system and the environmental disturbances.

Reachability analysis aims at predicting the probabilities of the reachable states for a finite time horizon given an initial state $s(0)$. This multi-step prediction is useful in many applications to evaluate the design (e.g., control policy) and/or to develop advanced control methodology. For instance, prediction the zone air temperature in buildings is important to ensure the user comfort for a given control policy. Therefore, our objective is to calculate $P(s(k)|s(0)), \forall k \in [1,T]$ for the finite receding horizon $[1,T]$, given a SHS model $\mathscr{H}$, a control policy $\pi(\mathscr{S})$ and a time-series disturbance model $E(k)$. Prediction of reachable states can be performed as an iterative process since $s(k)$ depends on $s(k-1)$. However, this is a challenging task because: (1) Predicting the continuous state $x(k)$ distribution requires prediction at an uncertain input since it depends on the distribution of $x(k-1)$, and also, the continuous state $x(k)$ evolves differently for each discrete state $q(k)$; (2) the discrete mode $q(k)$ is a discrete random distribution given the probability distribution of $q(k-1)$; and (3) the system trajectory depends on the switching times between the discrete states.

4.3 Online Learning and Reachability Analysis

To overcome these challenges, we propose a novel approach, which consists of three steps: (1) Collect data from the system and update the training data; (2) learn both the system model $\mathscr{H}$ and the time-series disturbance model $E(k)$; and (3) predict the reachable hybrid states of the system for a receding finite horizon. These steps are repeated iteratively in an online fashion as depicted in Figure 4.2.

Initially, we collect training dataset ($\mathscr{D}$) which consists of measurements of the continuous state, the control input, and the external disturbance. This dataset is used to initialize the system model. As new measurements are collected, we update the training dataset us-



Model Learning for SHS
Feature Extraction
Clustering (e.g., K-means)
Data Segmentation ($\mathcal{D}_1$, $\mathcal{D}_2$ … $\mathcal{D}_m$)
Learning a GP Model for Each Data Segment $\mathcal{D}_q$
Discrete Mode Sequence ($\mathcal{D}_q$)
1-α
α
1-β
β
S_1
S_2
Learning Periodic Markov Chain
Reachability Analysis based on MGP
Collect New Measurements
Update Training Data $\mathcal{D}$

Figure 4.2: Overview of the proposed approach

ing a moving window technique based on first-in-first-out (FIFO) policy. The FIFO policy maintains a fixed size of the training dataset which is then used to learn/update the system model in an online fashion as described in the next subsections.

4.3.1 Model Learning

In the model learning steps, we first identify the discrete modes of the SHS model using a clustering algorithm. Then, we segment and label the training data based on the identified modes. This allows us to learn the continuous dynamics for each discrete state using a distinct GP model. Moreover, we use the labeled data to learn the discrete dynamics using a periodic MC model. In comparison with the model learning algorithm in Chapter 3, we additionally learn the discrete dynamics using a periodic MC to facilitate multi-step prediction of the hybrid discrete/continuous states. In this section, we describe learning the discrete dynamics step only, the remaining learning steps (i.e., feature extraction, data clustering, data segmentation and learning GP models) are illustrated in Section 3.3.

Learning Discrete Dynamics

We illustrate the learning method of the discrete dynamics (i.e., $\delta(.)$), which represents the stochastic transitions between discrete modes. We represent $\delta(.)$ using a periodic MC and we consider SHS whose discrete state transitions are independent of the control signals and the continuous state. A typical MC has a stationary matrix that does not capture any periodic or time-dependent behaviors explicitly. However, the discrete dynamics of SHS models of buildings exhibit periodic behavior (e.g., occupancy patterns depend on the time of the day). Therefore, we represent the discrete dynamics using a periodic MC with non-stationary transition probabilities. The transition matrix can be used to calculate the probability of the discrete modes as follows:

$$p(q_{k+1}) = \delta(q_k) = p(q_k)A_{h(k)}$$

where $h(k) : k \to \{1, 2, \cdots, H\}$ maps the time-step k to the time of the day (e.g., hour of the day $\{1, 2, \cdots, 24\}$), and $A_{h(k)}$ is its associated transition matrix. A graphical representation of the periodic MC with 24 transition matrices (i.e., $H = 24$) and two states is depicted in Figure 4.3.

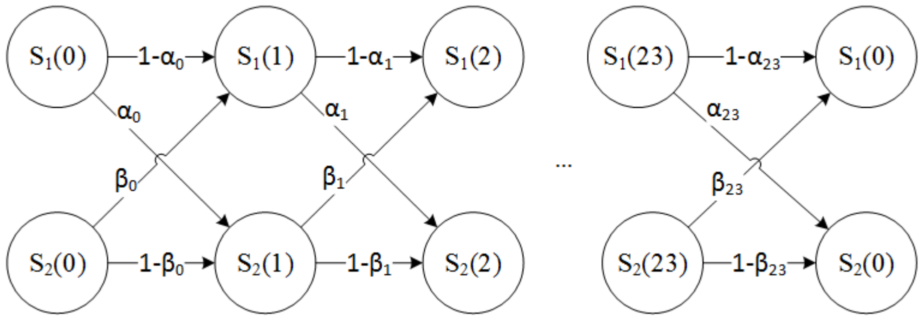


Figure 4.3: State-transitions diagram of a 2-state periodic MC model

We learn the periodic MC by identifying the parameters of the transition matrices (i.e., transition probabilities). We use the sequence of labels of the training dataset $\mathscr{D}_q$ from the previous step (clustering):

$$\mathscr{D}_q := \{q_k : q_k = \underset{q' \in \mathscr{Q}}{\arg\min} \parallel g(\hat{\mathbf{x}}_k) - c_{q'} \parallel^2, \\ \forall (\hat{\mathbf{x}}, \mathbf{y})_i \in \mathscr{D}, k = \{0, 1, \cdots, M\}\}$$

where $c_{q'}$ is the cluster centroid of the discrete mode q', $g(\hat{\mathbf{x}})$ is the feature vector of the data point $\hat{\mathbf{x}}$ and M is the size of the training dataset. $\mathscr{D}_q$ is used to learn/update the model parameters (i.e., the transition probabilities) for each transition matrix $A_i, i \in \{1, 2, \cdots H\}$. Each transition matrix A_i is identified independently by counting all the distinct sequences $(q_k, q_{k+1}) \in \mathscr{D}_q$ such that $h(k) = i$. So, its transition probabilities can be computed by:

$$a_{ab} = \frac{\text{total number of } q(a)q(b) \text{ occurrence}}{\text{total number of } q(a) \text{ occurrence}} \tag{4.1}$$

4.3.2 Mixtures of Gaussian Processes for Reachability Analysis

We represent the reachable states of $\mathscr{H}$ using mixtures of Gaussian processes (MGP) [97]. An MGP consists of a latent discrete variable, typically called gating network, and a set of GP functions, called the experts. The state of the discrete variable specifies the GP function used to calculate the system output at a given input. The MGP model is expressed as:

$$P(y|x) = \sum_{i=1}^{Z} P(z=i|x)\mathscr{GP}_i(m_i(x), k_i(x,x)) \tag{4.2}$$

where y is the output, x is the input, z is the discrete latent variable with Z states and $\mathscr{GP}_i$ is the GP function corresponding to the discrete state $z=i$. Our goal is to predict the probability of the continuous state $\mathbf{x}(k+1)$ given the probability of the hybrid state $s(k) = (\mathbf{x}(k), q(k))$. Thus, the one-step state prediction of SHS can be defined as:

$$P(\mathbf{x}(k+1)|s(k)) = \sum_{i=1}^{m} P(q(k+1)=i|q(k)) f_i(\hat{\mathbf{x}}(k)) \tag{4.3}$$

where $f_i(\hat{\mathbf{x}}(k)) \sim \mathscr{GP}_i(m_i(\hat{\mathbf{x}}), k_i(\hat{\mathbf{x}}, \hat{\mathbf{x}}))$ with $\hat{\mathbf{x}}$ defined as the tuple $(\mathbf{x}, u, v)$ and $P(q(k+1) = i|q(k)) \sim \delta(q(k))$ is the probability distribution of the discrete state at time-step $k+1$. In order to predict the probability distribution of $s(k), \forall k \in [1,T]$ for the finite-horizon T, we can apply (4.3) iteratively. However, this equation depends on $p(\mathbf{x}(k))$ which is represented by a Gaussian mixture model from the previous iteration (i.e., $p(\mathbf{x}(k)|s(k-1))$). Formally, let's define $p(\mathbf{x}(k))$ as:

$$p(\mathbf{x}(k)) = \sum_{j=1}^{C} w_j \mathscr{N}(\boldsymbol{\mu}_j, \Sigma_j) \tag{4.4}$$

where C is the number of Gaussian distribution components in the mixture, w_i is the weight of i^{th} Gaussian component with $\sum_{i=1}^{C} w_i = 1$, and $\boldsymbol{\mu}_i$, and Σ_i are the mean and the variance of i^{th} Gaussian component; respectively. Calculating $P(\mathbf{x}(k+1)|s(k))$ iteratively from Equation (4.3) and (4.4) is analytically intractable because the input of the MGP model in (4.3) is uncertain (represented by a mixture of Gaussian probability distributions as illustrated in Equation (4.4)). To overcome this limitation, we approximate the predictive distribution by propagating every Gaussian component in (4.4) independently as illustrated in Equation (2.7). Hence, the predictive distribution can be obtained as:

$$P(\mathbf{x}(k+1)) = \sum_{i=1}^{Q}\sum_{j=1}^{C} w_j P(q(k+1)=i|q_j(k))\tilde{f}_i(\mathbf{x}_j(k), u_j(k), v(k)) \tag{4.5}$$

where $\mathbf{x}_j$ is the j^{th} Gaussian component of $p(\mathbf{x}(k))$ with weight w_j, mean $\boldsymbol{\mu}_j$ and variance Σ_j, $q_j(k)$ is the discrete mode of the j^{th} Gaussian component and $\tilde{f}(.)$ is the approximation of GP posterior f_i defined in Equation (2.7). Algorithm 3 illustrates the prediction of the reachable states of SHS iteratively based on Equation (4.5).

Algorithm 3 Discrete-time SHS state prediction

```
Input: $s(0), N$
Output: $p(s(k))$ for $k \in [1,N]$
  Load GP function $f_{q(0)} \sim \mathcal{GP}_{q(0)}$
  k = 0
  while $k < N$ do
      ▷ Forecast the external input at time k
      $v(k) \leftarrow E(k)$
      for each $s_c(k) \in s(k)$ do
          for each $q \in \mathcal{Q}$ do
              ▷ Calculate the probability of the discrete state
              $p(q_c(k+1) = q|q_c(k)) \leftarrow \delta(q_c(k))$
              ▷ Calculate the new weight
              $w_c(k+1) \leftarrow p(q(k+1) = q_c(k+1)|q_c(k)) \times w_c(k)$
              ▷ Ignore component with very small probability
              if $w_c(k+1) > \delta_w$ then
                  ▷ Calculate the control input
                  $u_c(k) \leftarrow \pi(x_c(k), q_c(k+1))$
                  ▷ Predict the continuous state
                  $x_c(k+1) \leftarrow \tilde{f}_{q_c(k+1)}(x_c(k), u_c(k), v(k))$
                  $s_c(k+1) \leftarrow [x_c(k+1), w_c(k+1), q_c(k+1)]$
                  add $s_c(k+1)$ to $s(k+1)$
              end if
          end for
      end for
  end while
```

The prediction algorithm approximates the probability distribution of the discrete state by ignoring the discrete transitions which have probabilities less than δ_w. Moreover, it approximates the prediction of the continuous state by linearizing the posterior GP mean function. The accuracy of the prediction algorithm can be increased by tuning the threshold δ_w and/or by approximating the GP posterior using the exact moments [27]. The prediction algorithm is efficient and can run in an online fashion. The most expensive part is computing the inverse covariance matrix which requires $\mathcal{O}(n^3)$ time where n is the size of the data. Many studies have been conducted to improve GP complexity using different approximation algorithms. For instance, sparse Gaussian processes are developed to approximate the inverse of the covariance matrix K with a low rank matrix approximation of dimension $m \times m$, (where $m << n$) [81].

4.4 Evaluation

In this section, we demonstrate the efficiency of the proposed method on multi-zones buildings. We have implemented the approach using MATLAB® and the Statistics and Machine Learning Toolbox Release 2017a [60]. We evaluate the reachability analysis algorithm for: (1) A two-zones data center building and (2) a five-zones office building. We represent the physical behaviors of the system using EnergyPlus software [22]. EnergyPlus is an open-source cross-platform building energy simulator engine funded by the U.S. Department of Energys (DOE), and Building Technologies Office (BTO), and managed by the National Renewable Energy Laboratory (NREL). EnergyPlus is used by engineers, architects, and researchers for high fidelity simulation of buildings. EnergyPlus requires two inputs: (1) The ambient temperature and the environment data and (2) The building description. The building description defines its structure and layout, the construction materials, the thermal zones with their dimensions and area, the HVAC system, the control strategies and more. It also defines the building thermal loads with their schedules such as occupancy, lights, and electrical equipment. These detailed descriptions are used to construct several models (e.g., airflow network model, pollution model, on-site power model) by which EnergyPlus simulates the building behavior. Although such models can be used for high-fidelity simulation, multi-step prediction is only possible using Monte Carlo techniques.

In the following experiments, we initially generate data with a time-step of 1-hour and set the window size of the training data to four Weeks (i.e., $M = 672$). In addition to the initial 1-hour sampling period, we consider evaluating the presented approach using different sampling period in the five-zones building case study. The experiment goal is to predict the system behavior for the next day (i.e., the reachability receding horizon $N = 24$). Initially, we train the model using the first 4 weeks of EnergyPlus simulation data, then apply the proposed online approach to predict the system behavior for the next day. Afterward, we collect new data of the system response for the predicted day and update the training dataset to re-learn/update the model. We repeat this periodic 1-day predict/learn steps for rest of the simulated year. The following subsections discuss the experiment results for the two-zones data center building and the five-zones office building.

4.4.1 Two-Zones Data Center Building

The data center building consists of two zones: The West zone and the East-zone. The data center building was simulated for one-year using EnergyPlus to generate its dataset with one-hour sampling rate. Each data point consists of zone air temperature, ambient temperature, thermal load heating rate from IT equipment, lights, and electrical equipment, and HVAC unit cooling rate. The heat rate from the building thermal load depends on the building activities (e.g., how utilized or idle is the IT equipment). We consider the thermal load as a latent discrete state of the system and we use our proposed approach to estimate it from the measurable thermal data (i.e., zone temperature and HVAC unit cooling rate).

In the model learning step, we identify the model discrete states using the K-means clustering algorithm. Data clustering starts by extracting the time-domain features from the data. In this experiment, the used features are the average cooling rate (i.e., $u(k)$) and the zone air temperature difference (i.e., $\Delta x = x_k - x_{k-1}$)(note, the values of these features are normalized in order to unify their scale). Then, the number of the discrete states is estimated using the Silhouette analysis method. Based on this analysis, we identified the number of the discrete states as three for the West zone and five for the East-zone. Furthermore, we clustered the data to associate each point with its corresponding discrete mode and used it to learn the periodic Markov chain model for the discrete dynamic. Finally, we segment the data for each discrete mode and use them to learn distinct GP models for mode.

In the reachability analysis step, we use the proposed algorithm to generate a statistical distribution of the reachable state. The prediction distribution for both the discrete mode (i.e., estimated thermal load level), and the continuous state (i.e., zone air temperature) of three days are shown in Figure 4.4 and Figure 4.5 for the West-zone and the East-zone; respectively.

We compare the prediction distribution of the SHS model against a unimodal GP model. The unimodal GP model does not represent the system modes (i.e., thermal load level) and can be defined as:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, u_k, v_k)$$

where $\mathbf{x}_k$ is the zone air temperature, u_k is the heating/cooling rate from the HVAC unit, and v_k is the ambient temperature. The reachability analysis of the unimodal model is simpler than the SHS model. In this case, the reachable state can be calculated and approximated by propagating the prediction at each time step as illustrated in (2.7). Figure 4.6 and 4.7 show the prediction distribution of the zone air temperature using a single GP for both the West zone and the East zone, respectively.

We evaluated the performance of the reachability analysis using the root mean square

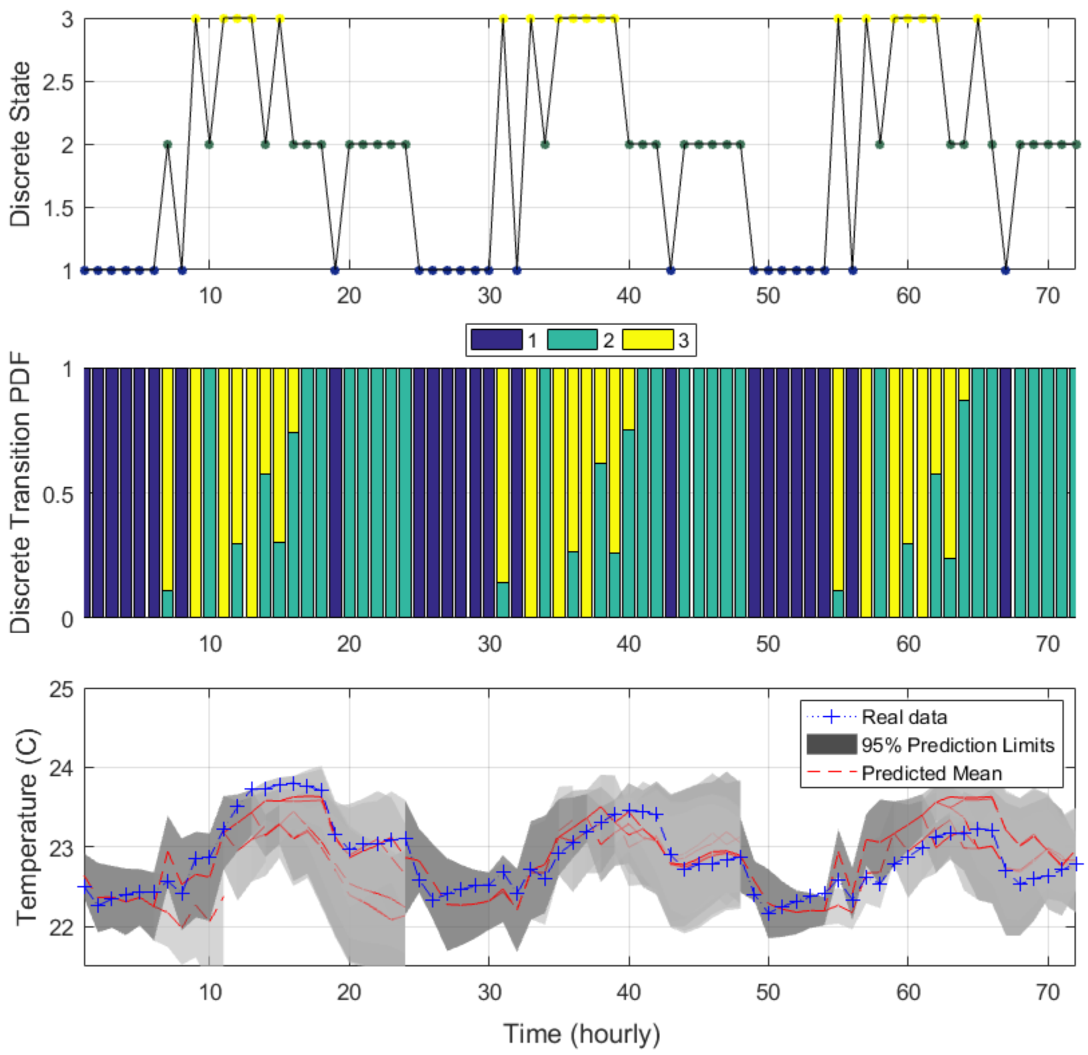


Figure 4.4: Prediction distribution of the hybrid states for the West-zone using the proposed approach

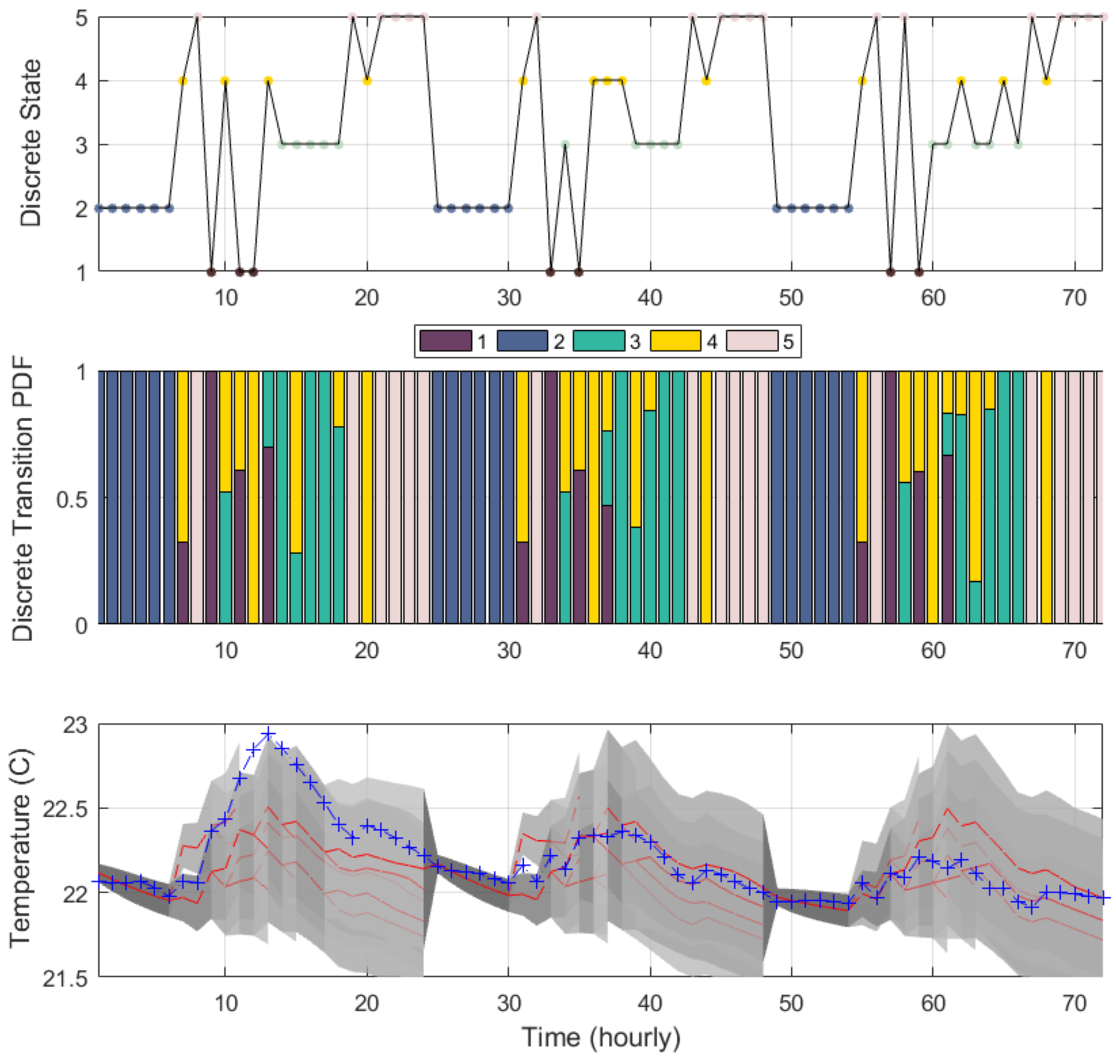


Figure 4.5: Prediction distribution of the hybrid states for the East-zone using the proposed approach

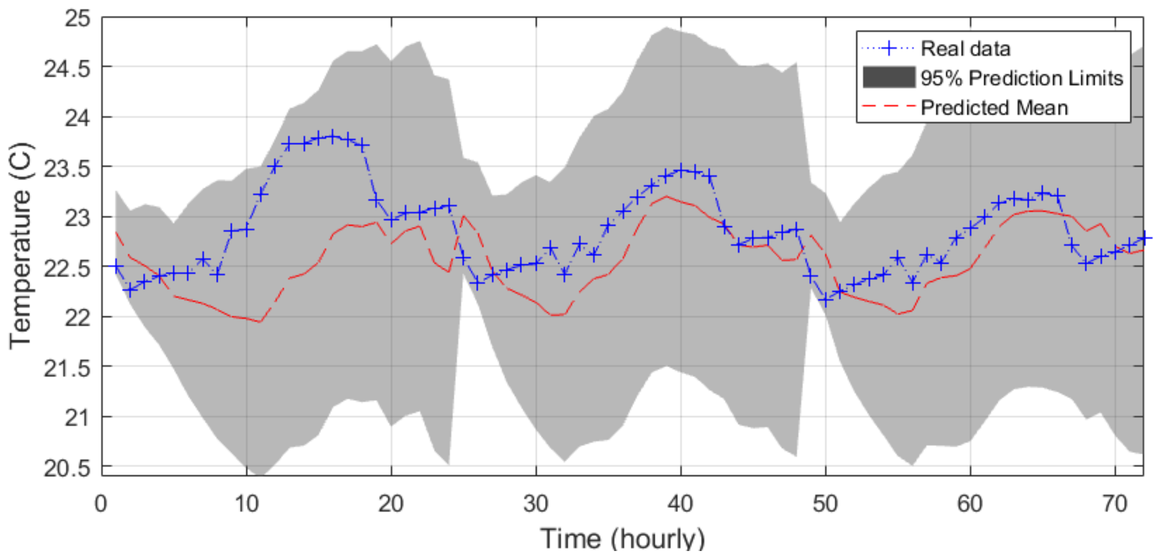


Figure 4.6: Prediction distribution of the West-zone air temperature using a unimodal single GP model

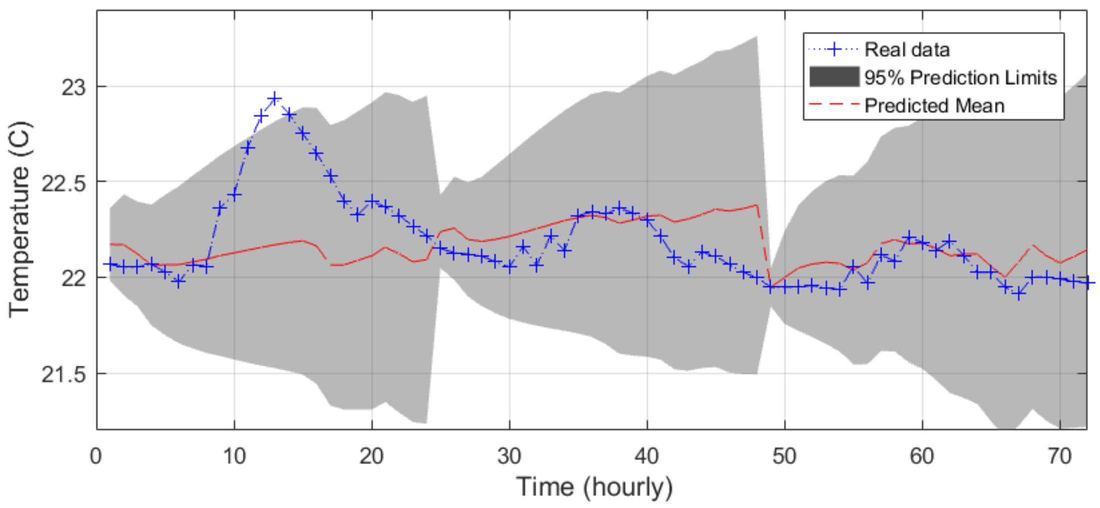


Figure 4.7: Prediction distribution of the East-zone air temperature using a unimodal single GP model

Table 4.1: Performance statistics of the reachability analysis with a comparison against a unimodal GP model

	West Zone		**East Zone**	
	SHS	**GP**	**SHS**	**GP**
RMSE	0.2810	0.7444	0.2034	0.4034
MRSE	0.0123	0.0322	0.0092	0.0180

error (RMSE) and the mean relative square error (MRSE) error metrics, defined as:

$$\begin{aligned} RMSE &= \sqrt{\frac{1}{N}\sum_{k=1}^{N} e(k)^2} \\ MRSE &= \sqrt{\frac{\sum_{k=1}^{N} e(k)^2}{\sum_{k=1}^{N} y(k)^2}} \end{aligned} \tag{4.6}$$

where $y(k)$ is the system output and $e(k) = \sum_c w_c(k)(\hat{y}_c(k) - y(k))$ is the weighted prediction error of the i^{th} time step, evaluated for each Gaussian components using the component mean $\hat{y}_c$ and the component weight w_c . The performance statistics for the SHS model compared against the unimodal GP model are shown in Table 4.1. The results show that the proposed reachability analysis approach has better performance and provides more confidence distribution of the system state because it considers explicitly the effects of the thermal load level and its periodic behaviors.

4.4.2 Five-Zones Office Building

In this experiment, we generate a dataset for a single-story office building with five zones. We simulate the office building based on a realistic occupant schedule for office spaces to evaluate the ability of our model to capture the stochastic discrete dynamics. The occupancy schedule is generated using the occupancy simulator developed at Lawrence Berkeley National Laboratory [30] [1]. This stochastic occupancy schedule is then used by EnergyPlus to simulate the thermal behavior of the office building and generate data for training. The major thermal sources for all the five zones are the HVAC unit heating and cooling supply air, the office lights and equipment, and the office occupancy. The dataset measures the building thermal behavior hourly for one year. The measurements consist of the ambient temperature, zone air temperatures, cooling/heating rate from the HVAC unit, and heating rate from the thermal load (lights, occupancy, and office equipment) aggregated and averaged for every hour.

We set the window size of the training data to four Weeks (i.e., 672 datapoint). The goal is to predict the system behavior for the next day (i.e., horizon $T = 24$). Initially, we train the model using the first four weeks of the simulation data, then we apply the proposed online approach to predict the system behavior for the next day. We collect new data for the predicted day and update the training dataset to re-learn/update the model and we repeat these predict/learn steps iteratively for the whole experiment.

For learning the SHS model, we identify the model discrete states using the K-mean clustering algorithm. Data clustering starts by extracting the time-domain features which are the average heating/cooling rate (i.e., $u(k)$) and the zone air temperature difference (i.e., $\Delta x = x_k - x_{k-1}$). The number of the discrete states is estimated using the Silhouette analysis method such that, the number of the discrete states for the Core-zone, South-zone, and East-zone is three, for the North-zone is two, and for the West-zone is five. Furthermore, we clustered the data to associate each point with its corresponding discrete mode and used it to learn the periodic MC for the discrete dynamics. Finally, we segment the data for each discrete mode and use them to learn distinct GP models for each mode. For reachability analysis, we use the proposed algorithm to generate a distribution of the reachable states. The prediction distribution for both the discrete mode (i.e., estimated thermal load level) and the continuous state (i.e., zone air temperature) of three days are shown in Figure 4.8 for the West zone.

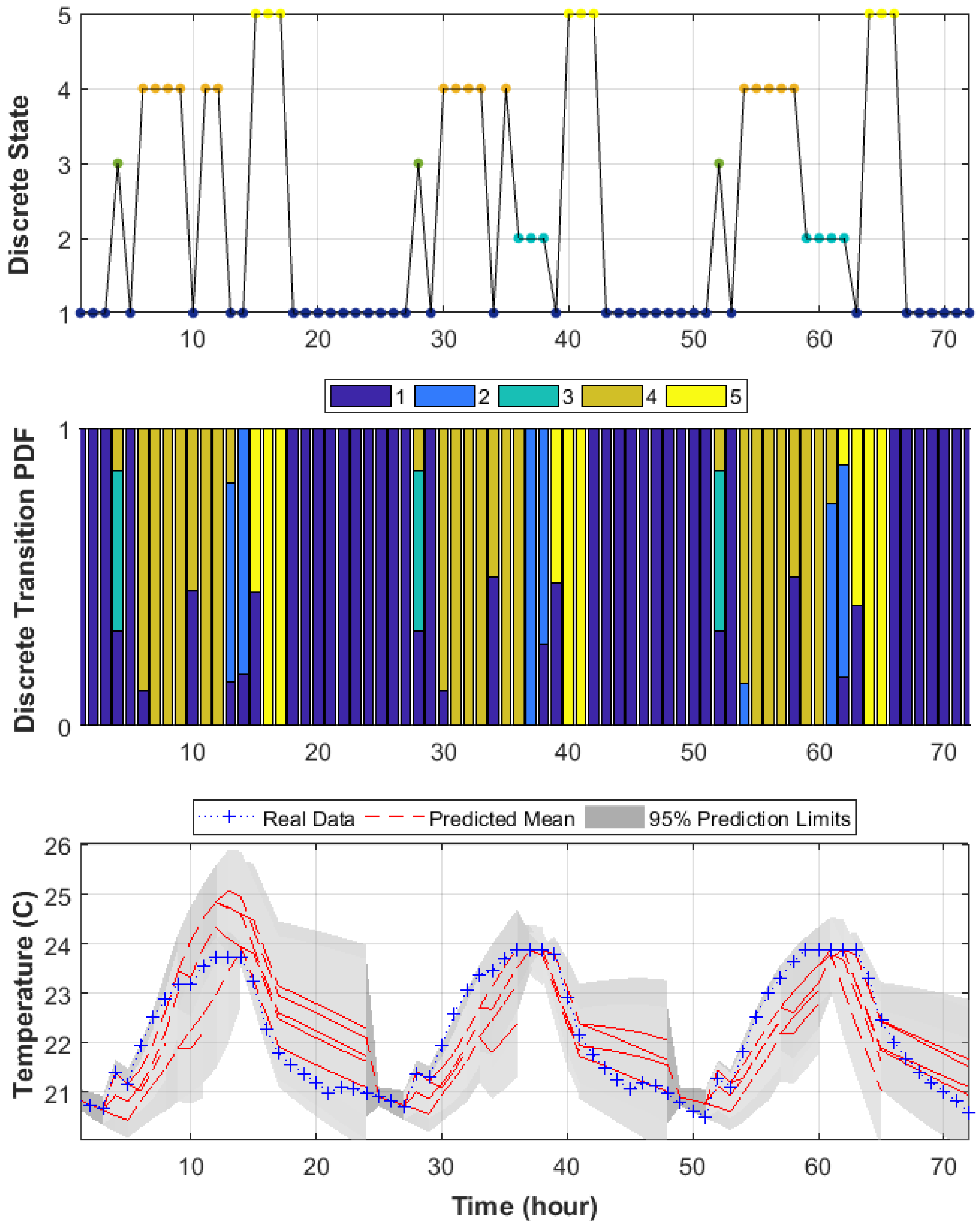


Figure 4.8: Prediction distribution of the hybrid states for the West-zone using the proposed approach

4.4.2.1 *Performance*

We evaluate the performance of the reachability analysis using weighted root mean square error (RMSE) and mean relative square error (MRSE) error metrics, defined in (4.6). Additionally, we compare the prediction of our approach against the prediction obtained using a unimodal single GP model and a full GP model. The unimodal GP model does not have discrete states associated with the thermal load level and it is defined by: $\mathbf{x}_{k+1} = f(\mathbf{x}_k, u_k, v_k)$ where $\mathbf{x}_k$ is the zone air temperature, u_k is the heating/cooling rate from the HVAC unit, and v_k is the ambient temperature. The full GP model assumes that thermal load is measured and it is defined by: $\mathbf{x}_{k+1} = f(\mathbf{x}_k, u_k, l_k, v_k)$ where l_k is the thermal load.

Reachability analysis using the unimodal model is a straightforward multi-step prediction as illustrated in (2.7). The same approach can be used also for the full GP model, however a time-series model should be learned to predict the thermal load for the finite-receding horizon. In our case, we use a time-series GP such that: $l_k \sim \mathcal{GP}(m(k), K(k,k'))$. A one day prediction for the West-zone using the unimodal GP and the full GP model are shown in Figure 4.9 and Figure 4.10; respectively.

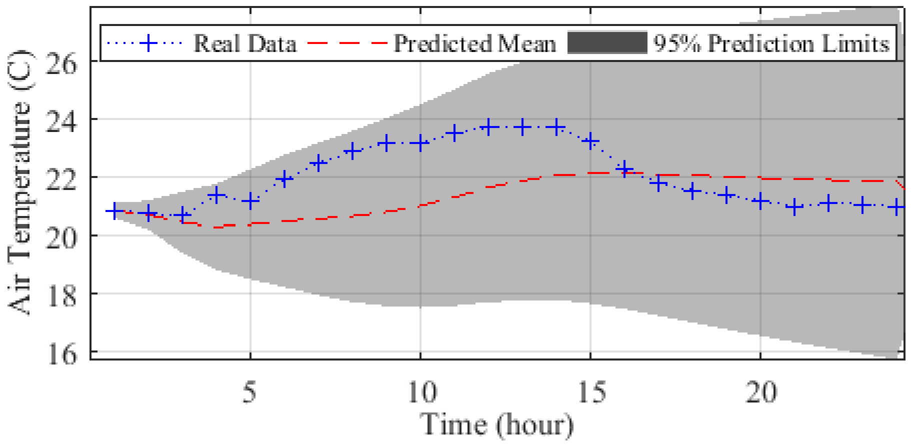


Figure 4.9: Prediction distribution of the West-zone air temperature using a unimodal GP model

The performance statistics for the SHS model compared against the unimodal GP and the full GP models are shown in Table 4.2. These results indicate that the proposed ap-

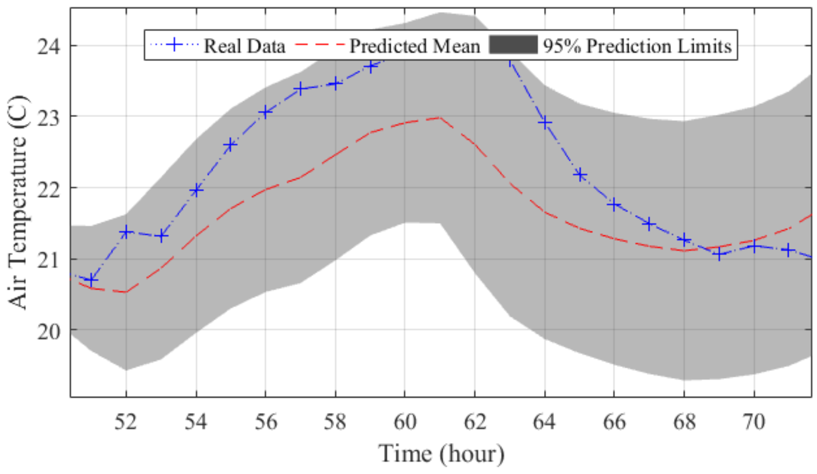


Figure 4.10: Prediction distribution of the West-zone air temperature using the full GP model

Table 4.2: Performance statistics of the reachability analysis prediction with a comparison against the full GP and the unimodal GP models

	RMSE (MRSE)		
	Full GP	**SHS**	**Unimodal GP**
Core Zone	1.10 (0.05)	0.92 (0.04)	2.28 (0.10)
South Zone	1.93 (0.09)	1.09 (0.07)	2.35 (0.10)
East Zone	4.08 (0.18)	1.17 (0.05)	3.82 (0.16)
North Zone	1.46 (0.07)	1.02 (0.04)	3.00 (0.13)
West Zone	1.95 (0.09)	1.28 (0.05)	2.15 (0.09)

proach outperforms both typical GP models since it takes into consideration the prediction of the thermal load level and its periodic patterns. In comparison to the SHS prediction, the unimodal GP lacks the prediction of the thermal load pattern, and therefore, the prediction distribution averages over the thermal load levels with high uncertainty.

Further, we compare the proposed online-learned model against an offline model, which is learned one time offline only, in order to evaluate the improvement of the online learning approach. In this experiment, we consider changes due to seasons as an example to represent the variability in the system and the uncertainty in the occupancy schedule. Figure 4.11 shows the RMSE statistics for both models calculated for each day of the year. The results indicate that online learning allows our model to adapt to the variation in the system with a good performance. On the other hand, the offline model fails to adapt to these variations.

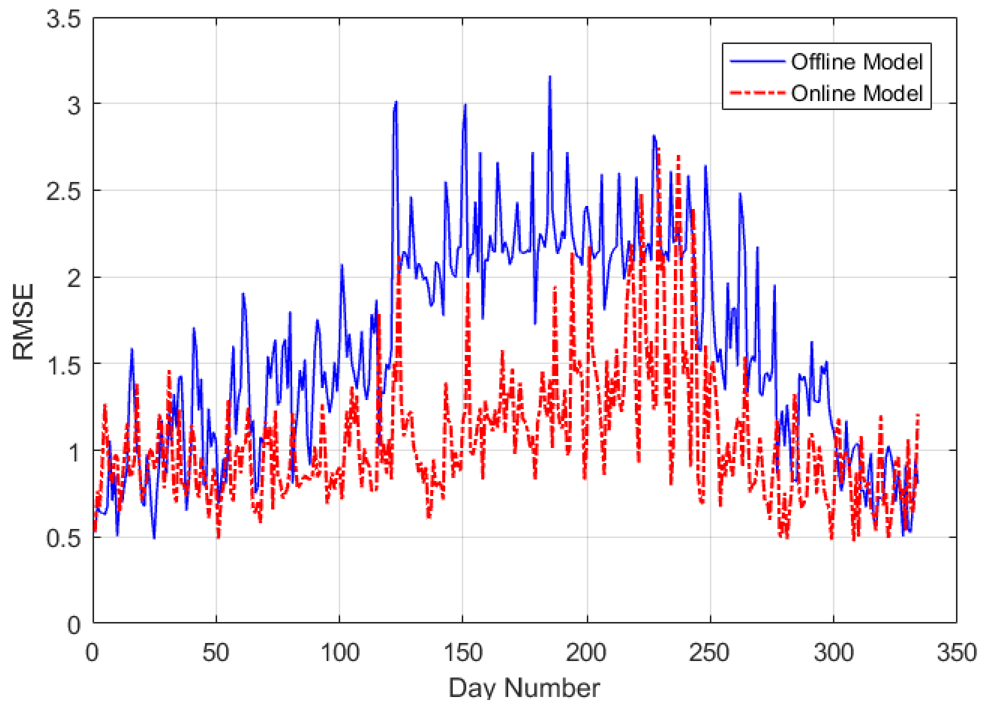


Figure 4.11: Error metric (RMSE) of the air temperature prediction for both the online and the offline learned SHS model, averaged over all zones

4.4.2.2 Efficiency

Despite the computation demand of machine learning algorithms, more specifically GPs, the proposed methodology is computationally efficient and can run in an online fashion with accepted performance. The most expensive part is computing the inverse covariance matrix of GPs which requires $\mathcal{O}(n^3)$ time where n is the size of the training data for each GP model. GP learning becomes more computationally expensive when the dimension of the model and/or the training dataset increases. The computation time because of the GP is evaluated using different experiments of the proposed approach with different sizes of the training dataset (i.e., M). Figure 4.12 shows the variation of the model learning and the reachability analysis average running time for five-zone and two-zone buildings with different dataset sizes. As indicated from these results, the running time becomes a major factor in the GP performance as we increase the training dataset size. However, the performance of the model prediction is still adequate for the relativety small dataset sizes since we segment this data and distrubite the computation for each discrete mode independently.

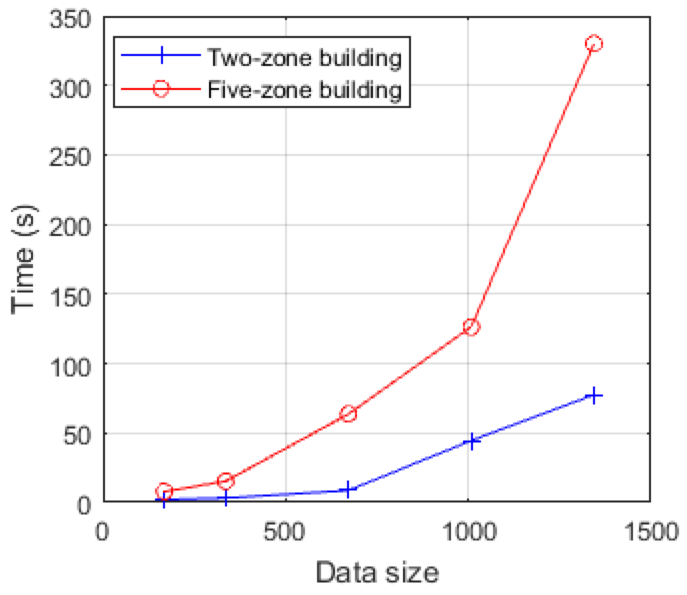


Figure 4.12: Average running time of the proposed framework using different sizes of the training dataset sizes for two buildings examples

Furthermore, we evaluated the proposed methodology performance for different sampling periods of 15, 30, 45, and 60 minutes. For all cases, we fixed the duration of the window size for the training data. However, the dataset sizes increase as the sampling periods decreases. In addition, the prediction horizon is fixed as one day ahead for all sampling periods. Figure 4.13 shows the running times of the model learning and the reachability analysis, respectively, using different sampling periods and training dataset sizes (i.e., 1-week and 2-weeks). The results show an exponential increase of the running time as the sampling period is decreased. The proposed approach runs efficiently with an accepted performance for sampling periods between 30 to 60 minutes. The RMSE, averaged over all zones, performance metrics depicted in Figure 4.14 show that, the reachability analysis performance increases as we increase the sample rate. This is expected since the prediction time-steps increases as we decrease the sampling rate, and therefore, the prediction uncertainty increases.

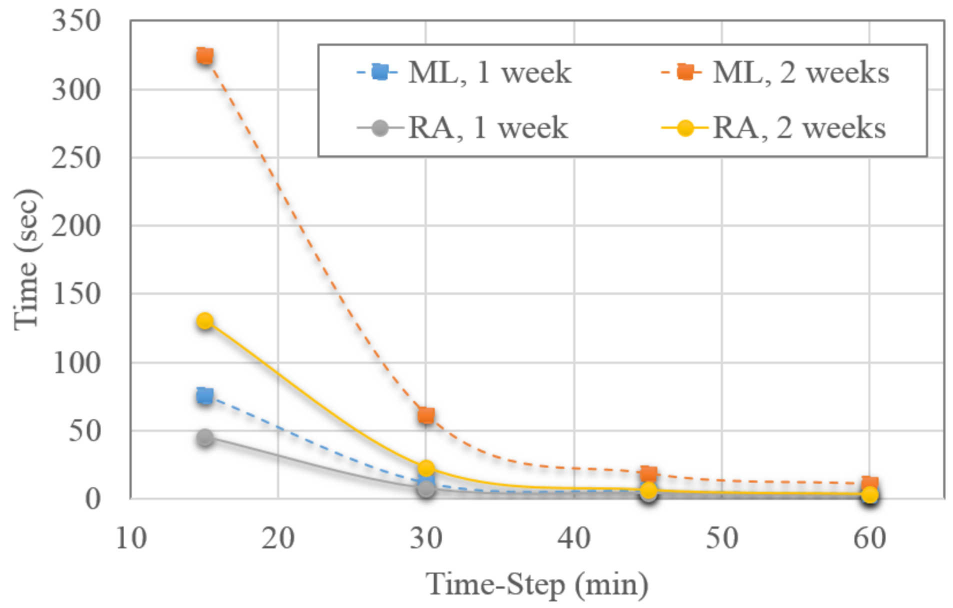


Figure 4.13: Average running times using different sampling periods and training dataset sizes for model learning (ML) and reachability analysis (RA)

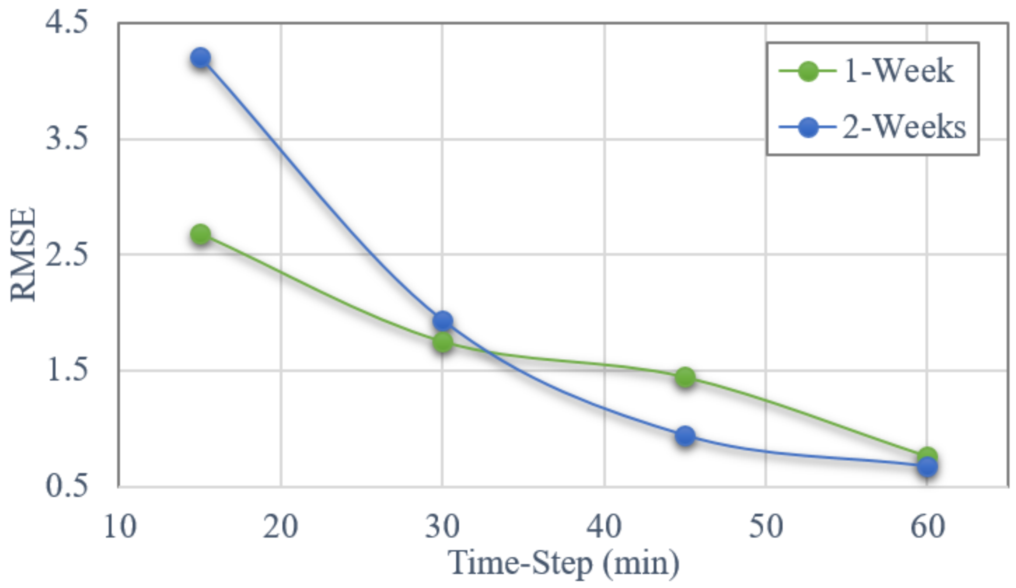


Figure 4.14: The RMSE statistics of the reachability analysis for the five-zones office building using different sampling periods

We conducted the above experiments in intel core i5 PC with 8 GB memory. The results indicate that the proposed approach is applicable for smart buildings applications in real time.

4.5 Reachability Analysis of Stochastic Hybrid Systems with Deterministic Discrete Dynamics

This section illustrates the implementation of the proposed reachability analysis methodology for SHS with deterministic discrete dynamics using a multi-room heating system that has been proposed as a benchmark for SHS in [29]. The multi-room heating system comprises h rooms where each room has its own heater and user setting. Additionally, each room is affected by its adjacent rooms and the ambient temperature. The discrete state represents the heater mode $q_i = \{\texttt{ON}, \texttt{OFF}\}$ for each room and the continuous state represents the room air temperature. The continuous state x_i for each room evolves according to the following stochastic difference equation [4]:

$$x_i(k+1) = x_i(k) + b_i(x_a(k) - x_i(k)) + \sum_{i \neq j} a_{ij}(x_j(k) - x_i(k)) + c_i \mathbb{I}_{Q_i}(q_i(k)) + \omega_i(k) \tag{4.7}$$

where $x_a(k)$ is the ambient temperature at time k, $\mathbb{I}_{Q_i}(\cdot)$ is the indicator function of set $Q_i = \{(q_1, \cdots, q_h) \in \mathscr{Q} : q_i = \texttt{ON}\}$, b_i is non-negative constants representing the average heat transfer rate from room i and the ambient x_a, a_{ij} is non-negative constants representing the average heat transfer rate from room i and room j, c_i is non-negative constants representing the heat rate supplied to room i by the heater, and ω_i is a Gaussian noise disturbance in room i.

The discrete transition function represents the heater operation using a typical controller and each room is controlled independently from the other rooms. The controller switches the heater on if the temperature is below a certain threshold xl, and switches the heater off if the temperature exceeds xu. Formally, the control policy can be described by:

$$\pi(s(k)) = \begin{cases} 0 & if\ q(k) = \texttt{ON}\ \&\ x(k) >= xu \\ 1 & if\ q(k) = \texttt{OFF}\ \&\ x(k) <= xl \end{cases} \tag{4.8}$$

The parameters in the multi-room heating system are hard to model for several reasons. The parameters differ from building to building (e.g., different geometry and materials) and they may change during the system operation either abruptly (e.g., opening window or door) or slowly because of aging. As a result, it is very hard to identify a parametric model of the system.

We consider the ambient temperature as the external uncontrolled input of the system ($v(k) = x_a(k) \in \mathbb{R}$), the controller as the control policy ($u(k) = \pi(s(k)) \in \{0,1\}$), the room temperature vector as the continuous variable ($x(k) \in \mathbb{R}^h$), and the heater state as the discrete state ($\mathscr{Q} = \{q_1, q_2, \cdots, q_h\} = \{\texttt{ON}, \texttt{OFF}\}^h$).

In this experiment, we implemented a system with two rooms (i.e. $h = 2$), and therefore, four discrete states. For each discrete state $q \in \mathscr{Q}$, the continuous state evolves according to

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{x}(k) + \Delta\mathbf{x}(k) \\ \Delta\mathbf{x}(k) &= f_q(\mathbf{x}(k), x_a(k)) \end{aligned} \tag{4.9}$$

where $f_q(.) \sim \mathscr{GP}_q$ is the the continuous dynamics of each mode modeled by a GP. The control input ($\mathbf{u}(k) = \pi(s(k))$) defines the guards for the discrete transitions. The uncontrolled external input is defined as $v(k) = x_a(k) = E(k)$, where $E(k)$ is a time-series model of the ambient temperature. We use the GP introduced in [88] to model $E(k)$ which build a time-series model for the difference between the building ambient and the city weather data.

We have implemented the approach using Matlab and used the model shown in Equation (5.7) to represent the physical system response with the following parameters: $b_1 = 0.4, b_2 = 0.45, a_{12} = a_{21} = 0.5, c_1 = 25, c_2 = 27$, and $\omega_i \sim \mathscr{N}(0,5)$. We generate data with a time-step of one minute and we use data for six hours to learn out model (i.e., $M = 360$). Our goal is to predict the system behavior every hour for the next one hour (i.e., $N = 60$). We train the model using the first six hours simulation data, then apply the online approach to predict the system behavior for the next hour. Next, we collect the data of the system response for the predicted hour and re-learn the model. We repeat the periodic 1-hour predict/learn steps for ten hours. To emulate changes in the system parameters, we increase the heat transfer rate with the ambient (i.e., b_i parameters) in the second hour, such that $b_1 = 0.8$ and $b_2 = 0.6$. In order to evaluate the advantages of online learning and reachability analysis, we also implement the prediction without updating the models online. In another word, we learn the model once offline using the initial training data of six hours only.

To evaluate the prediction accuracy, we use the following weighted mean absolute error (WMAE) as the metric:

$$WMAE = \frac{1}{N}\sum_{k}\left(\frac{1}{C(k)}\sum_{c=1}^{C(k)}(|\mathbf{x}_m(k) - \boldsymbol{\mu}_c(k)| \times w_c(k))\right)$$

where $C(k)$ is the number of Gaussian components at time k, $\mathbf{x}_m(k)$ is the real system measurement at time k, and $\boldsymbol{\mu}_c(k)$ and $w_c(k)$ is the mean and weight of Gaussian component c at time k respectively.

The proposed approach can generate a statistical distribution of the reachable SHS states. Figure 4.15 shows the prediction distribution of the discrete mode, room 1 temperature, and room 2 temperature for the fourth hour of the system operation. To evaluate the improvement, Figure 4.16 shows also the prediction distribution using the model learned offline only. The results also indicate that the online algorithm tracks the system changes when the supplied heat rate increased, such that the algorithm predicts the first discrete transition around $t = 15min$. On the other hand, the offline approach did not adapt and it still predicts the temperature using the higher heating rate, such that the first discrete transition is predicted to occur around $t = 10min$.

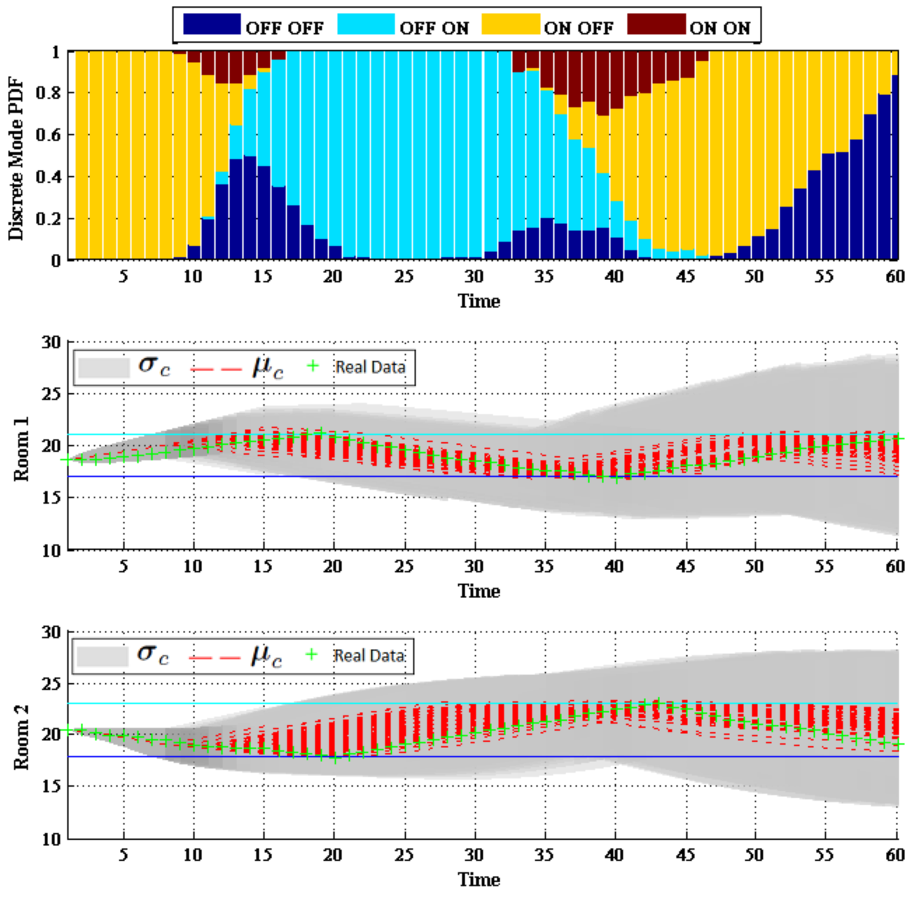


Figure 4.15: Prediction distribution of the hybrid states of the fourth hour using online learned model

The WMAE error for both online and offline models is shown in Figure 4.17 for the ten hours of system operation. The results show that the error is decreasing as the model aggregates more data. The average execution time of the learning and the reachability analysis algorithms are 7.1 and 4.7 sec; respectively. Figure 4.18 illustrates the accuracy of the proposed reachability analysis algorithm by comparing its results with sampled trajectories generated by Monte Carlo simulation from the parametric model in Equation (5.7).

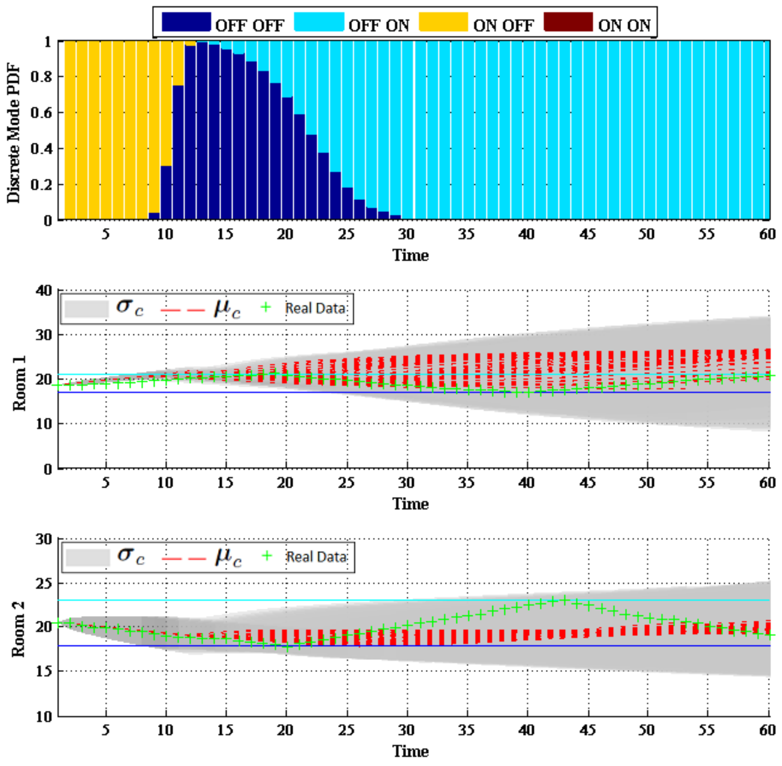


Figure 4.16: Prediction distribution of the hybrid state of the fourth hour using offline learned model

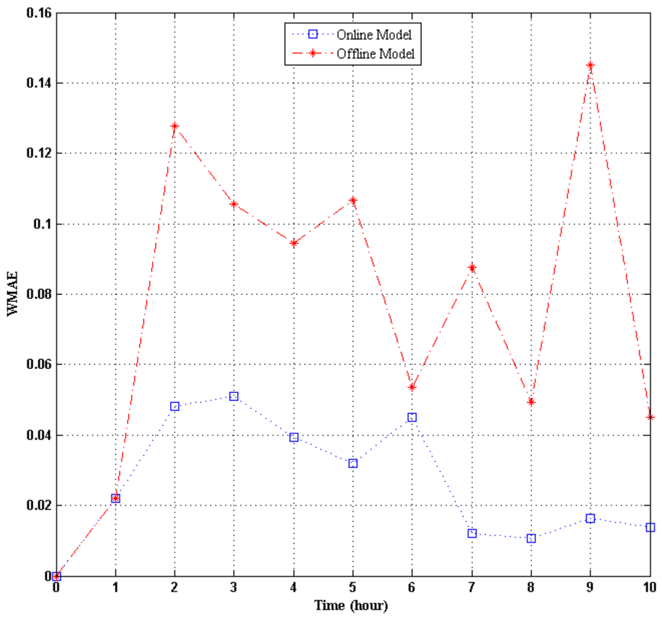


Figure 4.17: WMAE error of the 10-hours prediction for both the online and the offline learned model

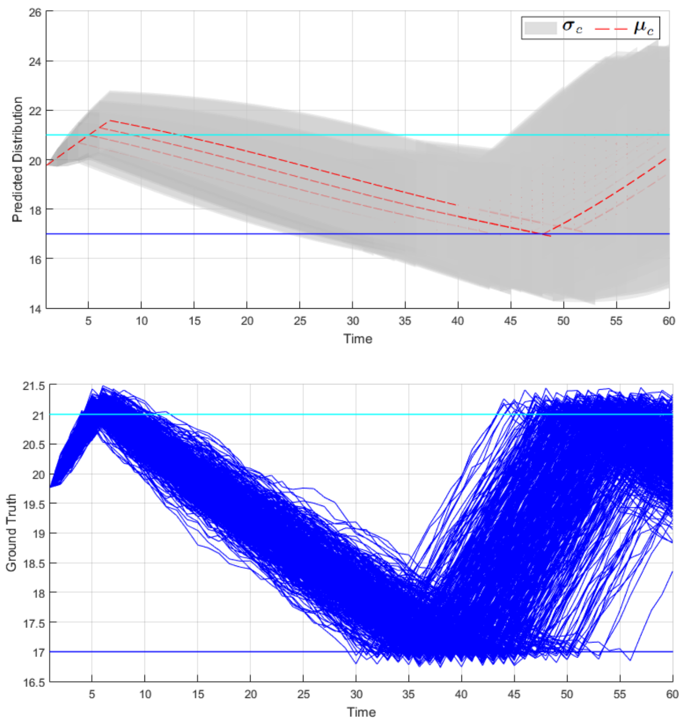


Figure 4.18: Prediction distribution of the reachable states vs ground truth

4.6 Conclusions

In this chapter, we present an online data-driven approach for learning and reachability analysis of SHS model. The novel reachability analysis approach infers a statistical distribution of the reachable state for a finite-horizon using mixtures of Gaussian processes. Further, it runs in online fashion to adapt to time-variability behavior in the system. The presented approach can be applied to many modern CPS with a multimodal behavior when a parametric model is hard to obtain. As a practical example, we evaluate the efficiency of the approach on smart buildings application. The experiment results indicate that our approach runs efficiently in an online fashion and provides a statistical distribution of the reachable state with a good performance and accuracy.

Chapter 5

Decision Making and Control of Stochastic Hybrid Systems

While precise modeling and analysis techniques play a major role in improving model-based design of CPS, the primary element which drives these systems autonomously to their desired behaviors is the control algorithm. Hence, design control methodologies and decision-making strategies for such systems are very important. However, developing such control methodologies and decision-making strategies for SHS is very challenging due to the nonlinearity and the stochasticity in the coupled continuous/discrete dynamics. In this chapter, we illustrate the application of a control approach known as stochastic model predictive control (SMPC) using a nonparametric SHS model. The proposed approach is based on approximating the SMPC problem by a deterministic scenario-based MPC. The scenario-based approximation is obtained by enumerating the uncertainty in the discrete dynamics which yields to a deterministic nonlinear optimization problem. Gradient-based nonlinear programming methods are then used to solve the optimization problem. We evaluate the performance and the efficiency of the proposed approach in smart buildings applications.

This chapter is organized as follows: Section 5.1 discusses the problem of SMPC for SHS with its challenges. Section 5.2 presents the proposed scenario-based MPC approximation of the SMPC problem, the approximated optimization problem, and its gradient analytic evaluation. Finally, Section 5.3 presents the implementation and the evaluation of the proposed method using buildings applications.

5.1 Stochastic Model Predictive Control of Stochastic Hybrid Systems

Stochastic model predictive control (SMPC) is based on optimizing a desired system variable (typically, the system state) for a receding horizon subject to constraints on the control inputs and/or the system states. The controlled system has stochastic nonlinear dynamics and can be described by a stochastic model. In this chapter, we consider dynamical systems with coupled discrete and continuous stochastic dynamics which are modeled using nonparametric SHS as defined in Section 4.1. Thus, the predictive distribution of the continuous state trajectory (i.e., the system state which we want to control) is described using a mixture of Gaussian processes:

$$p(\mathbf{x}_{k+1}|\hat{\mathbf{x}}_k) = \sum_i p(q_{k+1} = i|\mathbf{x}_k, q_k)\mathscr{GP}_i(m_i(\hat{\mathbf{x}}_k), k_i(\hat{\mathbf{x}}_k, \hat{\mathbf{x}}_k)$$

where $\hat{\mathbf{x}}_k := [\mathbf{x}_k^T \mathbf{u}_k^T v_k^T]^T$ is a tuple of the continuous state $\mathbf{x}_k$, the control input $\mathbf{u}_k$ and the uncontrolled input v_k, and $p(q_{k+1} = i|\mathbf{x}_k, q_k)$ represents the probabilistic discrete transition using a periodic Markov chain as described in the previous chapter.

The SMPC problem for SHS aims to calculate the control inputs that minimize the expectation of a given cost function h over the predicted continuous state trajectory and the control inputs for a receding finite horizon N. Formally, the SMPC problem can be described as the following optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{u}_{0:N-1}} \mathbb{E}[h(\boldsymbol{x}_{1:N}, \boldsymbol{u}_{0:N-1})] \\ &\text{subject to} \\ &\boldsymbol{u}_{0:N-1} \in \mathscr{U} \\ &p(\mathbf{x}_{k+1}|\hat{\mathbf{x}}_k) = \sum_i p(q_{k+1} = i|\mathbf{x}_k, q_k)\mathscr{GP}_i(m_i(\hat{\mathbf{x}}_k), k_i(\hat{\mathbf{x}}_k, \hat{\mathbf{x}}_k) \end{aligned} \tag{5.1}$$

where $h(.)$ is a cost function defined over the system state and the control inputs, and $\mathscr{U}$ defines the feasible regions of the control input.

The SMPC problem defined in Equation (5.1) should be solved at every time step. The optimized control input at the current step time only (i.e., $\mathbf{u}_0$) is applied to the physical system. Then, the initial state (i.e., $\mathbf{x}_0$) is updated after receiving new measurements from the physical system. Solving the above SMPC problem for SHS is very challenging because of the stochasticity in both the continuous and the discrete dynamics in addition to the nonlinearity of the continuous dynamics (represented by Gaussian processes). These challenges yield to a complex nonlinear stochastic optimization problem. Thus, an approximation solution is needed to mitigate these challenges. In the next section, we illustrate a scenario-based MPC approximation to solve the SMPC problem for nonparametric SHS.

5.2 Scenario-Based Model Predictive Control of Stochastic Hybrid Systems

The key idea behind scenario-based MPC is to enumerate deterministic realizations, known as scenarios, of the uncertainty in the system dynamics. These scenarios approximate the SMPC problem by a deterministic MPC. Thus, scenario-based MPC can be used to optimize the control input over these scenarios while ensuring that the constraints are satisfied for all of them. In our case, there are two sources of stochasticity in SHS. First, the stochastic behavior exhibited in the continuous dynamics which is modeled by Gaussian processes. Second, the stochastic behavior exhibited in the discrete dynamics which is modeled by a periodic Markov chain model. We can calculate analytically the uncertainty in the continuous state since it is represented by Gaussian processes. On the other hand, the estimation of the discrete state cannot be calculated analytically, and therefore, we need to enumerate the uncertainty of the discrete state.

We propose a scenario-based MPC approach for non-parametric SHS. A scenario represents a predicted hybrid state with a probability of reaching this scenario from the measured initial state of the controlled system. These scenarios construct an optimization tree with its root node as the current measured state of the controlled system and the tree height as the MPC receding finite horizon N. Formally, let's define the scenario tree as a set of nodes $\mathscr{T} = \{\mathscr{S}_0, \mathscr{S}_1, \cdots \mathscr{S}_m\}$ such that each node (or scenario) of the tree consists of the predicted discrete state q_i, the probability of reaching this discrete state from the initial state π_i, its associated control input u_i (optimization variable), the predicted uncontrolled input v_i and the predictive distribution (i.e., mean and variance) of the continuous state expressed using its corresponding Gaussian process $\mathscr{GP}_{q_i}$. We construct an optimization tree for the control horizon to include the scenarios with probabilities higher than a given threshold while ignoring scenarios with very small probabilities (i.e., below a certain threshold). The optimization tree is constructed as shown in Algorithm 4.

Algorithm 4 Design the Scenario-Based Optimization Tree

```
Input: $x_0$,$q_0$, $N$,$th_q$
Output: $\mathscr{T}$
  ▷ Initialize the root node using the current system state:
  $\mathscr{S}_0 \leftarrow [x_0, q_0, \pi_0 = 1, u_0]$
  $\mathscr{T} \leftarrow \mathscr{T} \cup \{\mathscr{S}_0\}$
  $\mathscr{T}_{par} \leftarrow \{\mathscr{S}_0\}$                              ▷ a temporary parent set
  $m \leftarrow 1, k \leftarrow 1$
  while $k \leq N$ do
      $v_k \leftarrow E(k)$                                ▷ Forecast the external input $v_k$
      ▷ Create the child nodes for each scenario in the parent set:
      for each $S_p \in \mathscr{T}_{par}$ do
          ▷ Calculate the discrete probabilities of each discrete state from $S_p$:
          $p(q) \leftarrow \delta(S_p.q, k-1)$
          for each $q \in \mathscr{Q}$ do
              ▷ Calculate the probability of the new scenario
              $pi_{new} \leftarrow p(q_k = q) \times S_p.pi$
              if $pi_{new} > th_q$ then
                  ▷ Add $\mathscr{S}_m$ to the optimization tree:
                  $\mathscr{S}_m \leftarrow [x_m \sim \mathscr{GP}_q, q_m = q, \pi_m = pi_{new}, u_m, v_k]$
                  $\mathscr{T} \leftarrow \mathscr{T} \cup \{\mathscr{S}_m\}$
                  ▷ and the next time-step parent set:
                  $\mathscr{T}_{par2} \leftarrow \mathscr{T}_{par2} \cup \{\mathscr{S}_m\}$
                  $m \leftarrow m+1$
              end if
          end for
      end for
      $\mathscr{T}_{par} \leftarrow \mathscr{T}_{par2}$                            ▷ Update the parent set
      $\mathscr{T}_{par2} \leftarrow \{\}$
      $k \leftarrow k+1$
  end while
```

The developed scenario-based MPC approximates the stochastic optimization problem defined in Equation (5.1) by a deterministic optimization problem. For simplicity, we design the controller objective function as a quadratic function over the continuous state and the control inputs[1]. In the SHS model, the system state is expressed by a Gaussian distribution (i.e., $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}_x, \Sigma_x)$), hence, the estimation of quadratic objective function over the state and the control inputs can be calculated analytically as follows:

$$\begin{aligned}
&\mathbb{E}[h(\boldsymbol{x}_{1:N}, \boldsymbol{u}_{0:N-1}))] = \\
&\sum_k \mathbb{E}[(\boldsymbol{x}_k - r_k)' Q_x (\boldsymbol{x}_k - r_k) + \boldsymbol{u}'_{k-1} Q_u \boldsymbol{u}_{k-1})] \\
&= \sum_k (\boldsymbol{\mu}_{x_i} - r_i)' Q_x (\boldsymbol{\mu}_{x_i} - r_i) + \mathrm{Tr}\{Q_x \Sigma_{x_k}\} \\
&\qquad + \boldsymbol{u}'_{k-1} Q_u \boldsymbol{u}_{k-1})
\end{aligned} \tag{5.2}$$

where r_k is the desired state trajectory at time step k. The SMPC defined in Equation (5.1) for a quadratic objective function can be reformulated based on the scenario-based optimization tree as:

$$\begin{aligned}
&\min_{\mathbf{u}} \sum_{i \in \mathcal{T} \setminus \{\mathcal{S}_0\}} \pi_i ((\boldsymbol{\mu}_{x_i} - r_i)' Q_x (\boldsymbol{\mu}_{x_i} - r_i) \\
&\qquad + \mathrm{Tr}\{Q_x \Sigma_{x_i}\} + \mathbf{u}'_{par(i)} Q_u \mathbf{u}_{par(i)}) \\
&\text{subject to} \\
&\mathbf{u}_{par(i)} \in \mathcal{U} \quad , \quad \forall i \in \mathcal{T} \setminus \{\mathcal{S}_0\} \\
&(\boldsymbol{\mu}_{x_i}, \Sigma_{x_i}) \sim \mathcal{GP}_{q_i}(m_{q_i}(\hat{\mathbf{x}}_i), k_{q_i}(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_i))
\end{aligned} \tag{5.3}$$

where $par(i)$ indicates the index of the parent node of node i and $\hat{\mathbf{x}}_i := [\mathbf{x}'_{par(i)} \mathbf{u}'_{par(i)} v'_i]'$. Problem (5.3) is a deterministic nonlinear constrained optimization problem which can be solved using nonlinear programming methods such as interior-point algorithms [18] [99].

Typically, nonlinear programming methods use numerical gradients calculated by finite-difference approximation. Numerical approximations are computationally demanding and are not very accurate. Alternatively, analytical expressions of the gradients can be used, if possible, to solve the optimization problem more accurately and efficiently.

[1]This is an arbitrary choice but the proposed approach can accept other types of objective functions if the gradients and statistical estimation can be obtained analytically.

5.2.1 Gradient Calculation

The gradient of the objective function in problem (5.3) can be obtained as follows:

$$\frac{d\mathbb{E}[h(.)]}{d\boldsymbol{u}} = \sum_{i \in \mathscr{T} \setminus \{\mathscr{S}_0\}} \pi_i \frac{d\mathscr{E}_i}{d\boldsymbol{u}} \tag{5.4}$$

with $\mathscr{E}_i = (\boldsymbol{\mu}_{x_i} - r_i)' Q_x (\boldsymbol{\mu}_{x_i} - r_i) + \mathrm{Tr}\{Q_x \boldsymbol{\Sigma}_{x_i}\} + \mathbf{u}'_{par(i)} Q_u \mathbf{u}_{par(i)}$.

Therefore, the gradient of each scenario $\frac{d\mathscr{E}_i}{d\boldsymbol{u}}$ can be derived using the chain rule:

$$\frac{d\mathscr{E}_i}{d\boldsymbol{u}} = \frac{\partial \mathscr{E}_i}{\partial \boldsymbol{\mu}_{x_i}} \frac{d\boldsymbol{\mu}_{x_i}}{d\boldsymbol{u}} + \frac{\partial \mathscr{E}_i}{\partial \boldsymbol{\Sigma}_{x_i}} \frac{d\boldsymbol{\Sigma}_{x_i}}{d\boldsymbol{u}} + \frac{\partial \mathscr{E}_i}{\partial \boldsymbol{u}} \tag{5.5}$$

where

$$\begin{aligned} \frac{d\boldsymbol{\mu}_{x_i}}{d\boldsymbol{u}} &= \frac{\partial \boldsymbol{\mu}_{x_i}}{\partial \boldsymbol{\mu}_{x_{par(i)}}} \frac{\partial \boldsymbol{\mu}_{x_{par(i)}}}{\partial \boldsymbol{u}} + \frac{\partial \boldsymbol{\mu}_{x_i}}{\partial \boldsymbol{u}} \\ \frac{d\boldsymbol{\Sigma}_{x_i}}{d\boldsymbol{u}} &= \frac{\partial \boldsymbol{\Sigma}_{x_i}}{\partial \boldsymbol{\Sigma}_{x_{par(i)}}} \frac{\partial \boldsymbol{\Sigma}_{x_{par(i)}}}{\partial \boldsymbol{u}} + \frac{\partial \boldsymbol{\Sigma}_{x_i}}{\partial \boldsymbol{u}} \end{aligned} \tag{5.6}$$

The partial derivatives $\frac{\partial \mathscr{E}_i}{\partial \boldsymbol{\mu}_{x_i}}, \frac{\partial \mathscr{E}_i}{\partial \boldsymbol{\Sigma}_{x_i}}$ and $\frac{\partial \mathscr{E}_i}{\partial \boldsymbol{u}}$ depend on the choice of the objective function. For the quadratic objective function, these derivatives can be calculated as:

$$\frac{\partial \mathscr{E}_i}{\partial \boldsymbol{\mu}_{x_i}} = 2(\boldsymbol{\mu}_{x_i} - r_i)' Q_x, \ \ \frac{\partial \mathscr{E}_i}{\partial \boldsymbol{\Sigma}_{x_i}} = Q'_x, \ \ \frac{\partial \mathscr{E}_i}{\partial \boldsymbol{u}} = 2\boldsymbol{u}' Q_u$$

The derivatives $\frac{\partial \boldsymbol{\mu}_{x_i}}{\partial \boldsymbol{\mu}_{x_{par(i)}}}, \frac{\partial \boldsymbol{\mu}_{x_i}}{\partial \boldsymbol{u}}, \frac{\partial \boldsymbol{\Sigma}_{x_i}}{\partial \boldsymbol{\Sigma}_{x_{par(i)}}}$ and $\frac{\partial \boldsymbol{\Sigma}_{x_i}}{\partial \boldsymbol{u}}$ depend on the GP kernel and mean functions[2]. They also depend on the approximation function used to calculate the GP predictive distribution at uncertain inputs. We use an approximation method known as exact moment matching to propagate the uncertainty of the continuous state for each scenario and the uncertainty in the foretasted uncontrolled input v_i. Details about prediction using the moment matching approximation method along with its derivatives can be found in [25].

With the analytic expression of the gradient, the scenario-based MPC is solved at every

[2]The derivatives $\frac{\partial \boldsymbol{\mu}_{x_{par(i)}}}{\partial \boldsymbol{u}}$ and $\frac{\partial \boldsymbol{\Sigma}_{x_{par(i)}}}{\partial \boldsymbol{u}}$ are known from the previous gradients calculation of the parent scenario

time-step to calculate the optimal control inputs for all possible scenarios efficiently. Only the optimal control input of the root scenario (i.e., $\mathbf{u}_0$) is applied to the physical system. Then, the initial scenario (i.e., $\mathscr{S}_0$) is updated again with the new measurement from the physical system and a new scenario-based optimization tree is constructed.

Algorithm 5 summarizes the major steps of SMPC for SHS with scenario-based approximation and online learning.

Algorithm 5 SMPC for SHS with online learning

```
while TRUE do
    Get measurements at time $t$;
    Update the SHS model;
    Update the forecast model $E(t)$;
    Design the Scenario-Based Optimization Tree by Algorithm 4;
    Solve the optimization problem defined in 5.3;
    Apply $u_0$ ;
end while
```

5.3 Evaluation

In this section, we illustrate the implementation of the proposed scenario-based MPC method to control an HVAC system in smart buildings.

5.3.1 Two-Zones Building

In this section, we consider controlling a heating system in buildings where the discrete state represents the buildings activities level (e.g., open/close windows by occupancy). The discrete state (the activity level) affects the heat transfer rates between the buildings zones internally and with the ambient externally. For instance, in the high activities mode, these heat transfer rates are increased due to the frequent building activities such as open/close windows.

Simulation Setup

We represent the physical system using a parametric model based on a multi-zone heating system benchmark [29]. The multi-zone heating system comprises h rooms/zones where each zone has its own heater and user setting. Additionally, each zone is affected by its adjacent zones and the ambient temperature. The continuous state x_i for each zone evolves according to the following stochastic difference equation [4]:

$$x_i(k+1) = x_i(k) + b_i^q(x_a(k) - x_i(k)) + \sum_{i \neq j} a_{ij}^q(x_j(k) - x_i(k)) + c_i u_i(k)) + \omega_i^q(k) \tag{5.7}$$

where $x_a(k)$ is the ambient temperature at time k, b_i^q is non-negative constants representing the average heat transfer rate from zone i and the ambient in the discrete mode q, a_{ij}^q is non-negative constants representing the average heat transfer rate from zone i and zone j in the discrete mode q, u_i is the control input that represents the supplied heating rate to zone i, c_i is non-negative coefficient of the supplied heat to zone i and ω_i^q is a Gaussian noise disturbance in zone i. The building model in (5.7) is used only to represent the physical system behavior and to generate the training data.

We implement this system using MATLAB® to simulate the physical system response with the following parameters: $b_1^1 = 0.0375, b_1^2 = 0.06, b_2^1 = 0.025, b_2^2 = 0.05, a_{12}^1 = a_{21}^1 = 0.0625, a_{12}^2 = a_{21}^2 = 0.09, c_1 = 0.0163, c_2 = 0.015, \omega_i^1 \sim \mathcal{N}(0, 0.15)$, and $\omega_i^2 \sim \mathcal{N}(0, 0.25)$. Further, we generate 295 data samples with 1-hour time-step to learn a SHS of the above system as follows. In this model, we consider the zone temperature vector as the continuous state ($\mathbf{x}(k) \in \mathbb{R}^h$) and the building activities levels as the discrete state $\mathcal{Q} = \{q_1, q_2, \cdots q_m\}$ with m modes, the heat supplied rate as the control input $\mathbf{u}(k) \in \mathbb{R}^h$ and the ambient temperature as the external uncontrolled input ($v(k) = x_a(k) \in \mathbb{R}$). In this experiment, we consider a building case study with two zones and two discrete states (low and high building activities levels). We also model the discrete dynamics using a periodic Markov chain with transition probabilities that reflect commercial buildings environments (i.e., the building has high probability to be in high mode between 8 am to 5 pm). The uncontrolled external input is defined as $v(k) = x_a(k) = E(k)$, where $E(k)$ is a time-series model of the ambient temperature. We also use a time-series GP model to represent $E(k)$.

The experiment goal is to optimize the control inputs for each hour using the proposed SMPC method, in order to achieve user comfort while optimizing energy consumption. The SMPC controller is designed with the following parameters: receding horizon $N = 3$, objective weight matrices $Q_x = \left(\begin{smallmatrix} 100 & 0 \\ 0 & 100 \end{smallmatrix}\right)$ and $Q_u = \left(\begin{smallmatrix} 0.01 & 0 \\ 0 & 0.01 \end{smallmatrix}\right)$, and $th_q = 0.01$. The desired state trajectory is to maintain both zones temperature at 16 (C) when the building in the Low activity level and at 18.5 (C), and 19.5 (C) for zone x_1 and x_2; respectively when the building in the high activity level. The bounding constraint of the control inputs is $-10 \leq u_i \leq 150$.

Performance Evaluation

We run the simulation for three days, collect data, and evaluate the performance of the proposed approach (i.e., SMPC controller) by comparing against two typical MPC controller strategies: **Reactive-based MPC**, and **Scheduled-based MPC**.

Reactive-based MPC optimizes the system objective function based on the current discrete-mode only. In other words, this controller optimizes the objective function based on the GP model of the current discrete-mode only. In this case, the MPC control problem is a typical GP MPC problem defined as follows:

$$\begin{aligned} &\min_{\boldsymbol{u}_{0:N-1}} \mathbb{E}[h(\boldsymbol{x}_{1:N}, \boldsymbol{u}_{0:N-1})] \\ \text{subject to} \quad & \\ &\boldsymbol{u}_{0:N-1} \in \mathscr{U} \\ &p(\mathbf{x}_{k+1}|\hat{\mathbf{x}}_k) = \mathscr{GP}_{q_0}(m_i(\hat{\mathbf{x}}_k), k_i(\hat{\mathbf{x}}_k, \hat{\mathbf{x}}_k)) \end{aligned} \tag{5.8}$$

Scheduled-based MPC is a common MPC control strategy where it predicts the system behavior based on a fixed schedule of the discrete modes (e.g., occupancy) [61]. The discrete mode schedule is set be in high mode during the typical business hours (i.e., between 8 am to 5 pm). In this case, the MPC problem ignores the uncertainty in the discrete dynamics and defines a typical GP MPC problem with deterministic discrete dynamics:

$$\begin{aligned} &\min_{\boldsymbol{u}_{0:N-1}} \mathbb{E}[h(\boldsymbol{x}_{1:N}, \boldsymbol{u}_{0:N-1})] \\ &\text{subject to} \\ &\boldsymbol{u}_{0:N-1} \in \mathscr{U} \\ &p(\mathbf{x}_{k+1}|\hat{\mathbf{x}}_k) = \mathscr{G}\mathscr{P}_{q_{h(k+1)}}(m_i(\hat{\mathbf{x}}_k), k_i(\hat{\mathbf{x}}_k, \hat{\mathbf{x}}_k) \end{aligned} \tag{5.9}$$

where q_h is the scheduled discrete mode of the hour of the day h.

Simulation Results

The simulation results for the zone air temperatures and the HVAC heating/cooling rate using the reactive-based MPC, scheduled-based MPC and the proposed scenario-based SMPC are shown in Figure 5.1, Figure 5.2 and Figure 5.3 respectively. These results indicate that the reactive-based MPC manage to maintain the temperature to the desired level as soon as the system stays within one discrete mode. However, it takes time to react to the systems changes when the system switches its discrete mode. The scheduled-based MPC controls the temperature based on a predefined schedule (8 am to 5 pm) as high activity mode regardless the current measurement of the discrete mode. On the other hand, the proposed scenario-based SMPC controls the temperature based on the probability of the discrete mode given the current measurement. This provides a predictive behavior which takes into consideration the current state and the probabilities of the next time steps.

We also calculated the actual cost values for all control strategies. The total cost reflects both the cost due to energy usage and the deviation from the desired system trajectory. The scenario-based SMPC approach results on the lowest total cost (33782). As comparing to

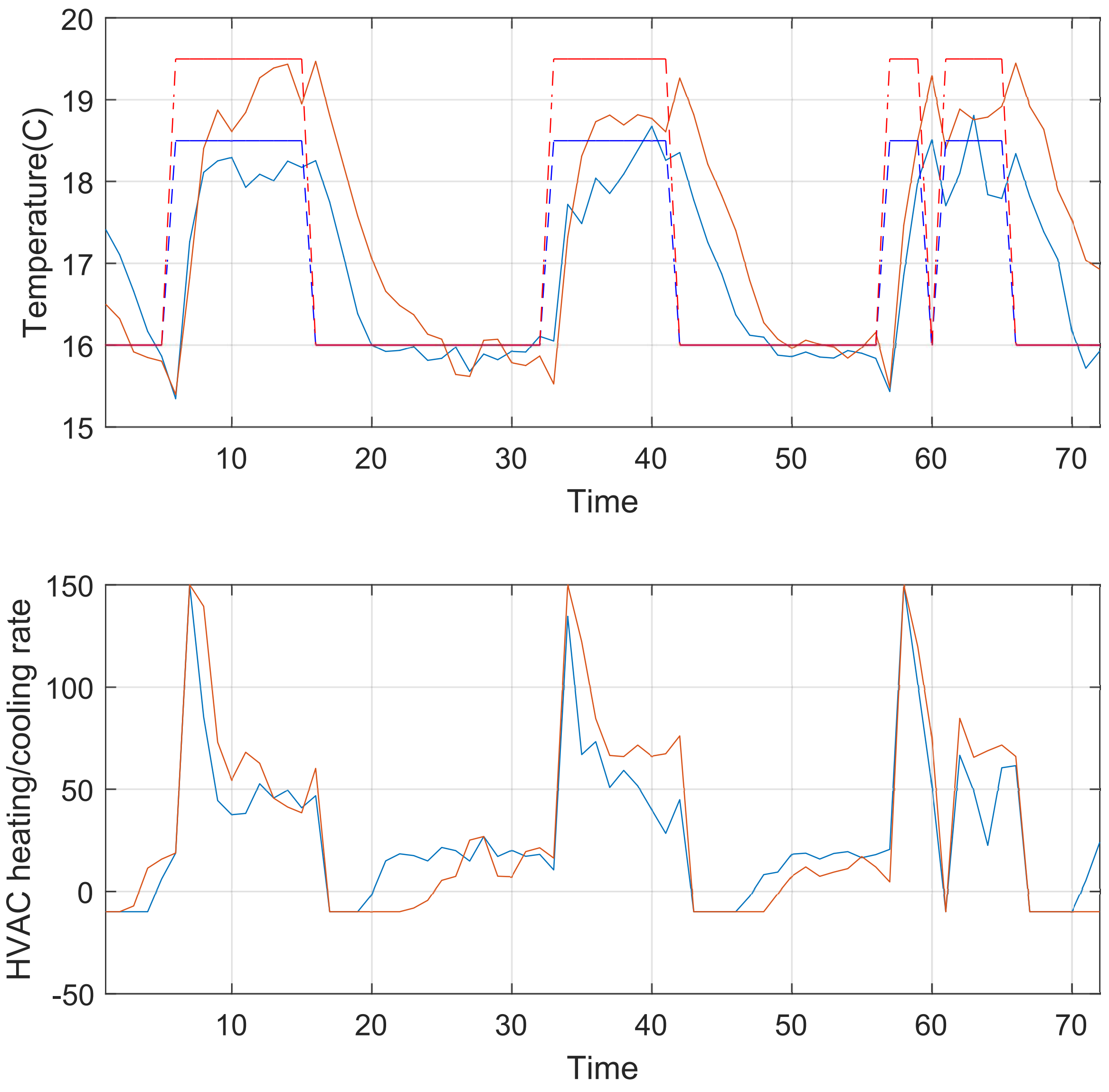
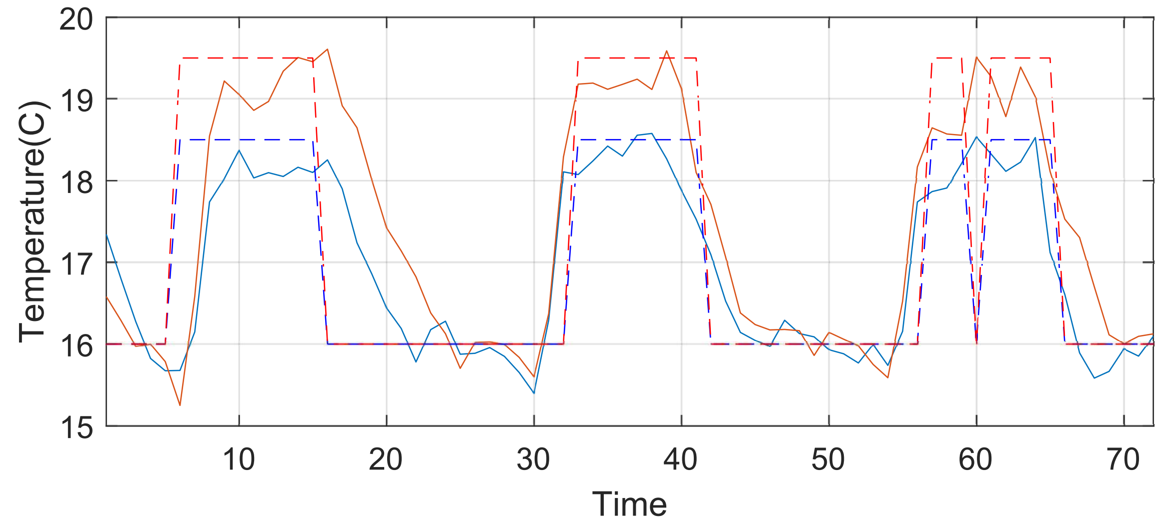


Figure 5.1: Simulation results for both zones temperatures and heater rate using reactive-based MPC. The dotted lines represent the desired trajectories for each zone



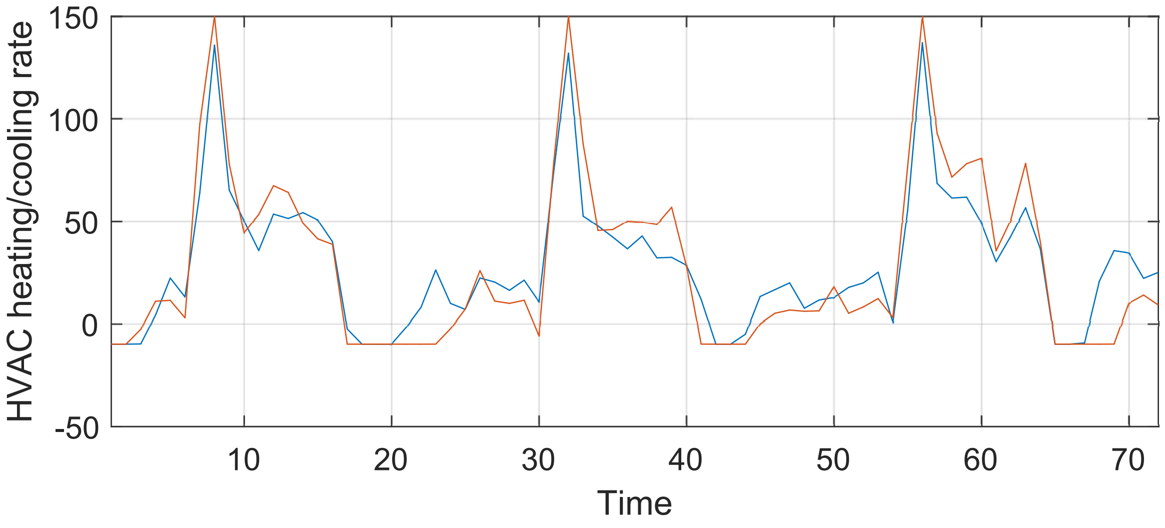


Figure 5.2: Simulation results for both zones temperatures and heater rate using scheduled-based MPC. The dotted lines represent the desired trajectories for each zone

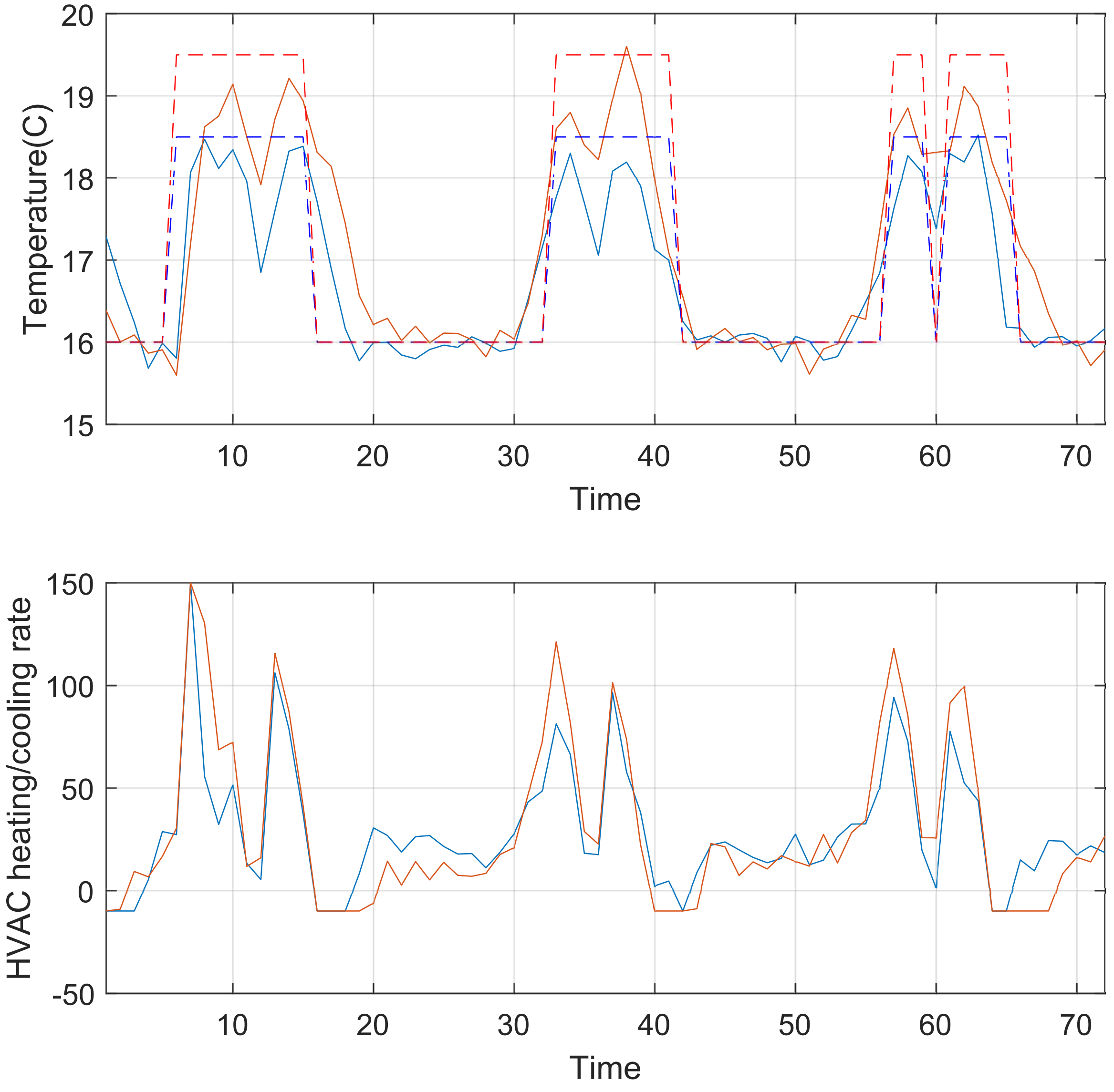


Figure 5.3: Simulation results for both zones temperatures and heater rate using the proposed SMPC. The dotted lines represent the desired trajectories for each zone

the scheduled-based MPC and the reactive-based MPC which result on total cost: (47572) and (45746); respectively. The proposed SMPC approach adapts to the variation in the discrete dynamics. On the other hand, scheduled-based MPC uses a fixed schedule which might be different than the actual behavior. This leads to degradation to the control performance. For instance, we run a scheduled-based MPC example with a scheduled set to be from 7 am to 4 pm. In this case, the total cost increased to 46040. The simulation results for this example is depicted in Figure 5.4.

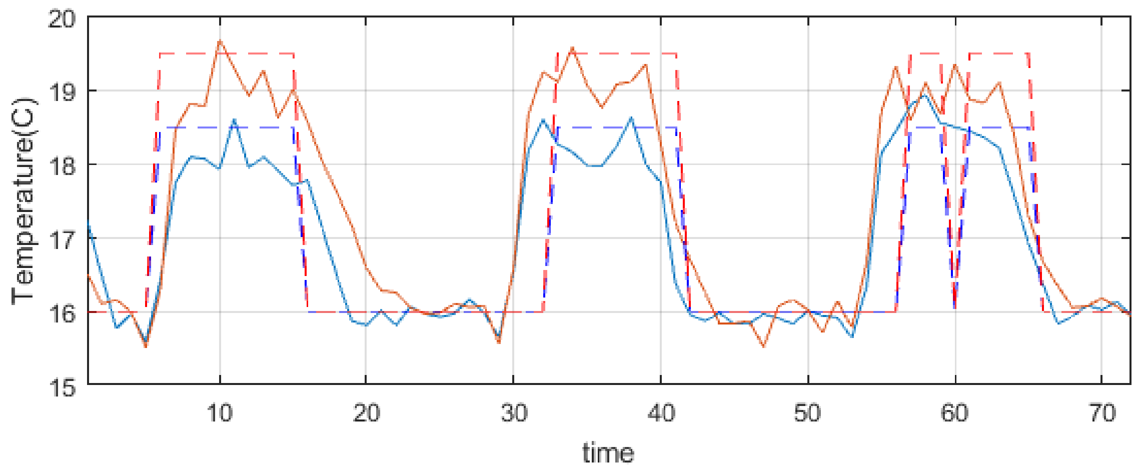


Figure 5.4: Simulation results for both zones temperatures using scheduled-based MPC (the high discrete mode is set from 7 am to 4 pm). The dotted lines represent the desired trajectories for each zone

5.3.2 Five-Zones Building

In this section, we evaluate the control approach for an office building with five zones. We simulate the office building response based on a stochastic occupant schedule for office spaces. The occupancy schedule is generated using the occupancy simulator developed at Lawrence Berkeley National Laboratory [30] [1]. We use the occupancy schedule, weather data of San Francisco, and a parametric model to simulate the physical system response. Our objective is to control this building using the proposed approach. Further, we assume that the sensor measurements do not include occupancy. Therefore, we learn a SHS model of this system with a latent discrete state as discussed in Chapter 3 and 4. The discrete state represents the thermal load level due to occupancy.

Simulation Setup

We represent the physical system using the occupancy schedule, weather data of San Francisco, and the following multi-zone parametric model:

$$\begin{aligned} x_i(k+1) = x_i(k) + & b_i(x_a(k) - x_i(k)) \\ & + \sum_{i \neq j} a_{ij}(x_j(k) - x_i(k)) \\ & + c_i u_i(k)) + d_i q(k) + \omega_i^q(k) \end{aligned} \tag{5.10}$$

where $x_i(k)$ is the i^{th} zone air temperature of zone at time k, $x_a(k)$ is the ambient temperature at time k, b_i is non-negative constant representing the average heat transfer rate from zone i and the ambient, a_{ij} is non-negative constant representing the average heat transfer rate between zone i and zone j, u_i is the control input that represents the supplied heating/-cooling rate to zone i, c_i is non-negative coefficient of the supplied heat to zone i, q_i is the number of occupant in zone i at time k, c_i is non-negative coefficient of the occupancy input for zone i,and ω_i is a Gaussian noise disturbance for zone i. The building model in (5.10) is used to generate the training data and to represent the physical system response only. We implement this system using MATLAB® to simulate the physical system response with the following parameters: $b^i = 0.01$, $a_{ij} = 0.00625$ for $i = 1$ and 0.005 otherwise, $c_i = 0.07$ for $i = 1$ and 0.065 otherwise, $d_i = 0.05$, and $\omega_i \sim \mathcal{N}(0, 0.05)$.

Further, we initially use a typical PID controller to generate a training dataset of three days with 30-minute time-step. The PID controller is developed with the following parameters ($k_p = 2, k_i = 1, k_d = 1$) and we used it to maintain the zones temperature at fixed setting (e.g., 17 (C)) in order to generate initial dataset of the system. This training dataset is then used to learn a SHS model that we use in the proposed SMPC control approach. In this model, we consider the zone temperature vector as the continuous state ($\mathbf{x}(k) \in \mathbb{R}^5$) and the thermal load levels as the discrete state $\mathscr{Q} = \{q_1, q_2, \cdots q_m\}$ with m modes, the heat supplied rate as the control input $\mathbf{u}(k) \in \mathbb{R}^5$ and the ambient temperature as the external uncontrolled input ($v(k) = x_a(k) \in \mathbb{R}$). Moreover, we learn a time-series model for the ambient temperature $v(k) = x_a(k) = E(k)$ using a GP. The occupancy schedule is generated based on a typical office-building occupancy patterns (i.e., the estimated arriving time is 08:00 am ± 60 min and the estimated leaving time is 05:00 pm ± 60 min). Also, the occupancy behavior is simulated for four office rooms and one conference room.

The objective is to optimize the control inputs for each time step using the proposed SMPC method. The SMPC controller is designed with the following parameters: receding horizon $N = 3$, objective weight matrices $Q_x = 100\mathbb{I}_5$ and $Q_u = 0.01\mathbb{I}_5$, and $th_q = 0.01$. The desired state trajectory is to maintain the temperature of the zones at 18.5 (C) between 08:00 am and 05:00 pm, 17 (C) during the lunch break (i.e., noon), and 16 (C) otherwise. The bounding constraint of the control inputs is $-10 \leq u_i \leq 100$.

Performance Evaluation

We run the experiment for one day of simulation and evaluate the performance of the proposed approach (i.e., SMPC controller) by comparing against two typical MPC controller strategies: **Reactive-based MPC** and **Scheduled-based MPC** as defined in (5.8) and (5.9); respectively. The simulation results for one zone using the reactive-based MPC, scheduled-based MPC and the proposed scenario-based SMPC are shown in Figure 5.5, Figure 5.6 and Figure 5.7; respectively. The results for the other zones have similar performance to the depicted one.

The results indicate that the proposed approach provides more accurate control than the reactive-based MPC approach. The scheduled-based MPC has the worst performance since its model is based on a fixed schedule rather than estimation of the thermal load level from the data. The controller performance is low at the beginning of the simulation but

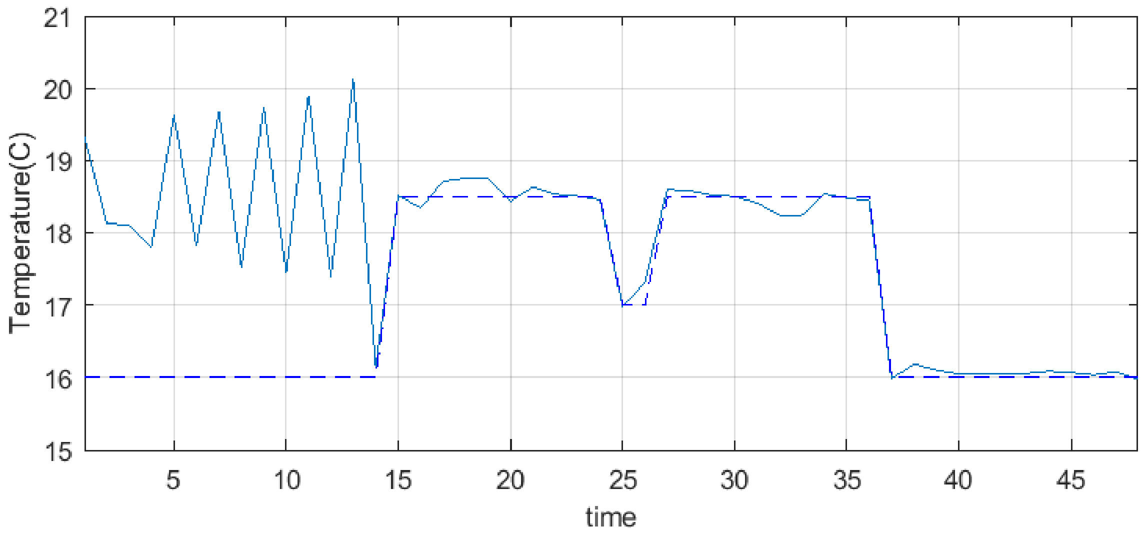


Figure 5.5: Simulation results of zone temperatures of one zone using reactive-based MPC. The dotted lines represent the desired trajectories for each zone

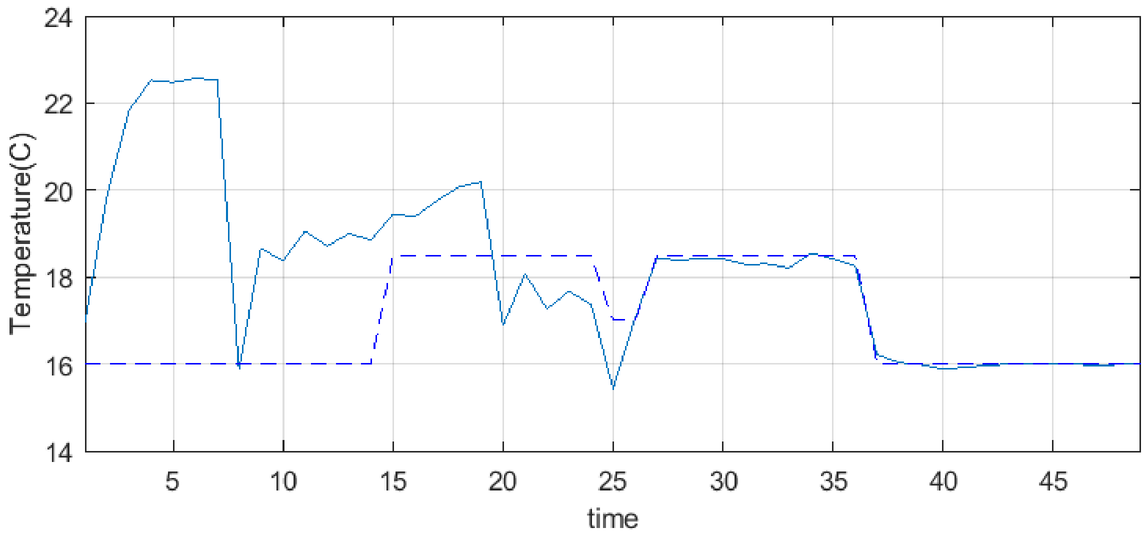


Figure 5.6: Simulation results of zone temperatures of one zone using scheduled-based MPC. The dotted lines represent the desired trajectories for each zone

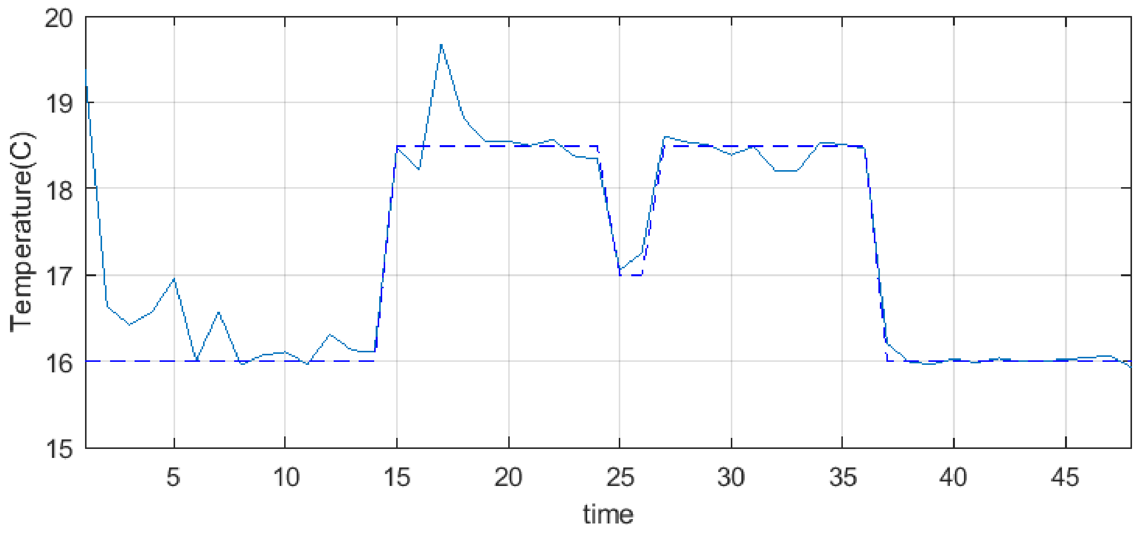


Figure 5.7: Simulation results of zone temperatures of one zone using the proposed SMPC. The dotted lines represent the desired trajectories for each zone

improves as the SHS model learns the system dynamics. We also calculate the actual total cost for all controllers based on the objective function. The total cost reflects both the cost due to energy usage and the deviation from the desired system trajectory for all zones. The scenario-based SMPC approach results on the lowest total cost (12e+03). As comparing to the scheduled-based MPC and the reactive-based MPC which result on total cost: (28e+03) and (26e+03); respectively. The scheduled-based MPC is based on a fixed schedule. However, in reality, the occupancy behavior is dynamic and can change over time. Thus, this variation leads to degradation of the scheduled-based MPC performance. For instance, we run another scheduled-based MPC experiment with an occupancy scheduled set to be from 7 am to 4 pm instead of 8 am to 4 pm. The cost of the new schedule is (106e+03) as compared to (28e+03) of the 8 am to 5 pm schedule case. The simulation results for this example are shown in Figure 5.8.

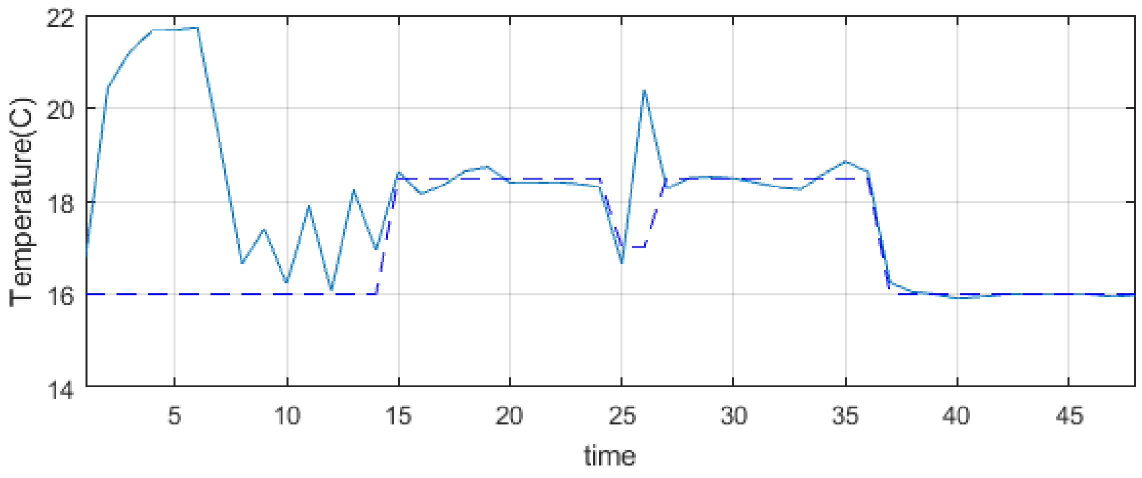


Figure 5.8: Simulation results of zone temperatures of one zone using scheduled-based MPC (the occupancy mode is scheduled from 7 am to 4 pm). The dotted lines represent the desired trajectories for each zone

Computational Efficiency

To study the efficiency of the proposed approach, we run several experiments with different training dataset sizes (i.e., number of simulated days) and with different prediction horizon of the control process. The running times for both the online learning step and the control optimization steps are shown in Figure 5.9 for different training dataset sizes. Figure 5.10 shows the running times of the control optimization process as we increase its prediction horizon. These results show that the running time increases exponentially as we increase the training dataset size and it increases linearly as we increase the prediction horizon. However, the proposed method runs relatively efficient with small and medium training dataset sizes with acceptable performance.

5.4 Conclusions

In this chapter, we present a scenario-based SMPC approach of non-parametric SHS which can be used to model complex systems with coupled stochastic continuous/discrete dynamics. We describe a scenario-based approximation of SMPC to provide a robust MPC.

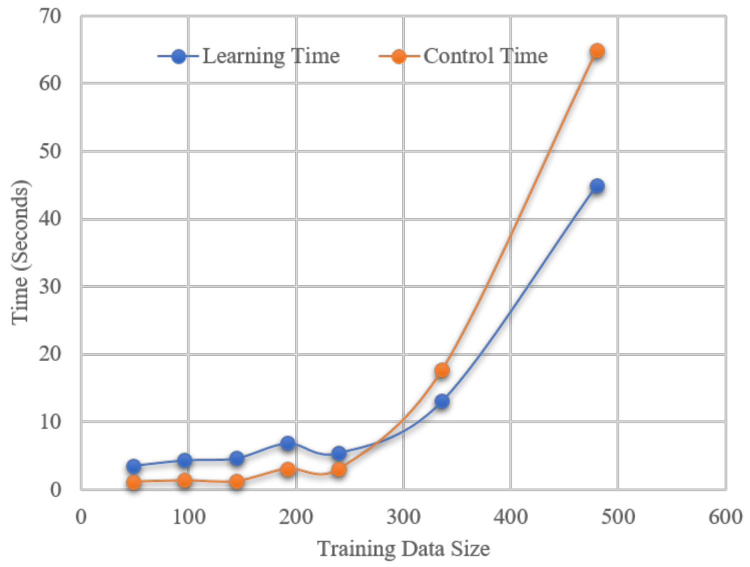


Figure 5.9: Running time of the model learning step and the control step using different training dataset sizes

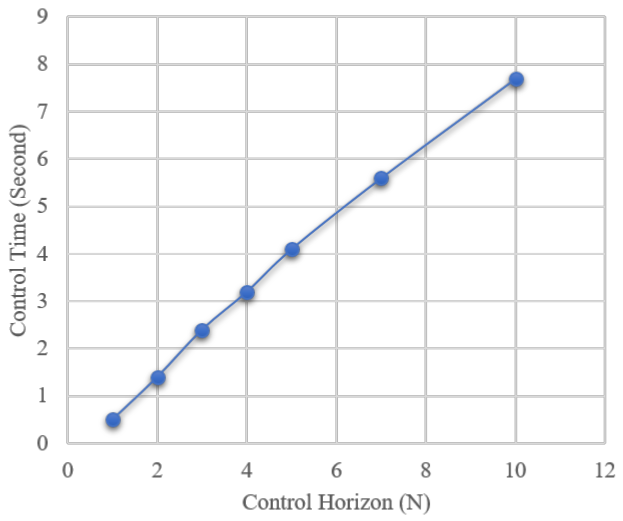


Figure 5.10: Running times of the control approach using different prediction horizon N

Moreover, we illustrate the analytic calculation of the objective function gradients to enhance the accuracy and the efficiency of the optimization routines. The approach is used for a smart building control application. The simulation results show the capability of this approach to control the efficiently while taking into consideration the uncertainty in the system behavior.

Chapter 6

Conclusions

Modern CPS impose complex behaviors with uncertainty. Model-based design of such complex CPS arises many great opportunities and research challenges. This dissertation provides a data-driven model-based design approach to mitigate various research challenges in model learning, reachability analysis and control for CPS modeled by nonparametric SHS. An online clustering-based model learning approach is illustrated to learn nonparametric SHS when the discrete dynamics is latent (i.e., the discrete state can not be measured explicitly). This modeling approach can be used in many modern CPS when a parametric model is hard to obtain. Further, online learning is useful to adapt the system models to time-variability behavior and to improve the model efficiency because the training data in the offline phase is not necessarily complete. Experimental results demonstrate the efficiency of the model learning approach, and, show that the learning process runs online with an adequate computation time. A novel reachability analysis approach is proposed to infer a statistical distribution of the reachable state for a finite-horizon using mixtures of Gaussian processes. The reachability analysis approach provides an efficient multi-step prediction of the reachable continuous and discrete states to provide predictive analysis for CPS modeled by nonparametric SHS. Moreover, a scenario-based SMPC approach is developed to provide a robust MPC for CPS modeled by nonparametric SHS. The gradients for the MPC optimization problem is estimated analytically to enhance the accuracy and the efficiency of the optimization routines. Experimental results show the capability of this approach to control the underline system to the desired behaviors efficiently while taking into consideration the uncertainty in the system behavior. Empirical results indicates that the developed approaches are applicable for many CPS systems. As a practical example, the efficacy of the developed approaches for smart buildings applications is shown using high-fidelity building simulator.

This dissertation provides a foundation of developing data-driven approaches for CPS modeled by SHS. Therefore, various research opportunities can be considered for future work. In particular, this dissertation provides a cluster-based learning approach to learn SHS with periodic and latent discrete state. Cluster-based approaches assume the data that lie close to each other to probably belong to the same discrete state. Thus, other learning approaches such as Bayesian procedure approaches should be studied to support other types of SHS where the clustering-based approaches are not applicable. We also recommend future work to extended the SMPC illustrated in chapter 5 to support SMPC with stochastic nonlinear constraints. In this case, the main challenge arises in approximating the optimization problem to avoid local minimum. These future work will facilitate the efficacy of data-driven SHS approaches for wide range of CPS applications.

LIST OF PUBLICATIONS

Journal Papers

1. Hamzah Abdel-Aziz and Xenofon Koutsoukos. ”Online Model Learning of Buildings Using Stochastic Hybrid Systems Based on Gaussian Processes”, *Journal of Control Science and Engineering*, Volume 2017, Article ID 3035892, 18 pages, August 2017.

2. Shekhar, Shashank, Hamzah Abdel-Aziz, Michael Walker, Faruk Caglar, Aniruddha Gokhale, and Xenofon Koutsoukos. ”A simulation as a service cloud middleware.” *Annals of Telecommunications* 71, no. 3-4 (2016): 93-108.

Conference Papers

1. Hamzah Abdel-Aziz, Xenofon Koutsoukos, ”Scenario-Based Model Predictive Control of Stochastic Hybrid Systems based on Gaussian Processes”, Submitted for publication.

2. Shashank Shekhar, Hamzah Abdel-Aziz, Anirban Bhattacharjee, Aniruddha Gokhale and Xenofon Koutsoukos, ”Performance-Aware Vertical Elasticity for Latency-Sensitive Applications”, Submitted for publication.

3. Hamzah Abdel-Aziz, Xenofon Koutsoukos, ”Data-Driven Online Learning and Reachability Analysis of Stochastic Hybrid Systems for Smart Buildings” Submitted for publication.

4. Hamzah Abdel-Aziz, Xenofon Koutsoukos, ”Learning and Reachability Analysis for Stochastic Hybrid Systems using Mixtures of Gaussian Processes” *In Control and Automation (MED), 2016* 24th *Mediterranean Conference on*, pp. 332-337. IEEE, Greece 2016.

Workshop Papers

1. Hamzah Abdel-Aziz, Faruk Caglar, Shashank Shekhar, Michael Walker, Xenofon Koutsoukos, Aniruddha Gokhale, ”Online Performance Model Learning to Minimize Performance Interference in Cloud Computing Infrastructure”, *IEEE* 22nd *International Conference on High Performance Computing Workshops (HiPCW)*, pp. 62, 2015.

BIBLIOGRAPHY

[1] Occupancy simulator. http://occupancysimulator.lbl.gov/.

[2] The annual energy outlook 2015 with projection to 2040. *US Energy Information Administration, Tech. Rep*, 2015.

[3] Alessandro Abate, Joost-Pieter Katoen, John Lygeros, and Maria Prandini. Approximate model checking of stochastic hybrid systems. *European Journal of Control*, 16(6):624–641, 2010.

[4] Alessandro Abate, Maria Prandini, John Lygeros, and Shankar Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11):2724–2734, 2008.

[5] Ali Abusnina and Daniel Kudenko. Adaptive soft sensor based on moving gaussian process window. In *Industrial Technology (ICIT), 2013 IEEE International Conference on*, pages 1051–1056. IEEE, 2013.

[6] Anayo K Akametalu and Claire J Tomlin. Temporal-difference learning for online reachability analysis. In *Control Conference (ECC), 2015 European*, pages 2508–2513. IEEE, 2015.

[7] Saurabh Amin, Alessandro Abate, Maria Prandini, John Lygeros, and Shankar Sastry. Reachability analysis for controlled discrete time stochastic hybrid systems. In *Hybrid Systems: Computation and Control*, pages 49–63. Springer, 2006.

[8] Anil Aswani, Neal Master, Jay Taneja, Valton Smith, Andrew Krioukov, David Culler, and Claire Tomlin. Identifying models of hvac systems using semiparametric regression. In *American Control Conference (ACC), 2012*, pages 3675–3680. IEEE, 2012.

[9] J Andrew Bagnell and Jeff G Sc Hneider. Autonomous helicopter control using reinforcement learning policy search methods. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1615–1620. IEEE, 2001.

[10] Radhakisan Baheti and Helen Gill. Cyber-physical systems. *The impact of control technology*, 12:161–166, 2011.

[11] Daniele Bernardini and Alberto Bemporad. Scenario-based model predictive control of stochastic constrained linear systems. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 6333–6338. IEEE, 2009.

[12] Christopher M Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[13] Lars Blackmore, Stephanie Gil, Seung Chung, and Brian Williams. Model learning for switching linear systems with autonomous mode transitions. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2007.

[14] Lars Blackmore, Masahiro Ono, Askar Bektassov, and Brian C Williams. A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *IEEE transactions on Robotics*, 26(3):502–517, 2010.

[15] Luminita Manuela Bujorianu. *Stochastic Reachability Analysis of Hybrid Systems*. Communications and Control Engineering. Springer London. DOI: 10.1007/978-1-4471-2795-6_12.

[16] Manuela L Bujorianu. Extended stochastic hybrid systems and their reachability problem. In *Hybrid Systems: Computation and Control*, pages 234–249. Springer, 2004.

[17] Manuela L Bujorianu and John Lygeros. General stochastic hybrid systems: Modelling and optimal control. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 2, pages 1872–1877. IEEE, 2004.

[18] Richard H Byrd, Jean Charles Gilbert, and Jorge Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89(1):149–185, 2000.

[19] Joaquin Quioñero Candela, Agathe Girard, Jan Larsen, and Carl Edward Rasmussen. Propagation of uncertainty in bayesian kernel models-application to multiple-step ahead forecasting. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 2, pages II–701. IEEE, 2003.

[20] Gang Cao, Edmund MK Lai, and Fakhrul Alam. Gaussian process model predictive control of unmanned quadrotors. In *Control, Automation and Robotics (ICCAR), 2016 2nd International Conference on*, pages 200–206. IEEE, 2016.

[21] Alongkrit Chutinan and Bruce H. Krogh. Verification of polyhedral-invariant hybrid automata using polygonal flow pipe approximations. In Frits W. Vaandrager and Jan H. van Schuppen, editors, *Hybrid Systems: Computation and Control*, number 1569 in Lecture Notes in Computer Science, pages 76–90. Springer Berlin Heidelberg. DOI: 10.1007/3-540-48983-5_10.

[22] Drury B Crawley, Linda K Lawrie, Frederick C Winkelmann, Walter F Buhl, Y Joe Huang, Curtis O Pedersen, Richard K Strand, Richard J Liesen, Daniel E Fisher, Michael J Witte, et al. Energyplus: creating a new-generation building energy simulation program. *Energy and buildings*, 33(4):319–331, 2001.

[23] Thao Dang and Oded Maler. Reachability analysis via face lifting. In *Hybrid Systems: Computation and Control*, pages 96–109. Springer, 1998.

[24] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[25] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423, 2015.

[26] Marc Peter Deisenroth and Jun Wei Ng. Distributed gaussian processes. In *International Conference on Machine Learning (ICML)*, volume 2, page 5, 2015.

[27] M.P. Deisenroth, D. Fox, and C.E. Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(2):408–423, Feb 2015.

[28] Stefano Di Cairano, Daniele Bernardini, Alberto Bemporad, and Ilya V Kolmanovsky. Stochastic mpc with learning for driver-predictive vehicle control and its application to hev energy management. *IEEE Transactions on Control Systems Technology*, 22(3):1018–1031, 2014.

[29] Ansgar Fehnker and Franjo Ivančić. Benchmarks for hybrid systems verification. In *Hybrid Systems: Computation and Control*, pages 326–341. Springer, 2004.

[30] Xiaohang Feng, Da Yan, and Tianzhen Hong. Simulation of occupancy in buildings. *Energy and Buildings*, 87:348–359, 2015.

[31] Roger Frigola, Fredrik Lindsten, Thomas B Schön, and Carl Edward Rasmussen. Bayesian inference and learning in gaussian process state-space models with particle mcmc. In *Advances in Neural Information Processing Systems*, pages 3156–3164, 2013.

[32] Roman Garnett, Michael A Osborne, and Stephen J Roberts. Sequential bayesian prediction in the presence of changepoints. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 345–352. ACM, 2009.

[33] Denny Gert. Stochastic model predictive control. 2014.

[34] N Haji Ghassemi and M Deisenroth. Analytic long-term forecasting with periodic gaussian processes. In *Proc. of AISTATS*, pages 303–311, 2014.

[35] Siddhartha Ghosh, Steve Reece, Alex Rogers, Stephen Roberts, Areej Malibari, and Nicholas R Jennings. Modeling the thermal dynamics of buildings: A latent-force-model-based approach. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(1):7, 2015.

[36] Agathe Girard and Roderick Murray-Smith. Gaussian processes: Prediction at a noisy input and application to iterative multiple-step ahead forecasting of time-series. In *Switching and Learning in Feedback Systems*, pages 158–184. Springer, 2005.

[37] Agathe Girard, Carl Edward Rasmussen, Joaquin Quinonero-Candela, and Roderick Murray-Smith. Gaussian process priors with uncertain inputs? application to multiple-step ahead time series forecasting. 2003.

[38] Sofie Haesaert, Paul MJ Van den Hof, and Alessandro Abate. Data-driven and model-based verification via bayesian identification and reachability analysis. *Automatica*, 79:115–126, 2017.

[39] He Hao, Timothy Middelkoop, Prabir Barooah, and Sean Meyn. How demand response from commercial buildings will provide the regulation needs of the grid. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pages 1908–1913. IEEE, 2012.

[40] Jason Hardy, Frank Havlak, and Mark Campbell. Multi-step prediction of nonlinear gaussian process dynamics models with adaptive gaussian mixtures. *The International Journal of Robotics Research*, page 0278364915584007, 2015.

[41] Jianghai Hu, John Lygeros, and Shankar Sastry. Towards a theory of stochastic hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*, pages 160–173. Springer, 2000.

[42] Qie Hu, Frauke Oldewurtel, Maximilian Balandat, Evangelos Vrettos, Datong Zhou, and Claire J Tomlin. Building model identification during regular operation-empirical results and challenges. *arXiv preprint arXiv:1603.06872*, 2016.

[43] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[44] Leonard Kaufman and Peter J Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, volume 344. John Wiley & Sons, 2009.

[45] Katarina Kavsek-Biasizzo, Igor Skrjanc, and Drago Matko. Fuzzy predictive control of highly nonlinear ph process. *Computers & chemical engineering*, 21:S613–S618, 1997.

[46] Juš Kocijan. Gaussian process models for systems identification. In *Proc. 9th Int. PhD Workshop on Sys. and Cont*, pages 8–15, 2008.

[47] Juš Kocijan. *Modelling and Control of Dynamic Systems Using Gaussian Process Models*. Springer, 2016.

[48] Juš Kocijan, Roderick Murray-Smith, Carl Edward Rasmussen, and Agathe Girard. Gaussian process model based predictive control. In *American Control Conference, 2004. Proceedings of the 2004*, volume 3, pages 2214–2219. IEEE, 2004.

[49] Konstantinos Koutroumpas, Eugenio Cinquemani, Panagiotis Kouretas, and John Lygeros. Parameter identification for stochastic hybrid systems using randomized optimization: A case study on subtilin production by bacillus subtilis. *Nonlinear Analysis: Hybrid Systems*, 2(3):786–802, 2008.

[50] Jaroslav Krystul and Henk AP Blom. Sequential monte carlo simulation of rare event probability in stochastic hybrid systems. In *Proceedings of the 16th IFAC World Congress*, pages 4–8. Citeseer, 2005.

[51] Andras Gabor Kupcsik, Marc Peter Deisenroth, Jan Peters, and Gerhard Neumann. Data-efficient generalization of robot skills with contextual policy search. In *AAAI*, 2013.

[52] Alexander B Kurzhanski and Pravin Varaiya. Ellipsoidal techniques for reachability analysis: internal approximation. *Systems & control letters*, 41(3):201–211, 2000.

[53] Harold Joseph Kushner. *Probability methods for approximations in stochastic control and for elliptic equations*, volume 129. Academic Press, 1977.

[54] Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems*, pages 1071–1079, 2014.

[55] Bojan Likar and Juš Kocijan. Predictive control of a gas–liquid separation plant based on a gaussian process model. *Computers & chemical engineering*, 31(3):142–152, 2007.

[56] Rudolf Lioutikov, Alexandros Paraschos, Jochen Peters, and Gerhard Neumann. Sample-based informationl-theoretic stochastic optimal control. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3896–3902. IEEE, 2014.

[57] Yudong Ma, Francesco Borrelli, Brandon Hencey, Brian Coffey, Sorin Bengea, and Philip Haves. Model predictive control for the operation of building cooling systems. *Control Systems Technology, IEEE Transactions on*, 20(3):796–803, 2012.

[58] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[59] MATLAB and Statistics Toolbox Release 2016a, 2016.

[60] MATLAB and Statistics Toolbox Release 2017a, 2017.

[61] Amin Mirakhorli and Bing Dong. Occupancy behavior based model predictive control for building indoor climatea critical review. *Energy and Buildings*, 129:499–513, 2016.

[62] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *IEEE Transactions on Automatic Control*, 50(7):947–957, July 2005.

[63] Ian M Mitchell, Alexandre M Bayen, and Claire J Tomlin. A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games. *Automatic Control, IEEE Transactions on*, 50(7):947–957, 2005.

[64] Hiroyuki Mori and Masatarou Ohmi. Probabilistic short-term load forecasting with gaussian processes. In *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, pages 6–pp. IEEE, 2005.

[65] Roderick Murray-Smith, Daniel Sbarbaro, Carl Edward Rasmussen, and Agathe Girard. Adaptive, cautious, predictive control with gaussian process priors. 2003.

[66] Jun Wei Ng and Marc Peter Deisenroth. Hierarchical mixture-of-experts model for large-scale gaussian process regression. *arXiv preprint arXiv:1412.3078*, 2014.

[67] Duy Nguyen-Tuong, Jan R Peters, and Matthias Seeger. Local gaussian process regression for real time online model learning. In *Advances in Neural Information Processing Systems*, pages 1193–1200, 2009.

[68] Duy Nguyen-Tuong, Matthias Seeger, and Jan Peters. Real-time local gp model learning. In *From Motor Learning to Interaction Learning in Robots*, pages 193–207. Springer, 2010.

[69] Wangdong Ni, Soon Keat Tan, and Wun Jern Ng. Recursive gpr for nonlinear dynamic process modeling. *Chemical engineering journal*, 173(2):636–643, 2011.

[70] Wangdong Ni, Soon Keat Tan, Wun Jern Ng, and Steven D Brown. Moving-window gpr for nonlinear dynamic system modeling with dual updating and dual preprocessing. *Industrial & Engineering Chemistry Research*, 51(18):6416–6428, 2012.

[71] Frauke Oldewurtel, Alessandra Parisio, Colin Jones, Manfred Morari, Dimitrios Gyalistras, Markus Gwerder, Vanessa Stauch, Beat Lehmann, and Katharina Wirth. Energy efficient building climate control using stochastic model predictive control and weather predictions. In *Proceedings of the 2010 American control conference*, number EPFL-CONF-169733, pages 5100–5105. Ieee Service Center, 445 Hoes Lane, Po Box 1331, Piscataway, Nj 08855-1331 Usa, 2010.

[72] Michael A Osborne, Stephen J Roberts, Alex Rogers, Sarvapali D Ramchurn, and Nicholas R Jennings. Towards real-time information processing of sensor network data using computationally efficient multi-output gaussian processes. In *Proceedings of the 7th international conference on Information processing in sensor networks*, pages 109–120. IEEE Computer Society, 2008.

[73] Yunpeng Pan and Jun Wang. Model predictive control of unknown nonlinear dynamical systems based on recurrent neural networks. *IEEE Transactions on Industrial Electronics*, 59(8):3089–3101, 2012.

[74] Simone Paoletti, Aleksandar Lj Juloski, Giancarlo Ferrari-Trecate, and René Vidal. Identification of hybrid systems a tutorial. *European journal of control*, 13(2):242–260, 2007.

[75] Alessandra Parisio, Luca Fabietti, Marco Molinari, Damiano Varagnolo, and Karl H Johansson. Control of hvac systems via scenario-based explicit mpc. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 5201–5207. IEEE, 2014.

[76] Gianluigi Pillonetto. A new kernel-based approach to hybrid system identification. *Automatica*, 70:21–31, 2016.

[77] Maria Prandini and Jianghai Hu. A stochastic approximation method for reachability computations. In *Stochastic Hybrid Systems*, pages 107–139. Springer, 2006.

[78] Maria Prandini, Jianghai Hu, John Lygeros, and Shankar Sastry. A probabilistic approach to aircraft conflict detection. *Intelligent Transportation Systems, IEEE Transactions on*, 1(4):199–220, 2000.

[79] Samuel Privara, Jiří Cigler, Zdeněk Váňa, Frauke Oldewurtel, Carina Sagerschnig, and Eva Žáčeková. Building modeling as a crucial part for building predictive control. *Energy and Buildings*, 56:8–22, 2013.

[80] S Joe Qin and Thomas A Badgwell. An overview of industrial model predictive control technology. In *AIChE Symposium Series*, volume 93, pages 232–256. Citeseer, 1997.

[81] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

[82] Ragunathan Raj Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: the next computing revolution. In *Proceedings of the 47th Design Automation Conference*, pages 731–736. ACM, 2010.

[83] Carl Edward Rasmussen and Zoubin Ghahramani. Infinite mixtures of gaussian process experts. *Advances in neural information processing systems*, 2:881–888, 2002.

[84] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[85] Derek Riley, Xenofon Koutsoukos, and Kasandra Riley. Reachability analysis for stochastic hybrid systems using multilevel splitting. In *Hybrid Systems: Computation and Control*, pages 460–464. Springer, 2009.

[86] Derek D Riley, Xenofon Koutsoukos, and Kasandra Riley. Simulation of stochastic hybrid systems using probabilistic boundary detection and adaptive time stepping. *Simulation Modelling Practice and Theory*, 18(9):1397–1411, 2010.

[87] Stephen Roberts, M Osborne, M Ebden, Steven Reece, N Gibson, and S Aigrain. Gaussian processes for time-series modelling. *Phil. Trans. R. Soc. A*, 371(1984):20110550, 2013.

[88] Alex Rogers, Sasan Maleki, Siddhartha Ghosh, and Jennings Nicholas R. Adaptive home heating control through gaussian process prediction and mathematical programming. In *Second International Workshop on Agent Technology for Energy Systems (ATES 2011)*, pages 71–78, May 2011. Event Dates: May 2011.

[89] Axel Rottmann and Wolfram Burgard. Learning non-stationary system dynamics online using gaussian processes. In *Joint Pattern Recognition Symposium*, pages 192–201. Springer, 2010.

[90] Yunus Saatçi, Ryan D Turner, and Carl E Rasmussen. Gaussian process change point models. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 927–934, 2010.

[91] Pedro Santana, Spencer Lane, Eric Timmons, Brian C Williams, and Carlos Forster. Learning hybrid models with guarded transitions. In *AAAI*, pages 1847–1853, 2015.

[92] Johan Schoukens and Rik Pintelon. *Identification of linear systems: a practical guideline to accurate modeling*. Elsevier, 2014.

[93] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264, 2005.

[94] Henry Stark and John W Woods. *Probability and Random Processes with Applications to Signal Processing: International Edition*. Pearson Higher Ed, 2014.

[95] David Sturzenegger, Dimitrios Gyalistras, Manfred Morari, and Roy S Smith. Semi-automated modular modeling of buildings for model predictive control. In *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, pages 99–106. ACM, 2012.

[96] Biao Sun, Peter B Luh, Qing-Shan Jia, Ziyan Jiang, Fulin Wang, and Chen Song. Building energy management: integrated control of active and passive heating, cooling, lighting, shading, and ventilation systems. *Automation Science and Engineering, IEEE Transactions on*, 10(3):588–602, 2013.

[97] Volker Tresp. Mixtures of gaussian processes. In *NIPS*, pages 654–660, 2000.

[98] Ryan D Turner, Marc Peter Deisenroth, and Carl Edward Rasmussen. State-space inference and learning with gaussian processes. In *AISTATS*, pages 868–875, 2010.

[99] Richard A Waltz, José Luis Morales, Jorge Nocedal, and Dominique Orban. An interior algorithm for nonlinear optimization that combines line search and trust region steps. *Mathematical programming*, 107(3):391–408, 2006.

[100] Oliver Watkins and John Lygeros. Stochastic reachability for discrete time systems: An application to aircraft collision avoidance. In *Decision and Control, 2003. Proceedings. 42nd IEEE Conference on*, volume 5, pages 5314–5319. IEEE, 2003.

[101] Yuanchang Xie, Kaiguang Zhao, Ying Sun, and Dawei Chen. Gaussian processes for short-term traffic volume forecasting. *Transportation Research Record: Journal of the Transportation Research Board*, (2165):69–78, 2010.

[102] Xiaojing Zhang, Georg Schildbach, David Sturzenegger, and Manfred Morari. Scenario-based mpc for energy-efficient building climate control under weather and occupancy uncertainty. In *Control Conference (ECC), 2013 European*, pages 1029–1034. IEEE, 2013.

[103] Hengyang Zhao, Daniel Quach, Shujuan Wang, Hai Wang, Haibao Chen, Xin Li, and Sheldon X-D Tan. Learning based compact thermal modeling for energy-efficient smart building management. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pages 450–456. IEEE Press, 2015.