Residual Analysis for Machine Learning

Author:
Xun Shen

Supervisor:
Assoc. Prof. Jiancang Zhuang

Doctor of Philosophy

Department of Statistical Science
School of Multidisciplinary Sciences
The Graduate University for Advanced Studies, SOKENDAI

March 2022

Declaration of Authorship

I, Xun SHEN, declare that this thesis titled, 'Residual Analysis for Machine Learning' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: 沈迅

Date: 2022.2.27

A dissertation submitted to Department of Statistical Science,
School of Multidisciplinary Sciences,
The Graduate University for Advanced Studies, SOKENDAI,
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Advisory Committee

1. Assoc. Prof. Jiancang Zhuang	Institute of Statistical Mathematics, SOKENDAI
2. Assoc. Prof. Keisuke Yano	Institute of Statistical Mathematics, SOKENDAI
3. Asso. Prof. Shinya Nakano	Institute of Statistical Mathematics, SOKENDAI
4. Assoc. Prof. Shunichi Nomura	Waseda University

"Do not boast about tomorrow, for you do not know what a day may bring."

— The Bible–Proverbs 27:1

Acknowledgements

Foremost, I would like to express my sincere appreciation to my advisor Asso. Prof. Jiancang Zhuang for accepting me as a Ph.D student, your warm encouragement, thoughtful guidance, critical comments, and correction of our articles and the thesis. You gave me a lot of insights on how to use residual analysis to improve state-space models and even how to approximate the chance constrained optimization. Even though I had studied some knowledge on state-space models and chance constrained optimization, I was not aware of how statistics could help us to get deep into the problems and broaden our views. With your guidance, I finished my transformation from an engineer to a statistician. I could not have imagined having a better advisor and mentor for my Ph.D study.

Besides my advisor, I would like to thank the rest of my thesis committee: Asso. Prof. Shinya Nakano, Assoc. Prof. Keisuke Yano, and Asso. Prof. Shunichi Nomura, for their encouragement, insightful comments, and hard questions.

My sincere thanks also goes to Prof. Satoshi Ito for giving the course of convex optimization and a lot of insightful advice on robust optimization and chance constrained optimization.

I thank Dr. Tinghui Ouyang for the stimulating discussions on the gap of machine learning, statistics and optimization. Also I thank Dr. Nan Yang for providing us experimental data set to validate our work.

I appreciate my parents for their material and spiritual support in all aspects of my life.

Last but not the least, I would like to thank my lovely wife, Dear Xin Qi. Without your supports and encouragements, I could not have finished this work.

The Graduate University for Advanced Studies, SOKENDAI

Abstract

School of Multidisciplinary Sciences
Department of Statistical Science

Doctor of Philosophy

Residual Analysis for Machine Learning

by Xun Shen

In this thesis, we consider the problem of applying residual analysis to improve machine learning algorithms, including model improvement for state-space models with nonlinear response (Chapter 3), approximate approach for solving chance constrained optimization (Chapter 4), and computing the probabilistic bounds on state trajectories for uncertain nonlinear systems (Chapter 5).

In Chapter 2, we summarizes the original concept of using residual analysis to check the goodness of fit and improve model. The example of residual analysis in regression problems is used to help understanding the concept. Besides, some recent applications of residual analysis have been reviewed, including applications in point processes, time series models, and hidden Markov models. At the end of Chapter 2, we give an intuitive introduction for the idea of applying residual analysis in model improvement of state space models and approximation of chance constrained optimizations, which are the main contributions of this thesis.

In Chapter 3, we address the problem of model improvement for state space models based on residual analysis. The residual of a state-space model is defined. For an SSM with unknown nonlinear response, we propose a novel algorithm for model learning and hidden state inference. A neural network model is used to approximate the unknown nonlinear part in the observation equation, and an Expectation-Maximization (EM) algorithm is proposed to infer the hidden state and learn the parameters in both the linear part and the neural network model, from the given sequences of input data and observation data. In the E-Step, the posterior mean and covariance for the system hidden state given the sequences of the system input and observations is inferred via a Kalman filter-based forward recursion and Rauch-Tung-Streibel smoother backward recursion. In the M-Step, the model parameters are optimized according to the inferred hidden state, input data, and observation data. The M-Step consists of two components: a reconstruction procedure, in which uses the residuals of the linear model to fit the neural network model, and a parametrization procedure, which identifies the parameters in the linear part of the state space model. We apply this newly proposed method to a numerical example and in a case study of battery capacity estimation. The results show that the proposed method can achieve better performance on the model learning and hidden state inference than previously developed tools.

In Chapter 4, we present the residual analysis-based algorithm design for approximate chance constrained program. After reformulating the probabilistic constraints as the quantile function, a sample-based neural network model is used to approximate the quantile function which can dramatically improve the efficiency of the algorithm. The

statistical guarantees of the neural approximation are discussed by showing the convergence of the approximate residual and feasibility analysis. Interval Predictor Model (IPM) of wind power is investigated to validate the proposed method.

In Chapter 5,we present the chance constrained optimization-based algorithms to compute predictive probabilistic bounds on state trajectories for uncertain nonlinear systems. A probabilistic constrained problem is formulated for calculating the probabilistic ellipsoidal bounds of the future trajectory of system states. Scenario approach and sample average approach are used to approximate the probabilistic constrained problem by formulating a deterministic problem with samples of the uncertain parameters in the system. For a given probabilistic level and upper bound of the violation probability, the least number of samples required for calculating the bound can be determined for both scenario approach and sample average approach. The optimality of the solutions obtained by scenario approach and sample average approach is discussed theoretically. The results of numerical example show that more samples will improve the feasibility by sacrificing optimality slightly for sample average approach. However the scenario approach needs more sacrifice on optimality.

Contents

I dedicate this thesis to my family for nursing me with affections and love and their dedicated partnership for success in my life.

Chapter 1

Introduction and Motivation

Residual analysis is often implemented for checking the fit of the statistical model, in order to evaluate how well the model captures the features of the data, and to provide insight into model improvements. At least in 1960s, residual analysis had been widely used in linear regression to check the fitness of the linear models [*Freund et al.*, 1961]. Residual analysis for the linear regression focuses on observing the error distribution because most regression estimators are only consistent if the assumed error distribution is correct [*Goldberger and Arthur*, 1961]. Thus, in regression analysis, residual analysis has been often implemented by residual plots, which is plots of residuals versus either the corresponding fitted values or explanatory variables [*Tsai et al.*, 1998]. Due to the origin of the residual analysis, for most researchers, even statisticians, residual analysis is restricted only to residual plots and regression problems. The fact is that the philosophy of residual analysis has been unconsciously applied in many fields, especially in machine learning algorithm design. Thus, it is necessary to summarize the philosophy of residual analysis formally and give potential insight into not only model improvement but also the machine learning algorithm design.

Fortunately, residual analysis has been developed for point processes and hidden Markov models in recent 30 years [*Baddeley et al.*, 2005, *Bray and Schoenberg*, 2013, *Bray et al.*, 2014, *Buckby et al.*, 2020, *Clements et al.*, 2012, *Ogata*, 1988, *Schoenberg*, 2003, *Zhuang*, 2006, 2015]. The residual analysis in points processes and hidden Markov models includes the definitions of residual, plots of residual and residual decimation-based algorithms for model improvement. Especially, residual decimation-based algorithms are

the main contribution of the above research. Inspired by the above research on residual analysis for point processes and hidden Markov models, the central topic of this thesis is on the development of residual analysis to improve machine learning systems including model improvement of state-space models, approximation of chance constrained optimization, approximation of non-Gaussian residuals. For model improvement of state-space models, we consider how to extract the residuals of the linear state-space models and use the residuals to learn the unknown nonlinear part of the potential real state-space model. For approximation of chance constrained optimization, the existed approach adopts sample-based approximation approaches. In sample-based approximation approaches, samples of random variables are extracted from the sample space to construct a function as an approximation of the original chance constraint which is essentially a function from decision variable to the probability of violation. There is difference between approximation function and the original chance constraint. This thesis investigates the convergence of the residual as sample number increases and how to improve the approximation by introducing neural networks as the approximation function. For approximation of non-Gaussian residuals, we applied mixture density networks to construct the model for the non-Gaussian residuals. In this way, we can improve the predictions by adding the generated predictive residuals.

Chapter 2 of this thesis gives a formal introduction for residual analysis. The development of residual analysis from linear regression to point process and hidden Markov models. The essentials of residual analysis and how to use residual analysis to improve the models and even machine learning algorithms have been formally described.

Chapter 3 of this thesis discusses the strategies in learning the unknown nonlinear parts in the state-space models. Definition of several residual of the state-space models have been reviewed. The state-of-the-art of model improvement methods for state-space models is also briefly reviewed. Besides, we proposed a mixed algorithm to improve the state-space models which combines Expectation-Maximization algorithm and gradient-descent algorithm. Non-parametric models has been used to model the unknown nonlinear parts in the state-space models. The residuals are used to update the nonlinear part in observation function. The rest of the parameters are updated by using gradient-descent algorithm. Numerical simulation has been implemented to validate the proposed algorithm. Besides, the proposed algorithm has been applied to an application case study of battery capacity estimation.

Chapter 4 of this thesis focuses on the residual convergence in approximating chance constrained optimization. First, sample-based approximation for chance constrained optimization has been reviewed. The residuals of approximation the chance constraints relates to the sample number and the function used for approximation. Chapter 4

introduces how the residual vanishes as sample number goes to infinite. Numerical examples and application case studies have been presented to validate the proposed algorithm.

Chapter 5 of this thesis addresses the problem of computing predictive probabilistic bounds of state trajectories for uncertain nonlinear systems. The state trajectory of the uncertain nonlinear system can be described by a nonlinear state-space model. With the fitted nonlinear state-space model and the information of the noise distribution (do not have to be Gaussian), we formulate the computation of the probabilistic bounds as a chance constrained optimization problem. The chance constrained optimization problem can be solved by using scenario approach and sample average approach. We implemented numerical simulations to validate our proposed method. This contribution can be applied to the anomaly detection for time series data based on state-space models.

Chapter 6 summarizes the whole paper and discusses the potential future work.

The material presented in Chapter 3 is based on the still unpublished paper by *Shen and Zhuang* [2021]. For Chapter 4, the content is based on the published papers by *Shen et al.* [2019, 2021]. For Chapter 5, the content is developed from the published paper by *Shen et al.* [2020].

Chapter 2

Residual Analysis

In this chapter, we start from introducing the concept of residual analysis. Then, recent applications of residual analysis have been introduced, including application in point processes, time series models and hidden Markov models. A brief summary of the main contributions of this thesis, residual analysis for state-space models and chance constrained optimization, is given in the last section of this chapter.

2.1 Concept

Before giving the introduction of the general residual analysis, we would like to start from the residual analysis for the regression model with one dimensional output, in which the residual is defined straightly.

Denote $y_i \in \mathbb{R}$ is the $i-$th observation and $x_i \in \mathbb{R}^n$ is the $i-$the input for $i = 1, ..., N$. We have assumptions on the relationship between x_i and y_i:

Assumption 2.1.1. There exists a function $f : \mathbb{R}^n \to \mathbb{R}$ from input to the deterministic part of observation. The uncertain part of the observation is identically independently distributed and obeys zero-mean Gaussian distribution. Thus, y_i can be written by

$$y_i = f(x_i) + \varepsilon_i, \tag{2.1}$$

where $\varepsilon_i \in \mathbb{R}$ denotes the uncertain part of the observation y_i.

Residual analysis is implemented to check whether the residual of the proposed model satisfies the assumption of the model.

If we have a model $\hat{f} : \mathbb{R}^n \to \mathbb{R}$ and it gives the predicted observation

$$\hat{y}_i = \hat{f}(x_i), \tag{2.2}$$

then the residual of model $\hat{f} : \mathbb{R}^n \to \mathbb{R}$ is

$$\hat{e}_i = y_i - \hat{y}_i. \tag{2.3}$$

Then, we can check the statistical property of residual $\hat{e}_i$ to evaluate the goodness of fit. If $\hat{e}_i$ is identically independent distributed and obeys zero-mean Gaussian distribution, we can conclude that $\hat{f}(\cdot)$ is a good regression model.

One possible way to implement the residual analysis is plotting the residual, which is often called residual plot. A residual plot is a graph that shows the residuals on the vertical axis and the independent variable on the horizontal axis. If the points in a residual plot are randomly dispersed around the horizontal axis, a regression model is appropriate for the data. Otherwise, we need to find a better model to fit the data. Residual plot works efficiently for the case with one dimensional observation.

Let us give an example of implementing the residual analysis for model checking of regression models. The toy data of inputs and observations are generated by the following model:

$$y_i = x_i + 0.3 * \sin(2\pi x_i) + \varepsilon_i \tag{2.4}$$

where $\varepsilon_i \sim N(0, 0.1^2)$. We can use a linear model

$$\hat{y}_{\text{linear},i} = \beta x_i \tag{2.5}$$

to fit the data. Besides, neural network model can also be applied, which is written as

$$\hat{y}_{\text{neu},i} = \sum_{k=1}^{m} g_k(x_i, a_i, b_i) \tag{2.6}$$

where $g_k(x_i, a_i, b_i)$ is the active function. Here, we use Gaussian kernel function as active function. The residual plots of two models are given in Fig. 2.1. Obviously, the uncertain part of the observation shows a fairly random pattern. However, the residuals of the linear model is a sin-shaped which shows that the nonlinear part of the determinant component in the observation has not been well addressed. On the other hand, the residual of the neural network model shows a similar random pattern with the uncertain part of the observation. Besides, Fig. 2.2 shows the scatter plots of the residuals. The

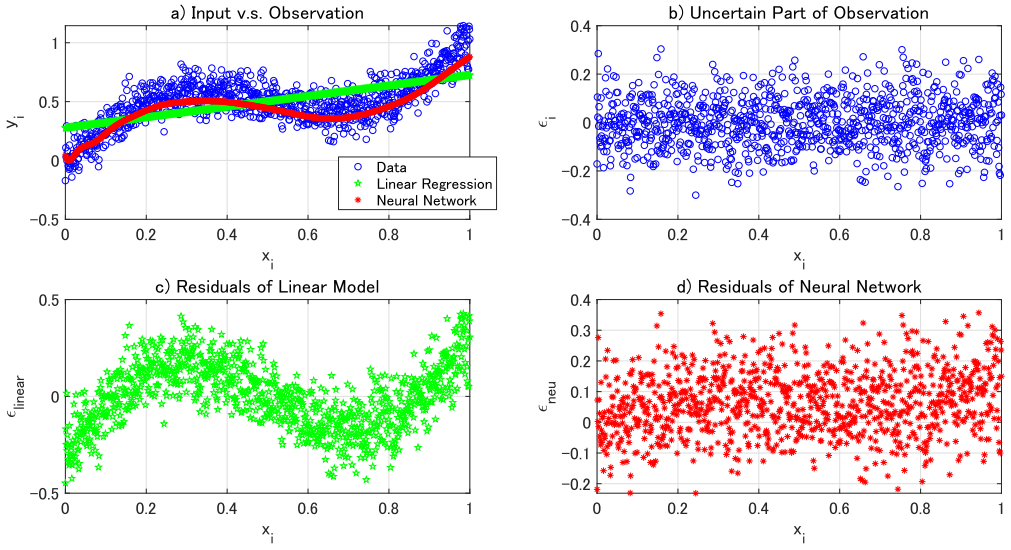

FIGURE 2.1: Residual plots of the toy example

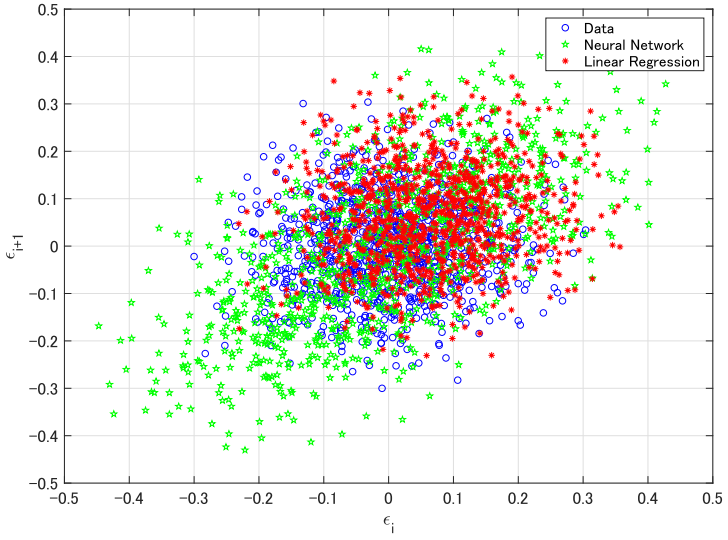

FIGURE 2.2: Scatter plots of the residuals

residual of the neural network model shows a nearly non-correlated pattern which is similar to the uncertain part of the observation. Thus, the neural network model fits the data much better than the linear model and it is close to the real model.

One advantage of the residual analysis using residual plot is that we can directly notice what kind of component should be added to the model. For the linear model in the toy example, we can introduce a sin function to improve the model since the residual of the linear model looks like a sin function. It can achieve even better results that the neural network although the model is much more simple and easier to be trained.

An alternative way is to check the autocorrelation of the residual and whether the residual distribution is Gaussian or not. For the autocorrelation, we can calculate the autocorrelation function $r(k)$ of the residual $\hat{\varepsilon}_i$ for different time lags k as

$$r(k) = \frac{\sum_{i=1}^{n-k}(\hat{\varepsilon}_i - \mathbb{E}[\hat{\varepsilon}])(\hat{\varepsilon}_{i+k} - \mathbb{E}[\hat{\varepsilon}])}{\sum_{i=1}^{n}(\hat{\varepsilon}_i - \mathbb{E}[\hat{\varepsilon}])^2}. \tag{2.7}$$

For checking the distribution, the following equation can be used

$$E = \sqrt{\sum_{i=1}^{N_o}(o_i - \hat{o}_i)^2}, \tag{2.8}$$

where $o_i = F(\varepsilon_i)$ denotes the relative frequency or cumulative probability of Gaussian distribution at ε_i while $\hat{o}_i$ denotes the one of the data.

From the example of residual analysis for the regression model, the preconditions of applying the residual analysis is:

1. a clear definition of residuals;
2. assumption about the statistical property of residuals of a perfect model.

If the preconditions are checked, the rest is to extract the residual and check whether the residual satisfies the assumption. Thus, the implementation of residual analysis for model checking and model improvement can be summarized as

1. Define the residual;
2. Find out the assumption of the residual for a perfect model;
3. Check whether the residual of a model satisfies the assumption or not;
4. Improve the model by making the residual of the model satisfy the assumption.

For the regression problems, the definition of residual is very clear. Although the assumption about the statistical property of residuals varies, we can check the residuals to see whether it satisfies the assumption. However, for the other problems, such as modeling of point processes, hidden Markov models, residual is not clear as in the regression problems. Recent applications of residual analysis have addressed model improvement of point process, hidden Markov models, which will be briefly reviewed in the next section.

2.2 Recent Applications of Residual Analysis

2.2.1 Application in point processes

Point processes are often used to model the intensity of events in the context of temporal or spatiotemporal data, such as seismicity. Most point process models can be specified using the form of moment intensity, conditional intensity or Papangelow intensity. For the sake of explaining the residual analysis for point processes clearly, the case of temporal point process is taken as an example here. A temporal point process is a random measure defined on the time line $T = [0, \infty)$ with Borel σ-algebra $\mathcal{T}$. Let $\mathcal{B}$ denote the ring of bounded Borel sets. A point measure μ on T is a measure that $\mu(A) \in \mathcal{N} = \{n \in \mathbb{Z} | 0 \leq n < \infty\}$ for every $A \in \mathcal{B}$. A fundamental point measure is the Dirac measure δ_x which is defined by

$$\delta_t(A) = \mathbf{1}(t \in A) = \begin{cases} 0, & \text{if } t \notin A \\ 1, & \text{if } t \in A. \end{cases} \tag{2.9}$$

Then, the point measure μ can be obtained by

$$\mu = \sum_{i=1}^{K} a_i \delta_{t_i}, \tag{2.10}$$

where t_i denotes a distinct point in T, $K \in [0, \infty)$ denotes the total number of the events, and a_i is positive integer for t_i. If $a_i = 1$ for every i, the point measure is a simple point measure.

Let $(\Omega, \mathcal{F}, P)$ be a probability space. A point process on T is a measurable mapping N of $(\Omega, \mathcal{F})$ into $M_p, \mathcal{M}_p$ where M_p is the set of all the point measures on T and $\mathcal{M}_p$ is the $\sigma-$algebra. For a random measure M and $A \in \mathcal{T}$, the random variable $M(A)$ is termed as the number of points falling into A, and for a function $h(t)$ in C_K, the integral $N(h) = \int_T f(t)N(dt)$.

The intensity measure m, also called mean measure or first moment measure, of a point process N is defined by the expectations

$$m(A) = \mathbb{E}[N(A)], \text{ for } A \in \mathcal{B}. \tag{2.11}$$

Hence, $m(A)$ is the expected number of points in the set A. More generally, for any positive integer k, let N^k be the point process given by

$$N^k(dt_1, ..., dt_k) = N(dt_1) \cdots N(dt_k). \tag{2.12}$$

The moment measure of order k of the point process N is the measure $m_N^k = \mathbb{E}[N^k]$ or m_k if only process N is considered.

Considering a one-dimensional (temporal) point process N, the moment intensity $v(t)$ can be defined by,

$$v(t)dt = \mathbb{E}[N([t, t+dt])], \tag{2.13}$$

where $N([t, t+dt))$ denotes the number of events occurring in $[t, t+dt)$. Besides, the conditional intensity of the temporal point process is informed by the history of the process and defined as,

$$\lambda(t)dt = \mathbb{E}[N([t, t+dt))|\mathcal{H}_t], \tag{2.14}$$

where $\mathcal{H}_t$ represents historical knowledge of temporal point process N up to time t but not including t. The study of *Zhuang* [2006] proposed residual analysis for point process models, using the conditional intensity to include the knowledge of the process expressed by the history data. Consider a temporal point process N with conditional intensity $\lambda(t)$, the following property holds: for any regular set B which is a countable union of intervals and a non-negative predictable function $h(t)$,

$$\mathbb{E}[\sum_{t_i \in N \cap B} h(t_i)] = \mathbb{E}[\int_B h(t)\lambda(t)dt] \tag{2.15}$$

since

$$\sum_{t_i \in N \cap [0,t]} h(t_i) - \int_{[0,t]} h(t)\lambda(t)dt \tag{2.16}$$

is a zero-mean martingale. Here, simply speaking, the predictable function $h(t)$ can be regarded as a weight and is a stochastic function determined only by previous data of the point process before time t. Thus, for any estimates $\hat{\lambda}(t)$ of $\lambda(t)$, the predictive residual is defined by

$$R(B, \hat{h}, \hat{\lambda}) = \sum_{t_i \in N \cap B} \hat{h}(t_i) - \int_B \hat{h}(t)\hat{\lambda}(t)dt, \tag{2.17}$$

where $\hat{h}(t)$ is used instead of $h(t)$ since $h(t)$ is not available and only the estimation $\hat{h}(t)$ is known. If we get the perfect estimation $\hat{h}(t), \hat{\lambda}(t)$ by choosing the model and tuning the parameters in the model, $R(B, \hat{h}, \hat{\lambda})$ should be a zeros-mean martingale. Thus, we regard $R(B, \hat{h}, \hat{\lambda})$ as the residual of the model and expect to obtain a zero-mean residual.

Finally, the Papangelou intensity is calculated as,

$$\lambda_p(t)dt = \mathbb{E}[N([t - dt/2, t + dt/2])|\mathcal{L}_{[t-dt/2, t+dt/2]}], \tag{2.18}$$

where $\mathcal{L}_{[t-dt/2, t+dt/2]}$ represents information of N everywhere except the time interval

$[t - dt/2, t + dt/2]$. *Baddeley et al.* [2005] and *Zhuang* [2015] used the Papangelou intensity to construct the residual analysis method for point process. Similar to the case of using the conditional intensity, the residual with respect to the Papangelou intensity can be defined by using the Georgii-Zessin-Nguyen formula. The residual related to the Papangelou intensity is also called the exvisive residual, which is written as

$$R(B, \hat{h}, \lambda_p) = \sum_{t_i \in N \cap B} \hat{h}(t_i) - \int_B \hat{h}(t)\lambda_p(t)dt, \tag{2.19}$$

where $\hat{h}(t)$ is an exvisible function instead of a non-negative predictable function $h(t)$ in the case of using conditional intensity. The exvisible function $h(t)$ is a function with respect to the occurence pattern of N throughout time, except at time t. Namely, not only the data before t is used, the data after t is also used. If $R(B, \hat{h}, \lambda_p) \approx 0$, we can assume that the fitted model is a good approximation of the real potential model.

2.2.2 Application in time series models

Wang et al. [2018] developed residuals for zero-inflated autoregressive time series by adjusting the concepts of residual analysis for point processes from *Zhuang* [2006].

For a time series X_t, the residual of the time series model is given by

$$R(n) = \sum_{t=1}^{n} \left(\hat{h}(t)X_t - \hat{h}(t)\mathbb{E}[X_t|\mathcal{H}_t] \right) \tag{2.20}$$

where $\mathcal{H}_t$ represents the history $X_1, ..., X_{t-1}$. Notice that $\mathbb{E}[X_t|\mathcal{H}_t]$ is obtained from the chosen time series model. $\hat{h}(t)$ can be chosen according to the requirement. If the raw residual is important, $\hat{h}(t)$ can be set as 1 for every t. In this case, we have discrete time t and

$$\mathbb{E}[\sum_{t=1}^{n} \hat{h}(t)X_t] = \mathbb{E}[\sum_{t=1}^{n} \hat{h}(t)\mathbb{E}[X_t|\mathcal{H}_t]]. \tag{2.21}$$

Consider the process

$$D(n) = \sum_{t=1}^{n} \left(\hat{h}(t)X_t - \sum_{t=1}^{n} \hat{h}(t)\mathbb{E}[X_t|\mathcal{H}_t] \right) \tag{2.22}$$

is a zero-mean martingale because,

$$\mathbb{E}[D(n) - D(n-1)|\mathcal{H}_n] = \mathbb{E}[h(n)X_n - h(n)\mathbb{E}[X_n|\mathcal{H}_n]|\mathcal{H}_n] = 0. \tag{2.23}$$

Similar to the residual analysis for the point processes, if $R(n) \approx D(n) \to 0$, it is reasonable to assume that the model has a good fit to the data. Here, n should be

sufficiently large to check the goodness of fit. The asymptotic normality of $D(n)$ is asserted by the central limit theorem for martingales. When n is sufficiently large, $R(n)$ and $D(n)$ should have the same distribution with zero-mean.

2.2.3 Application in hidden Markov models

An hidden Markov model is used to model time series data when the observations are dependent on an underlying unobserved Markov chain. Let x_t denote the observation at time $t = 1, ..., T$. x_t might be realized from the finite m members of a family of distributions, $f(x_t|S_t = i)$ with $i = 1, ..., m$. $S_t = i$ means that the hidden state of the Markov chain at time t is i. Notice that the conditional probability of hidden state and observation satisfies the following conditions,

$$P(S_t = s_t|S_1 = s_1, ..., S_{t-1} = s_{t-1}) = P(S_t = s_t|S_{t-1} = s_{t-1}) \tag{2.24}$$

and

$$f(x_t|x_1, ..., x_t - 1, S_1 = s_1, ..., S_t = s_t) = f(x_t|S_t = s_t). \tag{2.25}$$

The parameters to be estimated for an HMM include the parameters associated with the hidden state distribution, a vector $\delta \in \mathbb{R}^m$ with $\delta_i \in [0, 1], \sum_i \delta_i = 1$ for the initial probabilities and a matrix $\Gamma \in \mathbb{R}^{m\times m}$ for the transition probabilities. The element γ_{ij} in Γ is within $0, 1$ and $\sum_j \gamma_{ij} = 1$. Estimation of δ and Γ can be achieved by directly maximising the likelihood or by using the Expectation-Maximization (EM) algorithm.

To check the goodness of fit for the hidden Markov models, *Buckby et al.* [2020] extended the predictive and exvisive residuals to hidden Markov models.

The predictive residuals of hidden Markov models correspond to the conditional intensity of a point process model, namely, the expectation of the occurrence of an event in unit time given the history. In hidden Markov models, the expectation of X_t given the previous observations is,

$$\mathbb{E}[X_t|\mathcal{H}_t] = \mathbb{E}[X_t|X_1, ..., X_{t-1}] = \sum_{i=1}^{m} \frac{\alpha_{t-1}\Gamma_i\mathbb{E}[X_t|S_t = i]}{\alpha_{t-1}1'} \tag{2.26}$$

where α_{t-1} is the forward probability with the jth element,

$$\alpha_{t-1}(j) = P(X_1, ..., X_{t-1}, S_{t-1} = j). \tag{2.27}$$

Details for calculating forward and backward probabilities within the EM algorithm is introduced in *Zucchini and MacDonald* [2009]. Using the expectation calculated by

(2.26), the raw predictive residual with $h(t) = 1$ becomes,

$$R_n^p = \sum_{t=1}^{n} (X_t - \mathbb{E}[X_t|\mathcal{H}_t]) \tag{2.28}$$

where the parameters are those estimated from the data. The standardized residuals for $n = 2, ..., T$ are calculated as

$$\bar{R}_n^p = \frac{R_n^p}{\sqrt{\sum_{t=1}^{n}(R_t^p - R_{t-1}^p)^2}} \tag{2.29}$$

where $\sqrt{\sum_{t=1}^{n}(R_t^p - R_{t-1}^p)^2}$ is the standard deviation of R_n^p, as defined in *Wang et al.* [2018] and $R_0^p = 0$. Notice that $\bar{R}_n^p$ for $n = 1, ..., T$ are not iid and we just expect to see $\bar{R}_n^p$ fall within the 95% confidence interval of a standard normal distribution if the fitted model is close to the true model as $\bar{R}_n^p$ obeys $N(0, 1)$ for each large enough n. Furthermore, standardized raw residuals $\bar{R}_{k,L}^p$ for fixed intervals of observations can also be calculated, where $k = 0, 1, ..., K$ and L is the interval size. The raw residuals for fixed interval is written as

$$R_{k,L}^p = \sum_{t=Lk+1}^{Lk+L} (X_t - \mathbb{E}[X_t|\mathcal{H}_t]) \tag{2.30}$$

and then the standardized raw residuals is defined as

$$\bar{R}_{k,L}^p = \frac{R_{k,L}^p}{\sqrt{\sum_{t=Lk+1}^{Lk+L}(R_t^p - R_{t-1}^p)^2}}. \tag{2.31}$$

Notice that $\bar{R}_{k,L}^p$ can be considered iid for $k = 1, ..., K$ and goodness of fit can be assessed by comparing the distribution of $\bar{R}_{k,L}^p$ to the standard normal distribution.

For the raw exvisive residuals, it is determined in a similar way to the predictive residuals. The difference is that the expectation of X_t given all of the other observations should be considered while only observations prior to time t is needed in the predictive residuals. The expectation is written as

$$\mathbb{E}[X_t|\mathcal{E}_t] = \mathbb{E}[X_t|X_1, ..., X_{t-1}, X_{t+1}, ..., X_T] = \sum_{i=1}^{m} \frac{\alpha_{t-1}\Gamma_i\beta_t(i)\mathbb{E}[X_t|S_t = i]}{\sum_{j=1}^{m}\alpha_{t-1}\Gamma_j\beta_t(j)} \tag{2.32}$$

where $\beta_t(i) = P(X_{t+1}, .., X_T|S_t = i)$ is the backward probability. The raw exvisive residual can then be written as

$$R_n^e = \sum_{t=1}^{n} (X_t - \mathbb{E}[X_t|\mathcal{E}_t]) \tag{2.33}$$

where the parameters are those estimated from the data. The process of standardising the raw exvisive residuals is to divide the standard deviation of R_n^e and obtain the

following expression

$$\bar{R}_n^e = \frac{R_n^e}{\sqrt{\sum_{t=1}^{n}(R_t^e - R_{t-1}^e)^2 + 2\sum_{i\neq j}((R_i^e - R_{i-1}^e)(R_j^e - R_{j-1}^e))}}. \tag{2.34}$$

Again, we expect $\bar{R}_n^e$ to fall in 95% confidence interval of a standard normal distribution even though $\bar{R}_n^e$ is not iid. Similar to the case of predictive residuals, we can first create idd exvisive interval residuals, $R_{k,L}^e$ as

$$R_{k,L}^p = \sum_{t=Lk+1}^{Lk+L} (X_t - \mathbb{E}[X_t|\mathcal{E}_t]) \tag{2.35}$$

and then obtain the standardized exvisive interval residuals as

$$\bar{R}_{k,L}^e = \frac{R_{k,L}^e}{\sqrt{\sum_{t=1}^{n}(R_t^e - R_{t-1}^e)^2 + 2\sum_{i\neq j}((R_i^e - R_{i-1}^e)(R_j^e - R_{j-1}^e))}}. \tag{2.36}$$

Then, goodness of fit is assessed by comparing the distribution of $\hat{R}_{k,L}^e$ to the standard normal distribution.

2.3 Extending Residual Analysis to State-Space Models and Chance Constrained Optimizations

2.3.1 Contribution to model improvement of state space models

The general expression of state space models is written as

$$x_{t+1} = f(x_t, u_t) + v_t, \tag{2.37}$$

$$y_t = g(x_t, u_t) + w_t, \tag{2.38}$$

where $t \in \mathbb{Z}$ is the time index, $x_t \in \mathbb{R}^k$ represents a k-dimension state vector, $u_t \in \mathbb{R}^c$ is a c-dimension system input vector, $y_t \in \mathbb{R}^d$ is the system output, $v_t \in \mathbb{R}^k$ and $w_t \in \mathbb{R}^d$ denote the l-dimension system noise with probability density function as $q(v)$ and the m-dimension observation noise with probability density function as $r(w)$, respectively, and $f : \mathbb{R}^k \times \mathbb{R}^c \times \mathbb{R}^k \to \mathbb{R}^k$ and $g : \mathbb{R}^k \times \mathbb{R}^c \times \mathbb{R}^d \to \mathbb{R}^d$ represent general formulations of state equation and observation equation, respectively. The initial state vector x_0 is distributed according to the probability density $p_0(x_0)$.

The residual analysis for state space model is similar to the regression problems. The residual is the difference between y and model prediction. However, the model improvement is different since the hidden state inference should be considered in the state space models. Besides, the error of model does not only influence the parameter estimation but also the inference of the hidden state inference, which are integrated in the residual, which makes the residual analysis for the state space models challenging. The contributions for the model improvement of state space models in this thesis are summarized as

1. We give the definition of residual in state space models by adjusting the concepts of residual analysis for hidden Markov models in *Buckby et al.* [2020];
2. We propose an EM algorithm-based method to improve the state space model with nonlinear response by using the information in the residual. The idea was inspired by the algorithms for point process in *Zhuang* [2006, 2015];
3. We modify the model improvement method for the state space model with nonlinear response to the general state space model.

2.3.2 Contribution to approximating chance constrained optimizations

Chance constrained optimization can be generally expressed as:

$$\begin{aligned} &\min_{u\in\mathcal{U}} J(u) \\ &s.t. \quad \Pr\{h(u,\delta)\leq 0\}\geq 1-\alpha,\ \delta\in\Delta,\ \alpha\in(0,1) \end{aligned} \tag{2.39}$$

where $u\in\mathcal{U}\subset\mathbb{R}^{n_u}$ is the decision variable, the decision variable domain $\mathcal{U}$ is supposed to be bounded, $\delta\in\Delta\subset\mathbb{R}^{n_\delta}$ is an uncertain parameter vector, the set Δ is the sample space of δ on which a probability measure Pr is defined, α is a given probability level for violation of chance constraints. Moreover, $J(u):\mathcal{U}\to\mathbb{R}$ and $\forall\delta\in\Delta, h(u,\delta):\mathcal{U}\times\Delta\ \to\mathbb{R}$ are continuous and differentiable in u.

Notice that the feasible region of u defined by $\Pr\{h(u,\delta)\leq 0\}\geq 1-\alpha$ cannot be explicitly obtained in most cases. Without the knowledge of the feasible region, it is impossible to solve problem (2.39). The only possible way is to find an approximate function of constraint $\Pr\{h(u,\delta)\leq 0\}\geq 1-\alpha$ and use it instead of the chance constraint.

One fact is that $\Pr\{h(u,\delta)\leq 0\}$ is a function from $\mathbb{R}^{n_u}$ to the interval [0,1]. Let us denote $F(u)$ for $\Pr\{h(u,\delta)\leq 0\}$. In an alternative way, we can also use quantile function instead

of probability. The quantile function can be written as

$$Q^{1-\alpha}(X) = \inf\{x \in \mathbb{R} | \Pr\{X \leq x\} \geq 1 - \alpha\}. \tag{2.40}$$

The value of quantile function also depends on u. Let use denote $G(u)$ for it. The problem is that the information of neither $F(u)$ nor $G(u)$ is available. In this thesis, we extract samples of u and δ to obtain the approximations of $F(u)$ and $G(u)$. Then, use the approximations as the constraint to establish an approximate of the chance constrained optimization. By solving the approximate problem, we could get the approximate solution of the original chance constrained optimization. Notice that there is residual of the approximations of $F(u)$ and $G(u)$. We prove that the residual converges to zero if the sample numbers of u and δ increase to infinite. Also, the feasible region and optimal solution of the approximate problem also converge to the original ones.

Chapter 3

Model Improvement for State-Space Models

3.1 Background and Related Works

State-Space Models are general representations of systems observed over time and are widely used for dynamical system modeling [*Shen et al.*, 2020] and time series data analysis [*Durbin and Koopman*, 2000, *Katzfuss et al.*, 2020, *Kitagawa*, 1987, 1996, *Kreuzer and Czado*, 2020]. However, one has to solve several problems of computing the conditional distributions of state variables given all or part of the data and estimators of the unknown parameters to apply SSMs to data [*Sorenson*, 1982]. The above processes can also be regarded as model learning and hidden state inference.

The model learning and hidden state inference for the linear SSMs involved with white Gaussian noises are relatively simple. With maximizing the innovation form of the likelihood function of SSMs' parameter vector, the model learning and hidden state inference can be implemented simultaneously. In every iteration, the hidden state inference and output innovation can be computed by the Kalman filter. Then, the parameter vector will be updated by a gradient descent method [*Tanizaki*, 2009]. The Cholesky decomposition is implemented and then Newton-Raphson approach is used to maximize the likelihood to obtain the model parameters and hidden state estimation. Besides, the maximum likelihood method with penalized likelihood is proposed in *Zhu and Wu* [2007] to estimate the time-varying parameter in linear SSMs. *Watson and Engle* [1983]

and *Watanabe* [1985] achieved the above processes by using Expectation-Maximization (EM) method which improved the convergence speed.

However, for SSMs with the unknown nonlinear response (nonlinear observation function), the model learning and hidden state inference become more difficult. Kalman filter cannot obtain accurate hidden state inference for SSMs with the unknown nonlinear response. The error in the hidden state inference will propagate to the model learning. One typical application of SSMs with nonlinear response is the battery capacity estimation. The SSM for battery capacity estimation has linear state equation and nonlinear observation function [*Pattipati et al.*, 2014, *Zheng et al.*, 2016]. The battery capacity is defined as State-of-Charge (SoC). In this case, the linear SSM-based model learning and hidden state inference are with poor accuracy. Thus, in *Plett* [2006], sigma-point Kalman filtering is applied to estimate SoC of battery in the hybrid electric vehicle. A look up table from SoC to the Open-Circuit-Voltage (OCV) is used to modeling the nonlinear response in the observation function. However, the model for approximating the nonlinear response has to be fitted with the measurements of hidden state. Namely, the model learning, especially the approximation of the nonlinear response, cannot be done with only input and output data. In *Luzi et al.* [2019, 2020], neural network models has been applied to model the nonlinear response. Although the accuracy of the hidden state estimation has been improved by particle filter and accurate models, the calculation burden for hidden state estimation is increased. Model learning and hidden state inference for SSMs with nonlinear response should be investigated.

A lot of research has been done towards solving hidden state inference problem in the SSMs with nonlinearity. Mixture Kalman filter has been presented in *Cheng and Liu* [2000] which uses Gaussian mixture models to model the non-zero noise and includes some nonlinearity in the noise. *Niemi and West* [2010] improved the mixture Kalman filter by using Metropolis-Hastings method-based regenerating process to eliminate potential degeneracy of mixture components. *Koyama et al.* [2010] uses an asymptotic series expansion to approximate the state's conditional mean and variance together with a Gaussian conditional distribution. Nonlinearity can be included in the asymptotic series expansion. *Yang et al.* [2013] proposed a feedback particle filter in which the nonlinear filter is approximated by solving an optimal control problem to get the optimal state feedback adaptive Kalamn gain. A robust filter has been proposed in *Calvet et al.* [2015] to mitigate the degeneracy problems caused by uncertainty of parameters. It can also bound the hidden state estimation error caused by the unknown nonlinear response when the unknown nonlinear response is small. Moreover, an offline, iterated particle filter is proposed in *Guarniero et al.* [2017] to implement statistical inference including model learning and hidden state inference in general state space models. A sequence of positive functions are used as auxiliaries to approximate the distribution functions. The

above works have made many contributions to the development of hidden state inference. However, the model structure are fixed and the problem of model improvement is not touched.

The work involved model improvement is less. In *Ghahramani and Roweis* [1999], a general learning algorithm combining EM algorithm and Extended Kalman Filter (EKF) is proposed for model learning and hidden state estimation of nonlinear SSMs. The unknown nonlinear parts are approximated by Radial Basis Function (RBF). However, the implementation of EKF brings the error on the hidden state inference. The error will propagate to the parameters identification including both the parameters in linear and nonlinear parts. In *Deisenroth et al.* [2012], Gaussian Process (GP) models were applied to model both state equation and observation equation. Then, an iterative algorithm is proposed to learn the GP models and estimate the hidden state. In the validation, different nonlinear filters including EKF, unscented Kalman filter (UKF), Cubature Kalman filter (CKF), were compared.

3.2 Main Contributions

This thesis proposes a simplified algorithm compared to *Ghahramani and Roweis* [1999] and *Deisenroth et al.* [2012] from the point view of residual analysis, inspired by the works in *Wang et al.* [2018], *Zhuang* [2006, 2015], *Zhuang et al.* [2002], and *Buckby et al.* [2020]. *Zhuang* [2006, 2015] and *Zhuang et al.* [2002] present the residual analysis for model improvement of point process model. Non-parametric models has been applied to approximate the residual. Moreover, the proposed model improvement algorithm is with the similar framework as the EM algorithm. The stochastic declustering step is similar to the hidden state inference, and the following reconstruction and parametrization steps are for learning the non-parametric models and parametric models. The methodology has been extended to the time series model and hidden Markov model as described respectively in *Wang et al.* [2018] and *Buckby et al.* [2020]. Our new method for learning SSMs with nonlinear response is rooted in these previous works. The SSM with nonlinear response has been separated into two parts, linear part and nonlinear part. The linear part includes the linear state equation and linear part of the observation equation while the nonlinear part is the remained nonlinear part of the observation equation. Then, the similar framework can be applied. Using linear Kalman filter to estimate the hidden state which is expectation step, applying neural network models or non-parametric models to approximate the nonlinear part which is reconstruction step, and updating the parameters in the linear part which is the parametrization step. Namely, our new method can decimate the nonlinear response from the SSM and linear Kalman filter can

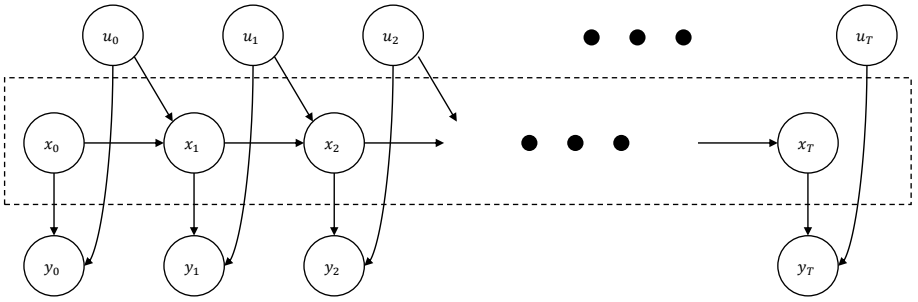

FIGURE 3.1: Probabilistic graphical model for stochastic dynamical systems with hidden states x_t, inputs u_t, and observations y_t.

be used to do the hidden state inference, which releases the computation burden and but also reserves the estimation accuracy.

3.3 Preliminaries

3.3.1 General state-space models

An SSM is often used to model time series data of a stochastic dynamical system with hidden states, inputs, and observations. A probabilistic graphical model for this stochastic dynamical system is shown in Fig. 3.1. The observation y_t depends on the input u_t and the hidden state x_t. Notice that the data of y_t and u_t is available while x_t cannot be measured. The general expression of SSMs is written as

$$x_{t+1} = f(x_t, u_t) + v_t, \tag{3.1}$$

$$y_t = g(x_t, u_t) + w_t, \tag{3.2}$$

where $t \in \mathbb{Z}$ is the time index, $x_t \in \mathbb{R}^k$ represents a k-dimension state vector, $u_t \in \mathbb{R}^c$ is a c-dimension system input vector, $y_t \in \mathbb{R}^d$ is the system output, $v_t \in \mathbb{R}^k$ and $w_t \in \mathbb{R}^k$ denote the system noise with probability density function as $q(v)$ and the observation noise with probability density function as $r(w)$, respectively, and $f : \mathbb{R}^k \times \mathbb{R}^c \times \mathbb{R}^k \to \mathbb{R}^k$ and $g : \mathbb{R}^k \times \mathbb{R}^c \times \mathbb{R}^k \to \mathbb{R}^d$ represent general formulations of state equation and observation equation, respectively. The initial state vector x_0 is distributed according to the probability density $p_0(x_0)$.

3.3.2 Residuals of state-space models

Here the residuals developed by *Zhuang* [2015] are adapted for use with state space models. The state state model is discrete with integer time index t. Besides, the distribution of the state vector should be considered at time t.

The predictive residuals of state space model is written as

$$R^p(n, D, \theta) = \sum_{t=0}^{n} (D(y_{t+1}) - D(\mathbb{E}\{y_{t+1}|Y_t, U_t\})) \tag{3.3}$$

for general cases or

$$R^p(n, h, \theta) = \sum_{t=0}^{n} (h(t+1)y_{t+1} - h(t+1)\mathbb{E}\{y_{t+1}|Y_t, U_t\}) \tag{3.4}$$

for the weighted cases where $h(t)$ is a predictable function. If $h(t) = 1$, the raw predictive residual is expressed as

$$R^p(n, \theta) = \sum_{t=0}^{n} (y_{t+1} - \mathbb{E}\{y_{t+1}|Y_t, U_t\}). \tag{3.5}$$

The above different kinds of predictive residuals have common part $\mathbb{E}\{y_{t+1}|Y_t, U_t\}$ which denotes the conditional expectation of y_{t+1} by giving the previous data $Y_t,\ U_t$. The conditional expectation corresponds to the conditional intensity of a point process model. It is written as

$$\mathbb{E}\{y_{t+1}|Y_t, U_t\} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} y_{t+1} p_{y_{t+1}}(y_{t+1}|x_{t+1}) p_{x_{t+1}}(x_{t+1}|Y_t, U_t) dx_{t+1} dy_{t+1}, \tag{3.6}$$

where $p_{x_{t+1}}(x_{t+1}|Y_t, U_t)$ denotes the predictive density for x_{t+1} with history data $Y_t,\ U_t$.

Similar to the predictive residuals, the smooth residuals of state space model is written as

$$R^s(n, D, \theta) = \sum_{t=0}^{n} (D(y_{t+1}) - D(\mathbb{E}\{y_{t+1}|Y_T, U_T\})) \tag{3.7}$$

for general cases. Note that we use the whole available data $Y_T,\ U_T$ instead of the history data $Y_t,\ U_t$. For the weighted cases, we use the following smooth residuals

$$R^s(n, h, \theta) = \sum_{t=0}^{n} (h(t+1)y_{t+1} - h(t+1)\mathbb{E}\{y_{t+1}|Y_T, U_T\}) \tag{3.8}$$

where $h(t)$ is a predictable function. If $h(t) = 1$, the raw smooth residual is expressed as

$$R^s(n, \theta) = \sum_{t=0}^{n} (y_{t+1} - \mathbb{E}\{y_{t+1}|Y_T, U_T\}). \tag{3.9}$$

As in the predictive residuals, the above different kinds of smooth residuals have common part $\mathbf{E}\{y_{t+1}|Y_T, U_T\}$ which denotes the conditional expectation of y_{t+1} by giving the whole data Y_T, U_T. It is written as

$$\mathbb{E}\{y_{t+1}|Y_T, U_T\} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} y_{t+1} p_{y_{t+1}}(y_{t+1}|x_{t+1}) p_{x_{t+1}}(x_{t+1}|Y_T, U_T) dx_{t+1} dy_{t+1}, \tag{3.10}$$

where $p_{x_{t+1}}(x_{t+1}|Y_T, U_T)$ denotes the probability density for x_{t+1} with Y_T, U_T which can be obtained by smoother algorithms [*Kitagawa*, 1996].

The above residuals of state-space models are constructed by the observations. Residuals based on the hidden state inference can also be constructed when we assume that both v_t and w_t obey white Gaussian and are independently identically distributed. Note that

$$x_{t+1} - f(x_t, u_t) = v_t, \tag{3.11}$$

thus,

$$\mathbb{E}\{x_{t+1} - f(x_t, u_t)\} = \mathbb{E}\{v_t\} = 0. \tag{3.12}$$

Thus, the predictive state residuals of state space model can be defined by

$$R_x^p(n, \theta) = \sum_{t=0}^{n} (\mathbb{E}\{x_{t+1}|Y_t, U_t\} - \mathbb{E}\{f(x_t, u_t)|Y_{t-1}, U_{t-1}\}). \tag{3.13}$$

Besides, the smooth state residuals of state space model can be defined by

$$R_x^s(n, \theta) = \sum_{t=0}^{n} (\mathbb{E}\{x_{t+1}|Y_T, U_T\} - \mathbb{E}\{f(x_t, u_t)|Y_T, U_T\}). \tag{3.14}$$

Due to equation (3.12), we know that both $R_x^p(n, \theta)$ and $R_x^s(n, \theta)$ should have zero-mean if the model is good fit for the data.

3.3.3 Hidden state inference for general dynamical systems

Introducing the Monte Carlo Filter into calculation of $\hat{y}_{t+1}$, we can obtain the recursive algorithm for inferring the hidden state of SSMs with Y_t, U_t. We use many of the realizations of each density function to approximate that distribution. More specifically, using N_p particles to express predictive distribution of x_t with Y_{t-1}, U_{t-1} as follows:

$$\{x_{p,t}^{(1)}, ..., x_{p,t}^{(i)}, ..., x_{p,t}^{(N_\mathrm{p})}\} \sim p_{x_t}(x_t|Y_{t-1}, U_{t-1}) \tag{3.15}$$

where $x_{p,t}^{(i)}$ is the $i-$th particle or realization of x_t according to $p_{x_t}(x_t|Y_{t-1}, U_{t-1})$. In effect, we approximate the distributions by the empirical distributions determined by

the set of particles. We can also use N_p particles to express filtered estimation of x_t with Y_t, U_t as follows:

$$\{x_{f,t}^{(1)}, ..., x_{f,t}^{(i)}, ..., x_{f,t}^{(N_\mathrm{p})}\} \sim p_{x_t}(x_t|Y_t, U_t) \tag{3.16}$$

where $x_{f,t}^{(i)}$ is the $i-$th particle or realization of x_t according to $p_{x_t}(x_t|Y_t, U_t)$.

$\{x_{p,t}^{(1)}, ..., x_{p,t}^{(i)}, ..., x_{p,t}^{(N_\mathrm{p})}\}$ are essentially independent realizations of the $p_{x_t}(x_t|Y_{t-1}, U_{t-1})$. Since

$$p_{x_t}(x_t|Y_{t-1}, U_{t-1}) = p_{x_t}(x_t|x_{t-1}) p_{x_{t-1}}(x_{t-1}|Y_{t-1}, U_{t-1}), \tag{3.17}$$

we can calculate $x_{p,t}^{(i)}$ by

$$x_{p,t}^{(i)} = f(x_{f,t-1}^{(i)}, u_{t-1}) + v_{t-1}^{(i)} \tag{3.18}$$

where $\{v_{t-1}^{(1)}, ..., v_{t-1}^{(i)}, ..., v_{t-1}^{(N_\mathrm{p})}\} \sim q(v)$ are the particles of system noise. Particle $x_{f,t}^{(i)}$ are essentially obtained by the resampling of $\{x_{p,t}^{(1)}, ..., x_{p,t}^{(i)}, ..., x_{p,t}^{(N_\mathrm{p})}\}$. Given the observation y_t, u_t, and the particle $x_{p,t}^{(i)}$, compute $\alpha_t^{(i)}$, the likelihood of the particle $x_{p,t}^{(i)}$ based on the observation y_t, u_t. That is

$$\alpha_t^{(i)} = p_{y_t}(y_t|x_{p,t}^{(i)}) = r(y_t - g(x_{p,t}^{(i)}, u_t)) \tag{3.19}$$

for $i = 1, ..., N_\mathrm{p}$. Note that $r(\cdot)$ is the density of the observation noise w. Then, for $i = 1, ..., N_\mathrm{p}$, $x_{f,t}^{(i)}$ is defined by

$$x_{f,t}^{(i)} = \begin{cases} x_{p,t}^{(1)} & \text{with probability } \alpha_t^{(1)}/(\alpha_t^{(1)} + ... + \alpha_t^{(N_\mathrm{p})}) \\ ... & ... \\ x_{p,t}^{(N_\mathrm{par})} & \text{with probability } \alpha_t^{(N_\mathrm{p})}/(\alpha_t^{(1)} + ... + \alpha_t^{(N_\mathrm{p})}). \end{cases} \tag{3.20}$$

The obtained $\{x_{f,t}^{(1)}, ..., x_{f,t}^{(i)}, ..., x_{f,t}^{(N_\mathrm{p})}\}$ can be regarded as the realizations of the filter, $p_{x_t}(x_t|Y_t, U_t)$ which is verified in *Kitagawa* [1996].

3.3.4 Hidden state inference for linear dynamical systems

If the dynamical system is linear and the noises are Gaussian, the optimal hidden state inference can be exactly obtained as a feedback of the predictive residuals. Linear dynamical systems with additive white Gaussian noises can be written as

$$x_{t+1} = Fx_t + Hu_t + v_t, \tag{3.21}$$

$$y_t = C + Gx_t + Ju_t + w_t. \tag{3.22}$$

Here, the prior probability distribution over initial states is assumed to be Gaussian with $P_{0|0} \in \mathbb{R}^{k\times k}$ as the covariance. The covariances of v_t and w_t are denoted by $Q \in \mathbb{R}^{k\times k}$ and $R \in \mathbb{R}^{k\times k}$, repectively. Since the prior probability distribution over initial states is taken to be Gaussian and the Gaussian distribution is closed under the linear operations applied by state evolution and observation mapping, then the joint probabilities of all states and observations at future times are also Gaussian. Therefor, all probability distributions over the hidden state variables can be fully described by their means and covariance matrices. Hidden state inference is to compute the posterior mean and covariance for every x_t given some sequence of system inputs and observations. The algorithm consists of two parts. The first one is a forward recursion, known as the Kalman filter, in which $u_0, ..., u_t$ and $y_0, ..., y_t$ are used to calculate the mean and covariance of x_t. The second one is a backward recursion which is known as Rauch-Tung-Streibel smoother or the Kalman smoother. $u_t, ..., u_{T-1}$ and $y_{t+1}, ..., y_T$ are used to calculate the mean and covariance of x_t. The algorithm of hidden state inference is summarized as following:

1. Set the mean value of x_0 as $\hat{x}_{0|0}$, the covariance of x_0 as $P_{0|0}$, the covariances of v_t, w_t as Q, R, respectively.

2. The Kalman filter [*Kalman*, 1960], repeat the following steps for $t = 1, ..., T$:

   a) Calculate the predict mean of state

   $$\mathbf{E}\{x_t|Y_{t-1}, U_{t-1}\} = \hat{x}_{t|t-1} = F\hat{x}_{t-1|t-1} + Hu_{t-1}; \tag{3.23}$$

   b) Calculate the predict covariance of state

   $$P_{t|t-1} = FP_{t-1|t-1}F^T + Q; \tag{3.24}$$

   c) Calculate the predict innovation

   $$I_{y_t} = y_t - \mathbb{E}\{y_t|Y_{t-1}, U_{t-1}\} \tag{3.25}$$

   where $\mathbb{E}\{y_t|Y_{t-1}, U_{t-1}\} = C + G\hat{x}_{t|t-1} + Ju_{t-1}$;

   d) Calculate the feedback gain

   $$K_t = P_{t|t-1}G^T(GP_{t|t-1}G^T + R)^{-1}; \tag{3.26}$$

   e) Calculate the filter mean of state

   $$\mathbb{E}\{x_t|Y_t, U_t\} = \hat{x}_{t|t} = \hat{x}_{t|t-1} + K_tI_{y_t} \tag{3.27}$$

and obtain the filter covariance of state

$$P_{t|t} = P_{t|t-1} - K_t G P_{t|t-1}. \tag{3.28}$$

3. Rauch-Tung-Streibel smoother (Kalman smoother):

   a) Initialization: $\tilde{x}_T = \hat{x}_{T|T}, \tilde{P}_T = P_{T|T}$.

   b) Repeat the following steps for $t = T-1, ..., 1$:

      * Calculate the gain

$$\tilde{K}_t = P_{t|t} F^T P^{-1}_{t+1|t}. \tag{3.29}$$

      * Update the mean

$$\tilde{x}_t = \hat{x}_{t|t} + \tilde{K}_t(\tilde{x}_{t+1} - F\hat{x}_{t|t} - H\hat{x}_{t|t}). \tag{3.30}$$

      * Update the covariance

$$\tilde{P}_t = P_{t|t} + \tilde{K}_t(\tilde{P}_{t+1} - P_{t|t})\tilde{K}_t^T. \tag{3.31}$$

3.3.5 Fitting state-space models

The state space model described by (3.1) and (3.2) often contains some unknown parameters such as the variances of the noises and the initial state, and the coefficients of the functions f and g. Such unknown parameters are included in a vector denoted by θ. Given the observations $Y_T = \{y_0, y_1, ..., y_T\}$ and inputs $U_T = \{u_0, u_1, ..., u_T\}$, the likelihood of the parameter θ of the model is written as

$$L(\theta) = p_y(y_0, ..., y_T, u_0, ..., u_T|\theta) = p_{y_0}(y_0) \prod_{t=0}^{T} p_{y_{t+1}}(y_{t+1}|Y_t, U_t, \theta), \tag{3.32}$$

and the log-likelihood is consequently written as

$$l(\theta) = \log p_{y_0}(y_0) + \sum_{t=0}^{T} \log p_{y_{t+1}}(y_{t+1}|Y_t, U_t, \theta). \tag{3.33}$$

Here $p_{y_0}(y)$ is calculated from $p_0(x_0)$ as

$$p_{y_0}(y_0) = \int_{-\infty}^{+\infty} p_{y_0}(y_0|x_0, \theta) dx_0, \tag{3.34}$$

and $p_{y_{t+1}}(y_{t+1}|Y_t, U_t, \theta)$ is calculated as

$$p_{y_{t+1}}(y_{t+1}|Y_t, U_t, \theta) = \int_{-\infty}^{+\infty} p_{y_{t+1}}(y_{t+1}|x_{t+1}) p_{x_{t+1}}(x_{t+1}|Y_t, U_t, \theta) dx_{t+1}. \tag{3.35}$$

If the system is linear and the noises are Gaussian, the log-likelihood can be written as

$$l(\theta) = -\{\frac{T\log 2\pi}{2} + \frac{1}{2}\sum_{t=1}^{T}\log(GP_{t|t-1}G^T + R) + \sum_{t=1}^{T}\frac{I_{y_t}^2}{2GP_{t|t-1}G^T + 2R}\}. \tag{3.36}$$

The maximum likelihood estimation $\hat{\theta}^*$ can be obtained by maximizing the log-likelihood. The maximum likelihood estimation may not be able to be obtained directly with the continuous probability density functions. *Kitagawa* [1996] discussed how to obtain the numerical approximation of the log-likelihood. The densities are approximated by many of the realizations from the corresponding distributions. Then, the approximation of the log-likelihood can be obtained. The maximum likelihood estimation can be calculated by maximizing the approximation of the log-likelihood.

Gradient descent method is one way to iteratively approach $\hat{\theta}^*$ from an initial $\hat{\theta}_0$. For the general case, since we use the discrete probability distribution to approximate the original continuous probability distribution in the particle filter, (3.35) can be approximately calculated by

$$p_{y_{t+1}}(y_{t+1} = g(x_{p,t}^{(i)}, u_t)|Y_t, U_t, \theta) = \frac{\alpha_t^i}{\alpha_t^1 + ... + \alpha_t^{N_\mathrm{p}}}. \tag{3.37}$$

Then, for a given $\hat{\theta}_k$, we can approximately calculate its log-likelihood $l(\hat{\theta}_k)$ by using (3.37) and (3.33). The estimation of parameter vector can be updated by

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \gamma\nabla\theta_{\mathrm{sd},k}. \tag{3.38}$$

Here, γ is a coefficient within (0,1). $\nabla\theta_{\mathrm{sd},k}$ denotes the steepest descent direction which is calculated as

$$\nabla\theta_{\mathrm{sd},k} = \arg\min_{\|v\|=1}\nabla l(\hat{\theta}_k)^T v \tag{3.39}$$

where $\nabla l(\hat{\theta}_k)$ denotes the gradient of log-likelihood at $\hat{\theta}_k$. We can add a small disturbance on each direction of $\hat{\theta}_k$ to get the approximate partial gradient on that direction and finally obtain $\nabla l(\hat{\theta}_k)$. γ can be obtained by exact line search or backtracking which refers to Chapter 9 of *Boyd and Vandenberghe* [2004]. Note that the gradient descent method will be very computation consuming in the high dimensional case:

1. The inference of state variables needs more samples and the update of the probability weight needs more computation time consequently;

2. The vector θ becomes huge and the process of obtaining the gradient needs the implement of particle filter again.

If the system is linear and the noises are white Gaussian, the computational burden is decreased compared to the general case in high-dimension case since the sampling is not required. Instead, only the matrix operation is needed.

An alternative way to optimize parameter vector θ is the Expectation-Maximization (EM) algorithm. The EM algorithm is an iterative parameter re-estimation procedure in which parameter θ is iteratively improved to maximaize the likelihood of the observed data $p(Y_T|U_T, \theta)$ in the presence of hidden variables X_T. For the case of stochastic dynamical systems described by SSMs, Y_T is the sequence of observations and U_T is the sequence of observed inputs, and X_T is the sequence of hidden variables as previously defined. The parameters of the model is defined by θ. Maximizing the likelihood as a function of θ is equivalent for maximizing the log-likelihood:

$$l(\theta) = \log p(Y_T|U_T, \theta) = \log \int_{X_T} p(X_T, Y_T|U_T, \theta) dX_T. \tag{3.40}$$

For any given distribution $Q(X_T)$ over the hidden variables, we can obtain a lower bound on l:

$$\begin{aligned} \log \int_{X_T} p(X_T, Y_T|U_T, \theta) dX_T &= \log \int_{X_T} q(X_T) \frac{p(X_T, Y_T|U_T, \theta)}{q(X_T)} dX_T \\ &\geq \int_{X_T} q(X_T) \log \frac{p(X_T, Y_T|U_T, \theta)}{q(X_T)} dX_T \\ &= \int_{X_T} q(X_T) \{\log p(X_T, Y_T|U_T, \theta) - \log q(X_T)\} dX_T \\ &= \mathcal{F}(q, \theta). \end{aligned} \tag{3.41}$$

Notice that the inequality (Jensen's inequality) in Eq. (3.41) can be proved by using the concavity of the log function. The EM algorithm alternates between maximizing $\mathcal{F}(q, \theta)$ with respect to the distribution q and the parameters θ, respectively, holding the other fixed [*Lange*, 2013]. Namely, starting from a initial parameters θ_0, we alternately apply:

$$\textbf{E-Step:} \quad q_{k+1} \leftarrow \arg\max_q \mathcal{F}(q, \theta_k), \tag{3.42}$$

$$\textbf{M-Step:} \quad \theta_{k+1} \leftarrow \arg\max_\theta \mathcal{F}(q_{k+1}, \theta). \tag{3.43}$$

In dynamical systems with hidden states, the E-Step corresponds exactly to solving the hidden state inference problem or the smoothing problem: inferring the hidden state trajectory given both the observations/inputs and the parameter values. Notice that the parameter values are given by the M-Step of the last iteration [*Ghahramani and Roweis*, 1999]. In the E-Step, a system identification problem is solved to update the parameter values by using the hidden state estimated from the E-Step.

3.4 Problem Formulation

State space models with nonlinear response which can be written as

$$x_{t+1} = Fx_t + Hu_t + v_t, \tag{3.44}$$

$$y_t = C + Gx_t + Ju_t + s(x_t) + w_t. \tag{3.45}$$

Here, f is linear function and g has linear part and also nonlinear part $s(x_t)$. Notice that any of the information about $s(x_t)$ is not available. $v_t \in \mathbb{R}^k$ and $w_t \in \mathbb{R}^d$ denote the k-dimension Gaussian white noise and d-dimension Gaussian white noise with covariance matrices Q and R, respectively. Moreover, the initial state x_0 also obeys Gaussian distribution. The unknown parameters in F, H, C, G, J, Q, R are gathered in a parameter vector denoted as θ_m.

In this case, we want to achieve model learning and hidden state inference based on the given system input set $U_T = \{u_0, u_1, .., u_T\}$ and observation set $Y_T = \{y_0, y_1, ..., y_T\}$. Model learning is to obtain the parameter vector θ_m and the nonlinear part $s(x_t)$. Notice that $s(x_t)$ is from a function space with a basis that consists of finite number of functions. Hidden state inference is to estimate x_t for every $t = 0, ..., T$. The problem is formally summarized in Problem 5.2.1.

Problem 3.4.1. Given the system input set $U_T = \{u_0, u_1, .., u_T\}$ and observation set $U_T = \{y_0, y_1, ..., y_T\}$, to obtain optimal inference

$$\{\hat{\theta}_m^*, \hat{s}^*(x_t)\} = \arg \max_{\hat{\theta}_m, \hat{s}(x_t)} \log p_{y_0}(y_0) + \sum_{t=0}^{T} \log p_{y_{t+1}}(y_{t+1}|Y_t, U_t, \hat{\theta}_m, \hat{s}(x_t)). \tag{3.46}$$

Here the objective function is the log-likelihood of $\{\hat{\theta}_m, \hat{s}(x_t)\}$ which can be denoted by $l(\hat{\theta}_m, \hat{s}(x_t))$. The likelihood is written by

$$L(\hat{\theta}_m, \hat{s}(x_t)) = p_y(y_0, ..., y_T|\hat{\theta}_m, \hat{s}(x_t)) = p_{y_0}(y_0) \prod_{t=0}^{T} p_{y_{t+1}}(y_{t+1}|Y_t, U_t, \hat{\theta}_m, \hat{s}(x_t)), \tag{3.47}$$

where $p_{y_0}(y)$ is calculated from $p_0(x_0)$ as

$$p_{y_0}(y_0) = \int_{-\infty}^{+\infty} p_{y_0}(y_0|x_0)dx_0, \tag{3.48}$$

and $p_{y_{t+1}}(y_{t+1}|Y_t, U_t, \hat{\theta}_m, \hat{s}(x_t))$ is calculated as

$$p_{y_{t+1}}(y_{t+1}|Y_t, U_t, \hat{\theta}_m, \hat{s}(x_t)) = \int_{-\infty}^{+\infty} p_{y_{t+1}}(y_{t+1}|x_{t+1}) p_{x_{t+1}}(x_{t+1}|Y_t, U_t, \hat{\theta}_m, \hat{s}(x_t)) dx_{t+1}. \tag{3.49}$$

The hidden state inference is included in Problem 5.2.1 with an implicit way. To achieve the maximal likelihood of observations, the probability distributions of the hidden states are necessary which should be inferred by the sequences of observed inputs and observations.

Remark 3.4.1. Problem 5.2.1 essentially aims at improving the linear state-space model by finding an approximation function of nonlinear residual in observation equation.

Assume $s(x_t)$ is continuous and then we could use neural network model to approximate $s(x_t)$. Since $s(x_t)$ is from $\mathbb{R}^k \to \mathbb{R}^d$, we can split $s(x_t)$ as

$$s(x_t) = [s_1(x_t), ..., s_d(x_t)]^T. \tag{3.50}$$

The non-parametric model used to approximate $s(x_t)$ is defined as $\hat{s}_{\mathrm{n}}(x_t)$. We can also split

$$\hat{s}_{\mathrm{n}}(x_t) = [\hat{s}_{\mathrm{n},1}(x_t), ..., \hat{s}_{\mathrm{n},d}(x_t)]^T. \tag{3.51}$$

Each $\hat{s}_{\mathrm{n},j}(x_t), j \in 1, ..., d$ can be written as

$$\hat{s}_{\mathrm{n},j}(x_t) = \sum_{i=1}^{N_{\mathrm{n}}} \beta_{ij} h(x_t, a_{ij}, b_{ij}) \tag{3.52}$$

where N_{n} denotes the number of hidden nodes of the non-parameter model, $h(\cdot)$ denotes the activation function, and β_{ij} denotes the weight connecting the $i-$th hidden node and the output, $a_{ij} = [a_{ij,1}, ..., a_{ij,k}]$ represents the weight vector towards x_t, and b_{ij} is the scalar threshold of the $i-$th hidden node. Note that the candidate for activation function $h(\cdot)$ can be sigmoid function

$$h(x_t, a_{ij}, b_{ij}) = \frac{1}{1 + \exp(-a_{ij} x_t - b_{ij})}, \tag{3.53}$$

or Fourier function

$$h(x_t, a_{ij}, b_{ij}) = \sin(a_{ij} x_t + b_{ij}), \tag{3.54}$$

or Gaussian function

$$h(x_t, a_{ij}, b_{ij}) = \exp(-b_{ij} \|x_t - a_{ij}\|^2), \tag{3.55}$$

or multiquadrics function

$$h(x_t, a_{ij}, b_{ij}) = (\|x_t - a_{ij}\|^2 + b_{ij}^2)^{1/2}, \tag{3.56}$$

or other functions such as Hardlimit function. In our thesis, we choose sigmoid function as our activation function because its derivative is easily to be obtained.

$\beta_j = [\beta_{1j}, .., \beta_{N_{\mathrm{n}}j}]^T$ denotes the weight vector connecting the hidden nodes and the output. According to the universal approximation theorem [*Cybenko*, 1989, *Selmic and Lewis*, 2002], $\forall \epsilon_s > 0, \exists N_{\mathrm{p}} \in \mathbb{N}^+, \beta_j, b_j \in \mathbb{R}^{N_{\mathrm{n}}}, a_j \in \mathbb{R}^{N_{\mathrm{n}} \times k}$, and $\exists h(\cdot)$, such that

$$\|\hat{s}_{\mathrm{n},j} - s_j(x_t)\| \leq \epsilon_s, \forall x_t \in \mathbb{R}^k. \tag{3.57}$$

If we use neural network to approximate the nonlinear response, the problem to choose $\hat{s}(x_t)$ is reduced to estimate $\beta_j, a_j, b_j, \forall j \in 1, ..., d$. All the unknown parameters in β_j, a_j, b_j can be included in a parameter vector $\theta_{s,j} \in \mathbb{R}^{N_{\theta_{s,j}}}$ where $N_{\theta_{s,j}} = N_{\mathrm{n}} + N_{\mathrm{n}} + N_{\mathrm{n}} \times k$. Define the parameter vector for $\hat{s}_{\mathrm{n}}(x_t)$ by

$$\theta_s = [\theta_{s,1}^T, ..., \theta_{s,d}^T]^T. \tag{3.58}$$

The estimation of θ_s is denoted by $\hat{\theta}_s$. Thus, the Problem 5.2.1 can be reformed by choosing $\hat{\theta}_s$ instead of $\hat{s}(x_t)$.

Problem 3.4.2. Given the system input set $U_T = \{u_0, u_1, .., u_T\}$ and observation set $Y_T = \{y_0, y_1, ..., y_T\}$, to obtain $\hat{\theta}_m^*$ and $\hat{\theta}_s^*$ by solving

$$\max_{\hat{\theta}_m, \hat{\theta}_s} \log p_{y_0}(y_0) + \sum_{t=1}^{T} \log p_{y_{t+1}}(y_{t+1}|Y_t, U_t, \hat{\theta}_m, \hat{\theta}_s). \tag{3.59}$$

3.5 Proposed Model Improvement Algorithms

The framework of the proposed algorithm is shown in Fig. 3.2 and is also summarized as follows:

- **Kalman filter (Expectation).** Estimate the state trajectory via Kalman filter based on the sequences of linear response and inputs.
- **Reconstruction (Maximization R).** Identify the nonlinear response by fitting the parameters in the neural network model based on the residuals of the linear model. We use MR as the abbreviation for Maximization R.

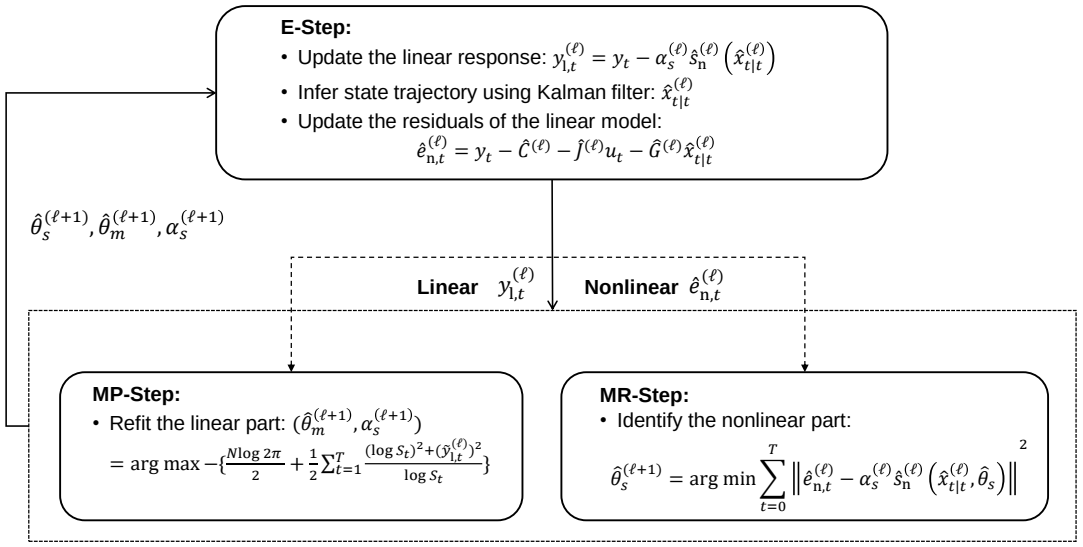

FIGURE 3.2: The framework of the proposed algorithm.

- **Parametrization (Maximization P).** Identify the parameters in the linear part of the SSM based on the sequences of linear response, inputs, estimated states. We use MP as the abbreviation for Maximization P.

The proposed algorithm is summarized as:

1. **(Initialization.)** Initialize iteration number $\ell = 0$, parameter decision $\hat{\theta}_m^{(\ell)}, \hat{\theta}_s^{(\ell)} = \mathbf{0}$, and relaxation factor $\alpha^{(\ell)} = 1$;

2. **(E-Step.)** Calcualte $\hat{s}_{\mathrm{n}}^{(\ell)}(\hat{x}_{t|t}^{(\ell)})$ based on $\hat{\theta}_s^{(\ell)}$ and obtain linear part of output $y_{l,t}^{(\ell)}$ as
$$y_{l,t}^{(\ell)} = y_t - \alpha^{(\ell)} \hat{s}_{\mathrm{n}}^{(\ell)}(\hat{x}_{t|t}) - \hat{C}^{(\ell)} - \hat{J}^{(\ell)} u_t. \tag{3.60}$$
(For $\ell = 0$, use $\hat{s}_{\mathrm{n}}^{(\ell)}(\hat{x}_{t|t}^{(\ell)}) = \mathbf{0}$.)

3. **(E-Step.)** Implement Kalman filter based on $\hat{\theta}_m^{(\ell)}$, $U_T = \{u_0, u_1, .., u_T\}$, $Y_{l,T}^{(\ell)} = \{y_{l,0}^{(\ell)}, y_{l,1}^{(\ell)}, ..., y_{l,T}^{(\ell)}\}$ and obtain the update of state estimation $\hat{x}_{t|t}^{(\ell)}$, nonlinear residual $\hat{e}_{\mathrm{nl},t}^{(\ell)} = y_t - \hat{C}^{(\ell)} - \hat{J}^{(\ell)} u_t - \hat{G}^{(\ell)} \hat{x}_{t|t}^{(\ell)}$ for all t;

4. **(MR-Step)** Update the parameter decision $\hat{\theta}_s$ by
$$\hat{\theta}_s^{(\ell+1)} = \min_{\hat{\theta}_s} \sum_t \|\hat{e}_{\mathrm{nl},t}^{(\ell)} - \alpha^{(\ell)} \hat{s}_{\mathrm{n}}(\hat{x}_{t|t}^{(\ell)}, \hat{\theta}_s)\|^2. \tag{3.61}$$

5. **(MP-Step.)** Update $\hat{\theta}_m, \quad \alpha$ by

$$\{\hat{\theta}_m^{(\ell+1)}, \alpha^{(\ell+1)}\} = \max_{\hat{\theta}_m, \alpha} \log p_{y_0}(y_0) + \sum_{t=1}^{T} \log p_{y_t}(y_t|\hat{\theta}_m, \alpha, \hat{\theta}_s^{(\ell+1)}, Y_{t-1}, U_{t-1}). \quad (3.62)$$

6. **(Termination.)** Set $\ell = \ell + 1$. If $\|\hat{\theta}_m^{(\ell)} - \hat{\theta}_m^{(\ell-1)}\| + \|\hat{\theta}_s^{(\ell)} - \hat{\theta}_s^{(\ell-1)}\| < \epsilon$ where ϵ is a sufficiently small positive number, stop and output $\hat{\theta}_m^* = \hat{\theta}_m^{(\ell)}, \hat{\theta}_s^* = \hat{\theta}_s^{(\ell)}, \alpha^* = \alpha^{(\ell)}$. Otherwise, go back to step 2.

The gradient descent method for updating $\hat{\theta}_s^{\ell}$ in (3.61) is summarized as follows:

1. **Initialization.** Randomly assign $\hat{\theta}_{s,0}^*$ and set $\ell_{\mathrm{g}} = 0$;

2. **Gradient calculation.**

$$\nabla\hat{\theta}_{s,\ell_{\mathrm{g}}}^* = \sum_{t=0}^{T} \frac{\partial \hat{s}_n^{(\ell)}(\hat{x}_{t|t}^{(\ell)}, \hat{\theta}_s)}{\partial \hat{\theta}_s} (\hat{s}_n^{(\ell)}(\hat{x}_{t|t}^{(\ell)}, \hat{\theta}_s) - \frac{\hat{e}_{n,t}^{(\ell)}}{\alpha_s^{(\ell)}})|_{\hat{\theta}_s = \hat{\theta}_{s,\ell_{\mathrm{g}}}^*}; \quad (3.63)$$

3. **Updata.** $\hat{\theta}_{s,\ell_{\mathrm{g}}+1}^* = \hat{\theta}_{s,\ell_{\mathrm{g}}}^* - \gamma_{\ell_{\mathrm{g}}} \nabla\hat{\theta}_{s,\ell_{\mathrm{g}}}^*$ where $\gamma_{\ell_{\mathrm{g}}} \in (0, 1)$;

4. **Check Terminal Condition.** If $\|\hat{\theta}_{s,\ell_{\mathrm{g}}+1}^* - \hat{\theta}_{s,\ell_{\mathrm{g}}}^*\| < \epsilon_{\theta_s}$ where ϵ_{θ_s} is a sufficiently small positive number, terminate and output $\hat{\theta}_{s,\ell_{\mathrm{g}}+1}^*$ as $\hat{\theta}_s^{(\ell+1)}$. Otherwise, set $\ell_{\mathrm{g}} = \ell_{\mathrm{g}} + 1$ and go back to step 2.

We can solve Eq. (3.61) with less computation burden by applying Extreme Learning Machine (ELM) algorithm [*Huang et al.*, 2006]. The details about ELM algorithm is summarized in Appendix A. The ELM algorithm for updating $\hat{\theta}_s^{\ell}$ is summarized as follows:

1. Randomly assign a_{ij}, b_{ij} for all $i = 1, ..., N_{\mathrm{n}}$ and $j = 1, ..., d$;

2. Calculate the hidden layer output matrix $H_{s,j}, \forall j = 1, ..., d$ as

$$\Phi_{s,j} = \begin{bmatrix} h(\hat{x}_{0|0}, a_{1j}, b_{1j}) & ... & h(\hat{x}_{0|0}, a_{ij}, b_{ij}) & ... & h(\hat{x}_{0|0}, a_{N_{\mathrm{n}}j}, b_{N_{\mathrm{n}}j}) \\ ... & .. & ... & ... & ... \\ h(\hat{x}_{t|t}, a_{1j}, b_{1j}) & ... & h(\hat{x}_{t|t}, a_{ij}, b_{ij}) & ... & h(\hat{x}_{t|t}, a_{N_{\mathrm{n}}j}, b_{N_{\mathrm{n}}j}) \\ ... & ... & ... & ... & ... \\ h(\hat{x}_{T|T}, a_{1j}, b_{1j}) & ... & h(\hat{x}_{T|T}, a_{ij}, b_{ij}) & ... & h(\hat{x}_{T|T}, a_{N_{\mathrm{n}}j}, b_{N_{\mathrm{n}}j}) \end{bmatrix} \quad (3.64)$$

   where the activation function $h(\cdot)$ is chosen as sigmoid function;

3. Calculate β_j by

$$\beta_j = (\Phi_{s,j}^T \Phi_{s,j})^{-1} \Phi_{s,j}^T \hat{E}_T \tag{3.65}$$

where $\hat{E}_T = [\hat{e}_{\text{nl},0}, ..., \hat{e}_{\text{nl},T}]^T$.

Besides, the algorithm to obtain $\hat{\theta}_m^*$ which includes F, H, G, J, C and α for MP-Step is summarized as follows:

1. For $i = 1, ..., k$, obtain a data matrix

$$\Phi_{f,i} = \begin{bmatrix} \tilde{x}_{0,1} & ... & \tilde{x}_{0,k} & u_{0,1} & ... & u_{0,c} \\ .... & ... & ... & ... & ... & ... \\ \tilde{x}_{t,1} & ... & \tilde{x}_{t,k} & u_{t,1} & ... & u_{t,c} \\ ... & ... & ... & ... & ... & ... \\ \tilde{x}_{T-1,1} & ... & \tilde{x}_{T-1,k} & u_{T-1,1} & ... & u_{T-1,c} \end{bmatrix} \tag{3.66}$$

and output vector

$$O_{f,i} = \begin{bmatrix} \tilde{x}_{1,i} \\ ... \\ \tilde{x}_{t,i} \\ ... \\ \tilde{x}_{T,i} \end{bmatrix}. \tag{3.67}$$

Then, calculate $\hat{\theta}_{f,i}$ by

$$\hat{\theta}_{f,i} = (\Phi_{f,i}^T \Phi_{f,i})^{-1} \Phi_{f,i}^T O_{f,i}. \tag{3.68}$$

and obtain $\hat{F}$ and $\hat{H}$ by

$$\hat{F} = \begin{bmatrix} \hat{\theta}_{f,1}^T(1:k) \\ ... \\ \hat{\theta}_{f,i}^T(1:k) \\ ... \\ \hat{\theta}_{f,k}^T(1:k) \end{bmatrix}, \tag{3.69}$$

$$\hat{H} = \begin{bmatrix} \hat{\theta}_{f,1}^T(k+1:k+c) \\ ... \\ \hat{\theta}_{f,i}^T(k+1:k+c) \\ ... \\ \hat{\theta}_{f,k}^T(k+1:k+c) \end{bmatrix}. \tag{3.70}$$

2. For $i = 1, ..., d$, obtain a data matrix

$$\Phi_{g,i} = \begin{bmatrix} \tilde{x}_{0,1} & ... & \tilde{x}_{0,k} & u_{0,1} & ... & u_{0,c} & 1 & \hat{s}_{\mathrm{n},i}(\hat{x}_{0|T}) \\ .... & ... & ... & ... & ... & ... & 1 & ... \\ \tilde{x}_{t,1} & ... & \tilde{x}_{t,k} & u_{t,1} & ... & u_{t,c} & 1 & \hat{s}_{\mathrm{n},i}(\hat{x}_{t|T}) \\ ... & ... & ... & ... & ... & ... & 1 & ... \\ \tilde{x}_{T,1} & ... & \tilde{x}_{T,k} & u_{T,1} & ... & u_{T,c} & 1 & \hat{s}_{\mathrm{n},i}(\hat{x}_{T|T}) \end{bmatrix} \tag{3.71}$$

and output vector

$$O_{g,i} = \begin{bmatrix} y_{0,i} \\ ... \\ y_{t,i} \\ ... \\ y_{T,i} \end{bmatrix}. \tag{3.72}$$

Then, calculate $\hat{\theta}_{g,i}$ by

$$\hat{\theta}_{g,i} = (\Phi_{g,i}^T \Phi_{g,i})^{-1} \Phi_{g,i}^T O_{g,i}. \tag{3.73}$$

and obtain parameters by

$$\hat{G} = \begin{bmatrix} \hat{\theta}_{g,1}^T(1:k) \\ ... \\ \hat{\theta}_{g,i}^T(1:k) \\ ... \\ \hat{\theta}_{g,d}^T(1:k) \end{bmatrix}. \tag{3.74}$$

$$\hat{J} = \begin{bmatrix} \hat{\theta}_{g,1}^T(k+1:k+c) \\ ... \\ \hat{\theta}_{g,i}^T(k+1:k+c) \\ ... \\ \hat{\theta}_{g,d}^T(k+1:k+c) \end{bmatrix}. \tag{3.75}$$

$$\hat{C} = \begin{bmatrix} \hat{\theta}_{g,1}^T(k+c+1) \\ ... \\ \hat{\theta}_{g,i}^T(k+c+1) \\ ... \\ \hat{\theta}_{g,d}^T(k+c+1) \end{bmatrix}. \tag{3.76}$$

$$\hat{\alpha} = \begin{bmatrix} \hat{\theta}_{g,1}^T(k+c+2) \\ ... \\ \hat{\theta}_{g,i}^T(k+c+2) \\ ... \\ \hat{\theta}_{g,d}^T(k+c+2) \end{bmatrix}. \tag{3.77}$$

3. Calculate $\hat{Q}$ as

$$\hat{Q} = \frac{1}{T-1}\sum_{t=0}^{T-1}(e_{x,t} - \bar{e}_x)(e_{x,t} - \bar{e}_x)^T \tag{3.78}$$

where $e_{x,t} = \tilde{x}_{t+1} - \hat{F}\tilde{x}_t - \hat{H}u_t$ and $\bar{e}_x = \frac{1}{T}\sum_{t=0}^{T-1} e_{x,t}$.

5. Calculate $\hat{R}$ as

$$\hat{R} = \frac{1}{T-1}\sum_{t=0}^{T-1}(e_{y,t} - \bar{e}_y)(e_{y,t} - \bar{e}_y)^T \tag{3.79}$$

where $e_{y,t} = y_t - \hat{C} - \hat{G}\tilde{x}_t - \hat{J}u_t - \hat{s}_{\mathrm{n}}(\tilde{x}_t)$ and $\bar{e}_y = \frac{1}{T}\sum_{t=0}^{T-1} e_{y,t}$.

Remark 3.5.1. Our proposed algorithm integrates the residual analysis to improve the linear state-space model. Even though the information of $s(x_t)$ is unknown, we can implement universal approximation to approximate $s(x_t)$ by using the residuals of the linear state-space model.

With the trained $\hat{\theta}_m^*$, $\hat{\theta}_s^*$, and α^*, we modify the Kalman filter into Decimating Kalman filter for state space models with nonlinear response, which repeats the following steps for $t = 1, ..., T$:

a) Calculate the predict mean of state

$$\mathbf{E}\{x_t|Y_{t-1}, U_{t-1}\} = \hat{x}_{t|t-1} = \hat{F}\hat{x}_{t-1|t-1} + \hat{H}u_{t-1}; \tag{3.80}$$

b) Calculate the predict covariance of state

$$P_{t|t-1} = \hat{F}P_{t-1|t-1}\hat{F}^T + \hat{Q}; \tag{3.81}$$

c) Calculate the predict innovation

$$I_{y_t} = y_t - \mathbb{E}\{y_t|Y_{t-1}, U_{t-1}\} \tag{3.82}$$

where $\mathbb{E}\{y_t|Y_{t-1}, U_{t-1}\} = \hat{C} + \hat{G}\hat{x}_{t|t-1} + \hat{J}u_{t-1} + \alpha^*\hat{s}(\hat{x}_{t|t-1}, \hat{\theta}_s^*)$;

d) Calculate the feedback gain

$$K_t = P_{t|t-1}\hat{G}^T(\hat{G}P_{t|t-1}\hat{G}^T + \hat{R})^{-1}; \tag{3.83}$$

e) Calculate the filter mean of state

$$\mathbb{E}\{x_t|Y_t, U_t\} = \hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t I_{y_t} \tag{3.84}$$

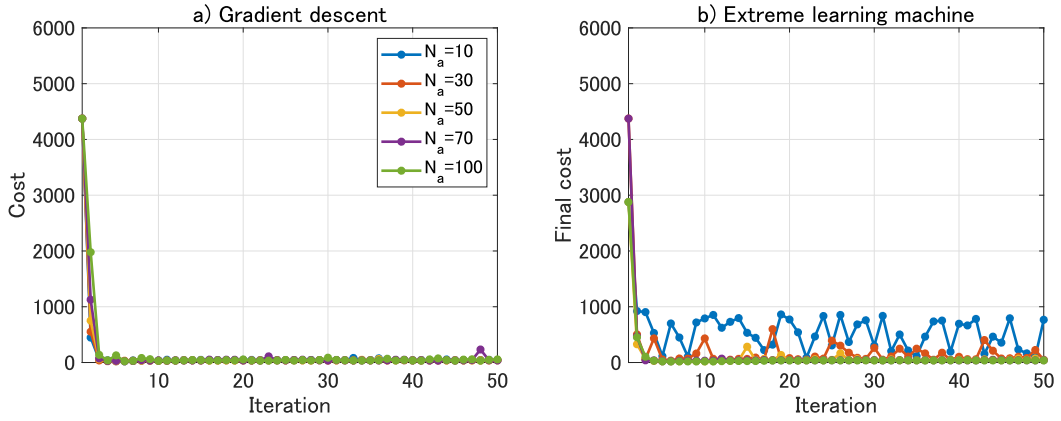

FIGURE 3.3: Results of cost: a) cost evolution by gradient descent method; b) cost evolution by ELM algorithm.

and obtain the filter covariance of state

$$P_{t|t} = P_{t|t-1} - K_t \hat{G} P_{t|t-1}. \tag{3.85}$$

The only difference is that the nonlinear residual is included in the predict innovation. The rest is the same.

3.6 Numerical Examples

3.6.1 Model for numerical examples

In the numerical example, we consider the following stochastic dynamical system

$$x(t+1) = 0.9x(t) + u(t) + Q_{\text{Num}}\mathcal{N}(0,1), \tag{3.86}$$

$$y(t) = x(t) + \frac{5\sin(x(t))}{x(t)} + R_{\text{Num}}\mathcal{N}(0,1), \tag{3.87}$$

$$u(t) = 3\cos(0.2t). \tag{3.88}$$

In the simulation, Q_{Num} is fixed as 0.1. R_{Num} takes different values: 0.1, 0.2, 0.3, 0.5, 1. We generated two groups of data for different R_{Num}. Each group has 500 time series data. One group is used for training and another is used for testing the trained model.

3.6.2 About number of activation functions

First, numerical simulations have been conducted to check how the number of activation functions influences the fit performance. Besides, the comparison of using gradient

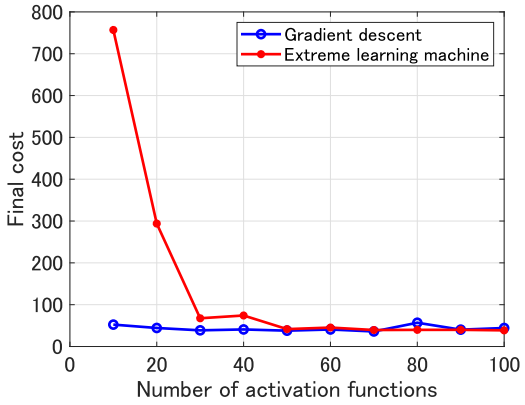

FIGURE 3.4: Final cost comparison between gradient descent method and ELM algorithm.

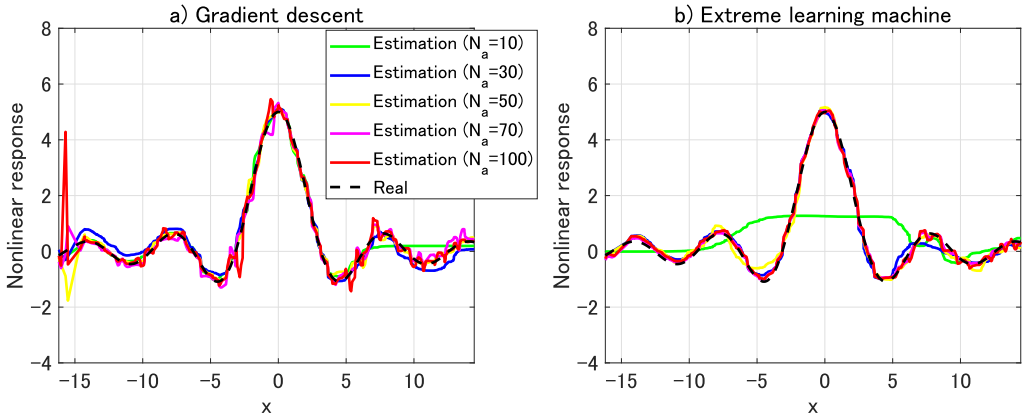

FIGURE 3.5: Results of nonlinear response fitting: a) gradient descent method; b) ELM algorithm.

descent method to learn the nonlinear part and using ELM algorithm to learn the nonlinear part are given in this validation. In this validation, we only use the training data and the testing data with $R_{\text{Num}} = 0.1$. The activation function adopts sigmoid function.

Fig. 3.3 shows the cost evolution by gradient descent method and ELM algorithm. The cost is denoted by the minus of $l(\theta)$ in (3.36) where I_{y_t} is the replaced by $I_{y_{l,t}}$ which is the innovation for the linear part. Final costs for different numbers of activation functions are summarized in Fig. 3.4. When the number of activation functions is less than 40, gradient descent method shows a much better fitting performance. When the number of activation functions surpluses 50, ELM algorithm shows the same performance as the gradient descent method. The gradient descent method optimizes a_{ij} and b_{ij} while ELM algorithm randomly assigns a_{ij} and b_{ij}. When the number of activation function is larger enough, the probability of getting good a_{ij} and b_{ij} by random assignment increases. We do not need every a_{ij} and b_{ij} is good in ELM algorithm. If we have more random assignments, the number of good a_{ij} and b_{ij} assignments will also increase. The fitting

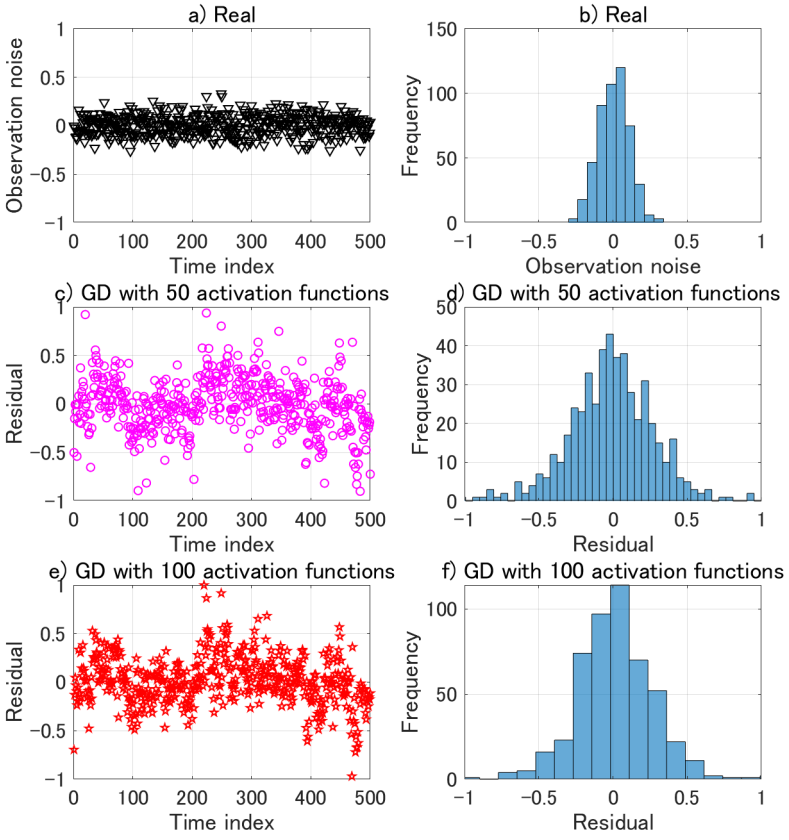

FIGURE 3.6: Predictive residual by gradient descent method.

plots of the nonlinear response are summarized in Fig. 3.5 which corresponds to the results of cost performance. Besides, Fig. 3.6 and 3.7 show the predictive residuals of gradient descent method and ELM algorithm. The performance of both methods are similar.

Fig. 3.8 shows the training times of gradient descent method and ELM algorithm for 200 iterations in the cases of different activation function numbers. The training time is much less when using ELM algorithm.

According to the above analysis, the following conclusions are obtained:

- For this numerical example, 100 activation functions are enough;
- ELM algorithm can achieve nearly the same performance as gradient descent method with much less computation time.

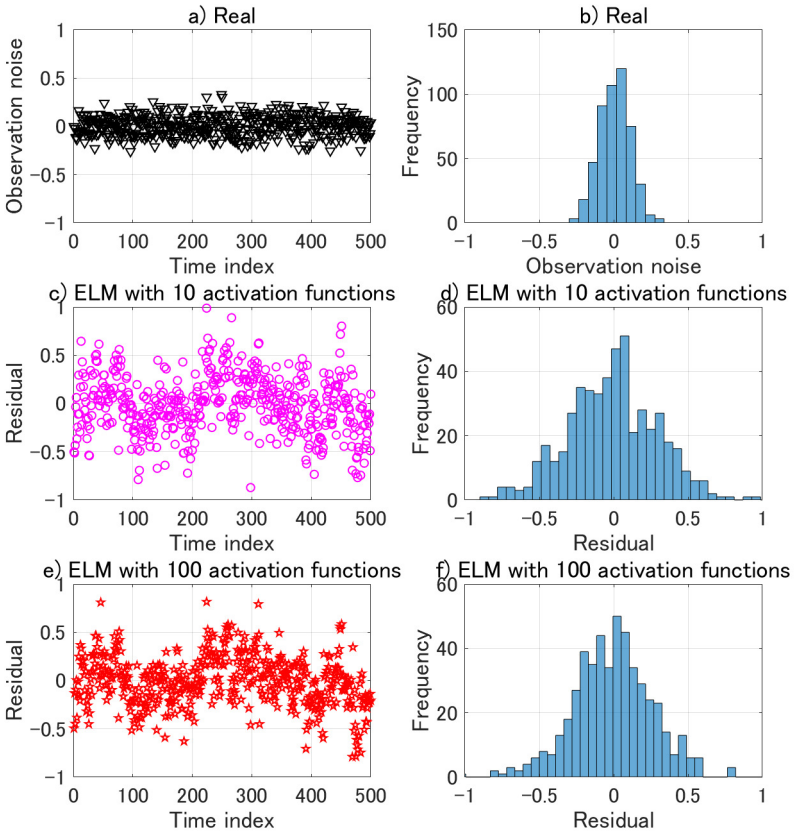

FIGURE 3.7: Predictive residual by ELM algorithm.

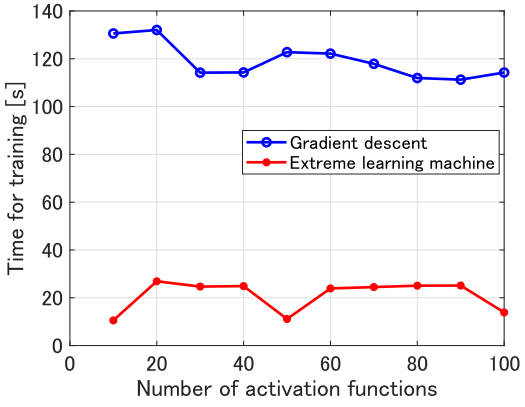

FIGURE 3.8: Training time comparison between gradient descent method and ELM algorithm.

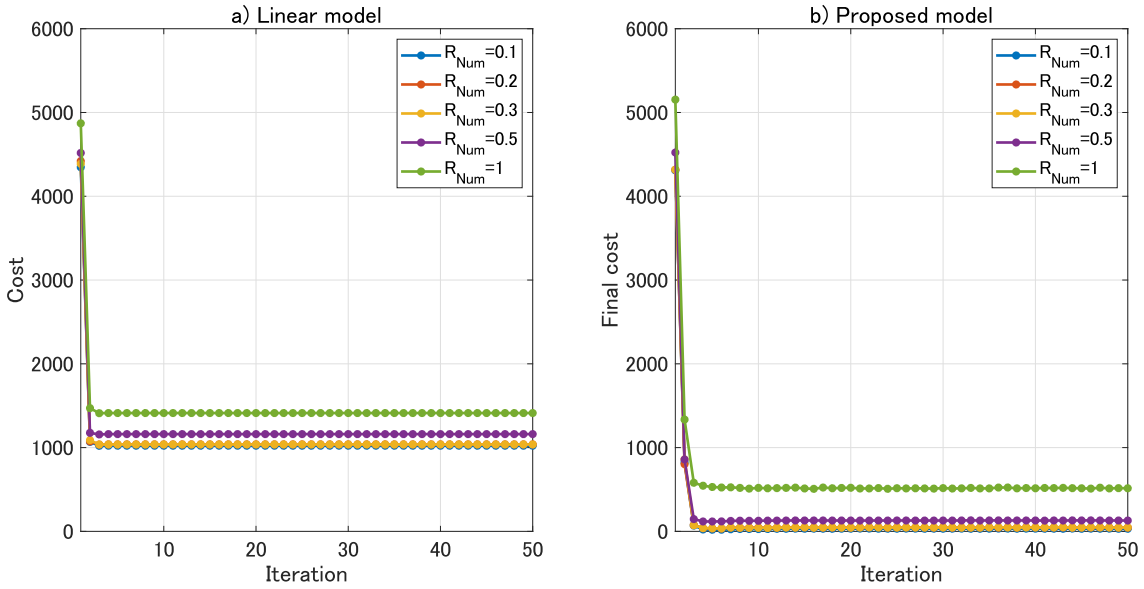

FIGURE 3.9: Results of cost: a) cost evolution with linear model; b) cost evolution with proposed model.

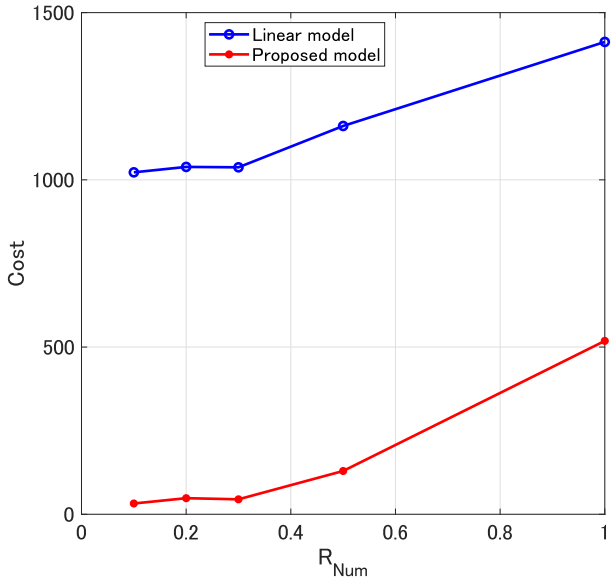

FIGURE 3.10: Final cost comparison.

3.6.3 Comparison with linear model

Fig. 3.9 and 3.10 show the results of cost in the training process which is $\sum_t (y_t - \hat{y}_t)^2$. The proposed model shows better performance than linear model. Moreover, if R_{Num} increases, the cost converges to a larger value. For the proposed method, the cost converges to nearly 500 when $R_{\text{Num}} = 1$. If we take the mean value of the cost which is the Mean Square Error, it fits to the variance of y_t: R_{Num}. The rest cases also satisfy this.

Besides, Fig. 3.11 shows the results of parameter identification. The parameters converge to the real values if the proposed model is applied. However, if we uses linear model, the converged parameter estimations deviate from the real values.

Fig. 3.12 shows the results of state estimation in the testing. The proposed decimating Kalman filter gives more accurate state estimation than the linear Kalman filter. The MSE of the state estimation is not affected by R_{Num}. Fig. 3.13 shows the results of predictive residual. The predictive residual was decreased by the decimating Kalman filter. Moreover, compared to the predictive residual by the linear Kalman filter, the predictive residual by the proposed decimating Kalamn filter is more close to the white Gaussian noise.

A more intuitive data virtualization of residual observation and approximation are given in Fig. 3.14. Fig. 3.14 a) shows the relation between state and predictive residual by using Kalman filter based on linear model. As R_{Num} increases, the variance of the predictive residual gets larger while the means converges at the same place where the real nonlinear response locates. Fig. 3.14 b) shows the relation between state and residual observations $y_t - \hat{x}_{t|t-1}$ in the decimating Kalman filter, namely, the remained value after decimating the estimated nonlinear response. Fig. 3.14 c) gives the comparisons between the estimated nonlinear responses and real nonlinear responses. The proposed method perfectly decimated the nonlinear response and then ensures the accuracy of linear Kalman filter on the linear part.

3.7 Application Case Study

The State of Charge (SoC) is defined as the ratio of the remaining capacity to the maximum available capacity which can be expressed as follows:

$$SoC_\tau = SoC_{\tau_0} + \frac{\eta_c}{C_n} \int_{\tau_0}^{\tau} i_{\tau'} d\tau' \tag{3.89}$$

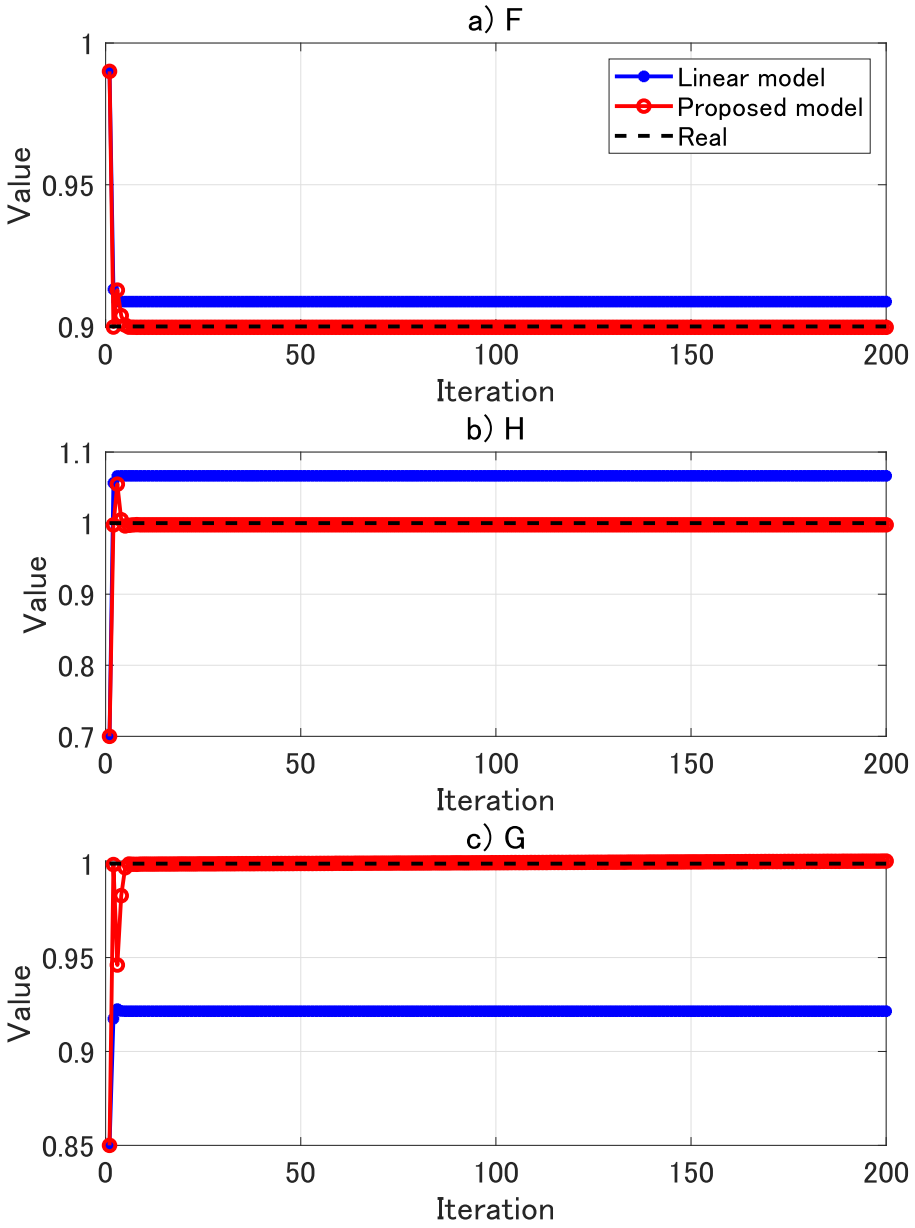

FIGURE 3.11: Results of parameter identification: a) F; b) H; c) G.

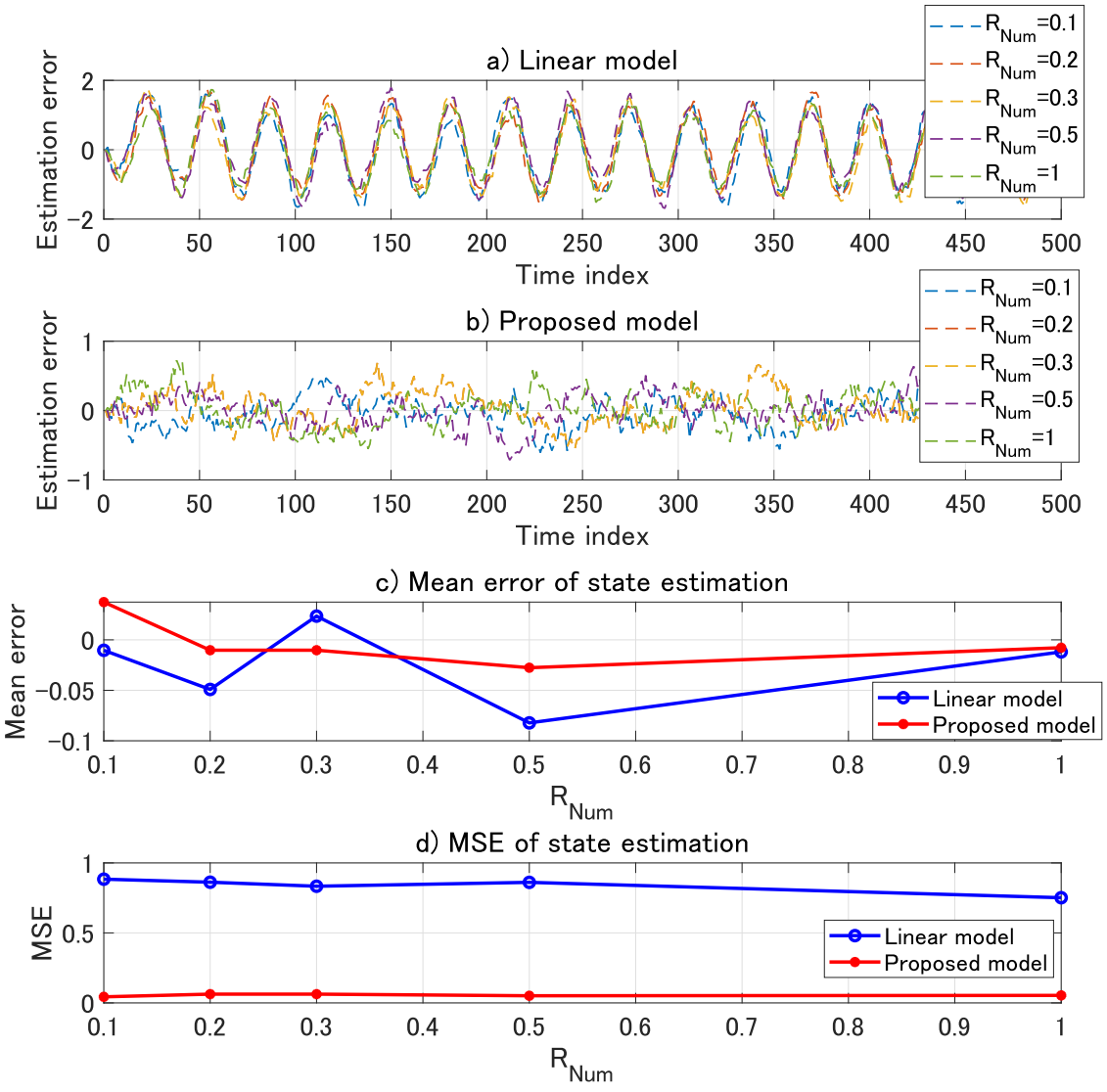

FIGURE 3.12: Results of state estimation error: a) linear model; b) proposed model; c) mean error comparison d) MSE comparison.

where SoC_τ, SoC_{τ_0} are the value of SoC at time τ, τ_0, respectively. $i_{\tau'}$ represents the charging current at time τ', η_c is the coulombic efficiency, C_n is the total available capacity of the battery.

Based on the dynamic characteristics and working principles of the battery, the equivalent circuit model is developed by using resistors, capacitors, and voltage sources to form a circuit network. Fig. 3.15 shows the schematic diagram of an equivalent circuit model with RC networks. As can be seen from the figure, the U_{OCV} is used to denote the battery voltage source. R_o represents the internal ohmic resistance. E_e and C_e denote the dynamic characteristics include polarization, diffusion, hysteresis and so on.

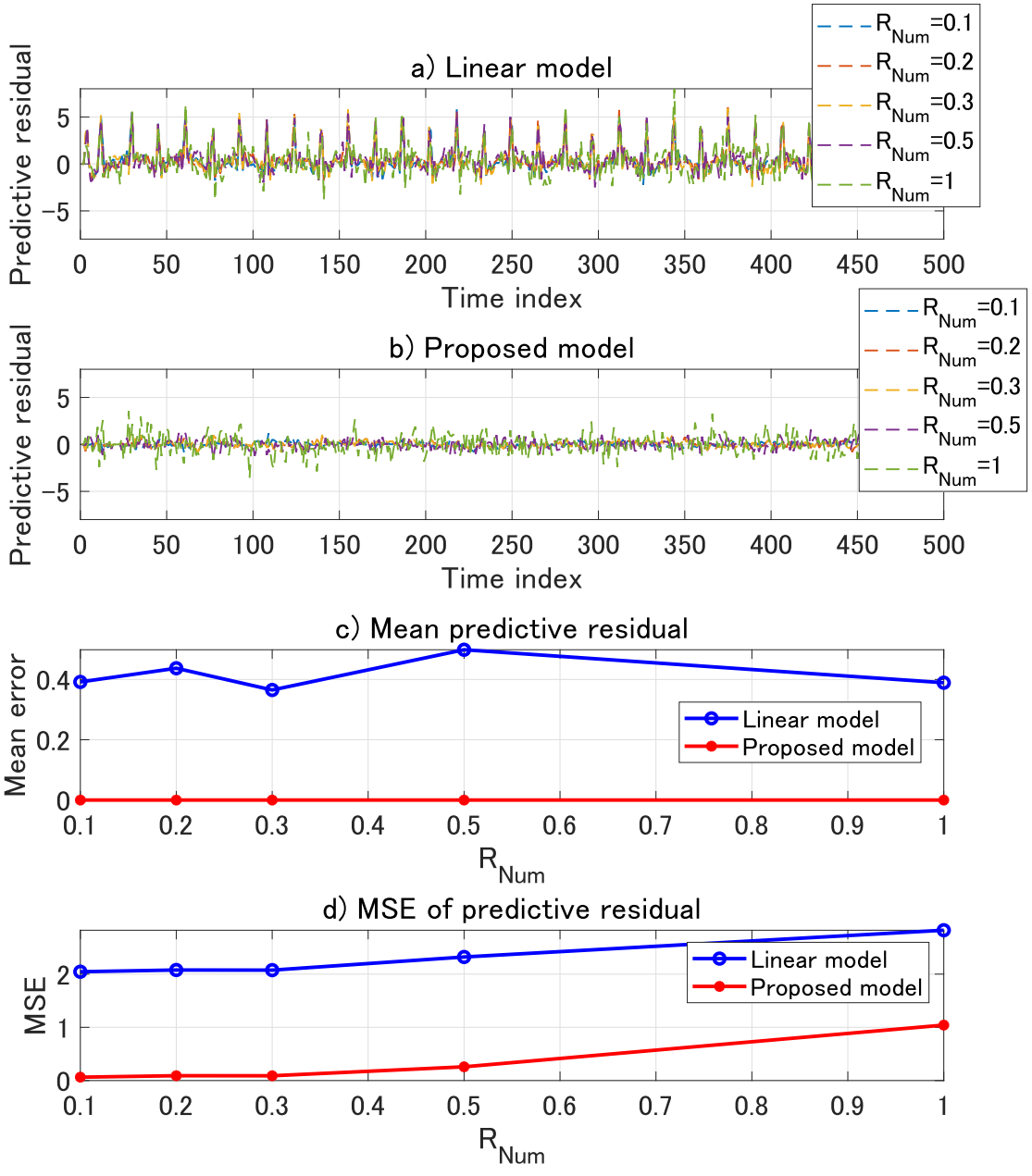

FIGURE 3.13: Results of predictive residual: a) linear model; b) proposed model; c) mean predictive residual comparison d) predictive residual's MSE comparison.

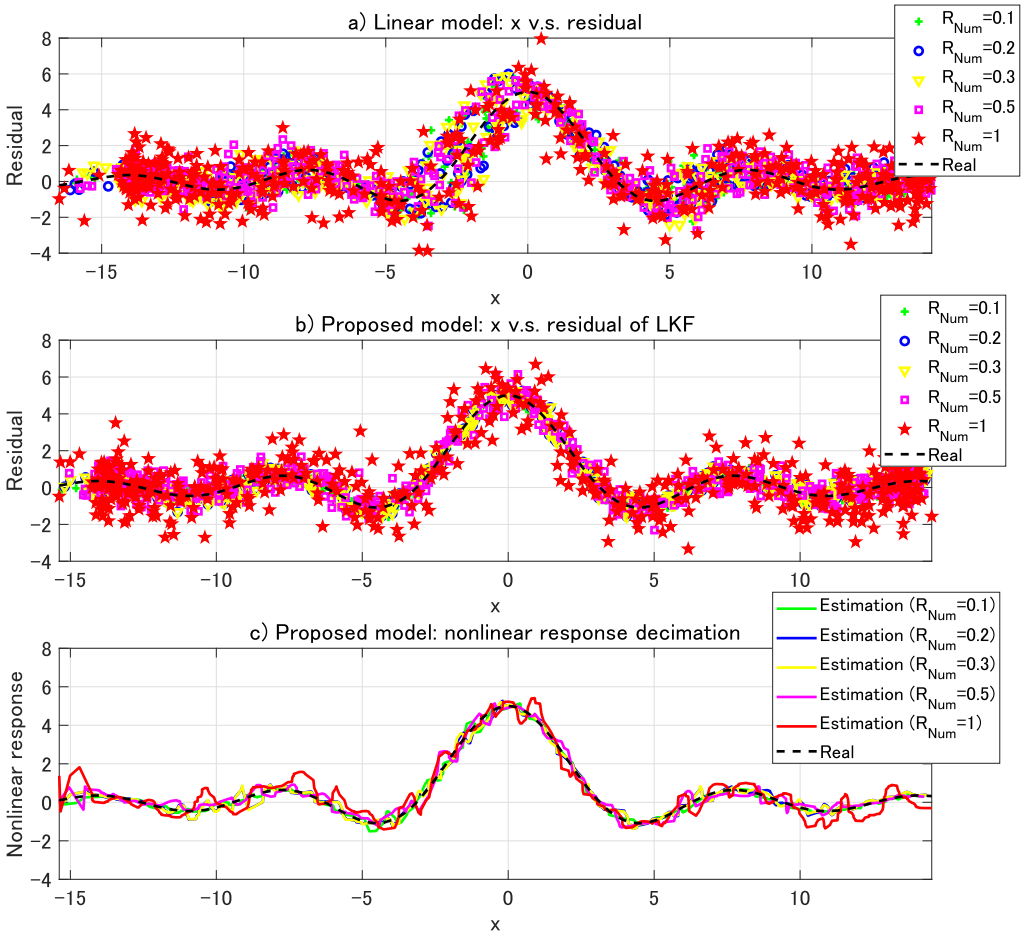

FIGURE 3.14: Residual observation and approximation: a) residual observation: linear model; b) LKF residual observation: proposed model; c) residual approximation by proposed method.

The electrical behavior of the RC model can be expressed as follows:

$$\dot{U}_{e,\tau} = -\frac{U_{e,\tau}}{R_e C_e} + \frac{i_\tau}{C_e} \tag{3.90}$$

$$U(\tau) = U_{OCV}(\tau) + i(\tau)R_o + U_e(\tau) \tag{3.91}$$

where i_τ is the charging current, U_τ is the terminal voltage and $U_{e,\tau}$ represents the voltage of the RC circuit.

The battery OCV is related to SoC and can be written as

$$U_{OCV,\tau} = U_{OCV,0} + K_0 SoC_\tau + s(SoC_\tau) \tag{3.92}$$

where $U_{OCV,0} + K_0 SoC_\tau$ is the linear part of the open-circuit voltage of a battery cell, and $s(SoC_\tau)$ is the nonlinear part of the open-circuit voltage of battery cell. Notice that the strucutre and information of $s(SoC_\tau)$ is not available.

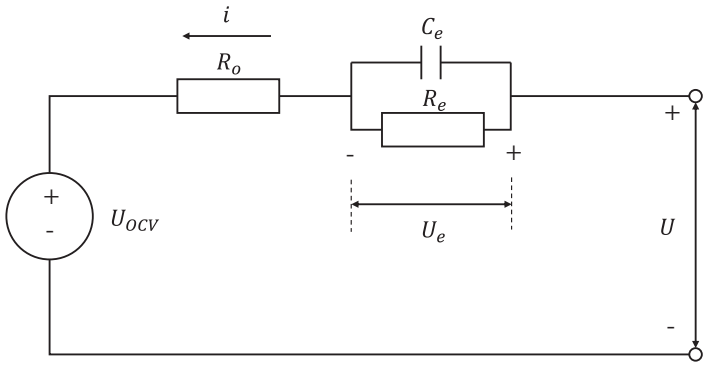

FIGURE 3.15: Schematic diagram of the RC equivalent circuit model.

To perform the model based SoC estimation, a discrete state space to describe the battery model is required. The discretization form of battery state-space equation for SoC estimation can be written as:

$$\begin{bmatrix} U_{e,t+1} \\ SoC_{t+1} \end{bmatrix} = \begin{bmatrix} e^{-\frac{\Delta\tau}{R_e C_e}} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} U_{e,t} \\ SoC_t \end{bmatrix} + \begin{bmatrix} 1 - e^{-\frac{\Delta\tau}{R_e C_e}} \\ \frac{\eta_c \Delta\tau}{C_n} \end{bmatrix} i_t + \begin{bmatrix} v_{1,t} \\ v_{2,t} \end{bmatrix}, \tag{3.93}$$

$$U_t = U_{OCV,0} + K_0 SoC_t + s(SoC_t) + i_t R_o + U_{e,t} + w_t. \tag{3.94}$$

Here, $v_t = [v_{1,t}, v_{2,t}]^T$ and w_t are the process and the measurement noises which are Gaussian white noise with covariance Q and R, respectively.

We used the equivalent circuit model discussed in *Paschero et al.* [2016] as a data generator and built the two synthetic data sets. This model is shown in Fig. 3.15 and it implements the quasi-stationary, the instantaneous, and the dynamic timescales by means of a nonlinear capacitor, a resistor, and a RC group. The nonlinear capacitor is calibrated by a look-up table from SoC to OCV with the experiment data of a lithium polymer cell. The capacity is 15 Ah and the nominal voltage is 3.6 V. The internal ohmic resistance is 0.01035 Ω. The parameters of the RC group is $R_e = 0.03726, C_e = 12000$.

The data generated with the circuit of Fig. 3.15 can be considered as a representative of a real electrochemical cell [*Paschero et al.*, 2016]. The charging current i has been generated by simulating the use of an energy storage system in a pure electric vehicle, synthesizing current and voltage sequences as close as possible to those measurable in a real cell during its activity. The software CarSim has been used for generating data related to energy systems of electric vehicles. Both the training set and the test set have been generated by simulating a typical daily journey which are shown in Fig. 3.16 a) and Fig. 3.17, respectively. The data have been collected considering a sampling time $\Delta\tau = 1$ s. In order to generate the sequences of SoC, U_e, U_{OCV}, U, the obtained current profiles have been applied to the related equivalent circuit. The circuit model

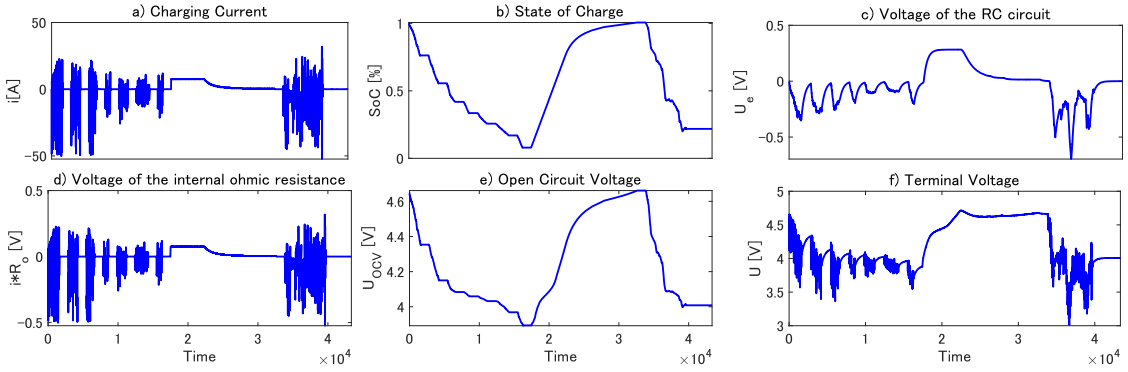

FIGURE 3.16: Generated training set: a) charging current; b) state of charge; c) voltage of the RC circuit; d) Voltage of the internal ohmic resistance; e) open circuit voltage; f) terminal voltage.

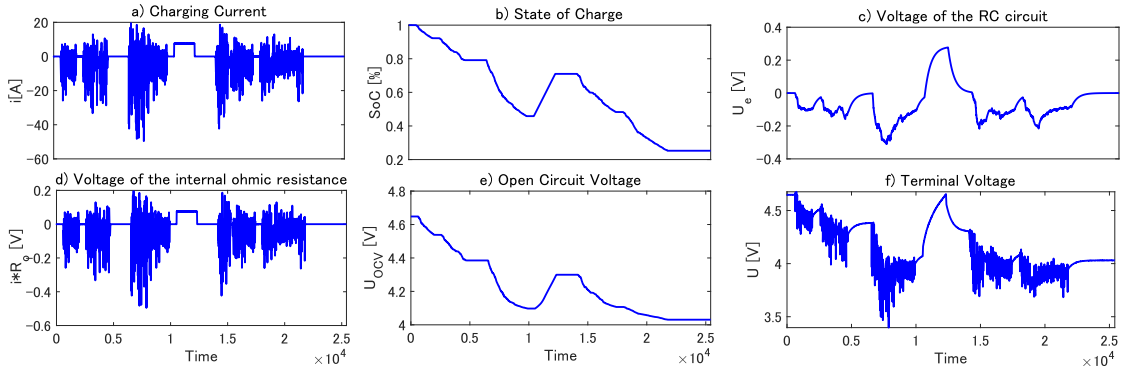

FIGURE 3.17: Generated testing set: a) charging current; b) state of charge; c) voltage of the RC circuit; d) Voltage of the internal ohmic resistance; e) open circuit voltage; f) terminal voltage.

has been initialized for considering a full charged cell in a stationary condition, namely, the state variables SoC and U_e have been initialized to one and zeros, respectively. Furthermore, the inaccuracy of the voltage and current sensors has been emulated by adding a zeros mean Gaussian noise to the charging current and the terminal voltage. The Gaussian noise added to the terminal voltage has a standard deviation equal to $1e-4$ V, while the one of the input current has a standard deviation equal to $3e-3$ A which match the standard accuracy of real voltage and current sensors. The traces of i, SoC, U_e, iR_o, U_{OCV}, U of the training set and testing set are shown in Fig. 3.16 and Fig. 3.17, respectively.

We compared our proposed method with the Sigma-point Kalman Filter (SKF) applied in [*Plett*, 2006] and Neural Netwokr circuit model-based Sigma-point Kalman Filter (NN-SKF) proposed in [*Luzi et al.*, 2019, 2020]. SKF is a classical method used in the SoC estimation. NN-SKF can be regarded as the state-of-the-art in the SoC estimation.

The estimation and prediction results of U_e, SoC, U_{OCV}, U are plotted in Fig. 3.18. The MSE performances are summarized in Tab. 1. SKF represents the Sigma-point

TABLE 3.1: Summary of estimation and prediction performances (MSE)

Method/Subject	U_e	SoC	U_{OCV}	U
SKF	2.3892e-04	8.4945e-04	2.3687e-04	7.4554e-07
NN-SKF	1.1354e-04	1.7975e-04	1.1250e-04	7.4175e-07
DKF	**2.5399e-08**	**3.3358e-07**	**1.8979e-08**	**1.5031e-08**

Kalamn filter. The function from SoC to OCV adopts the experiment data-based look-up table. Linear interpolation is used to look up values in the table. NN-SKF represents the SKF with neural network model to approximate the function from SoC to OCV. In both SKF and NN-SKF, the nonlinear response is supposed to be known. DKF is the proposed decimating Kalman filter. We trained the parameters of linear part and nonlinear part based on the presented training algorithm in Section 3.5 with training set and then used the decimating Kalman filter introduced in Section 3.5 to the testing set. The neural networks used in both NN-SKF and DKF are single layer with 100 activation functions. The results shows that NN-SKF has better performance than SKF since neural network model has better fitting performance than the linear interpolation of the look-up table. The proposed DKF outperforms than SKF and NN-SKF because the optimal feedback gain can be obtained after decimating the nonlinear response in DKF while SKF can only approximate the optimal estimation by extracting samples.

For our proposed method, we also checked the fitting performance of open circuit voltage which is shown in Fig. 3.19. The linear part of U_{OCV} have been separated.

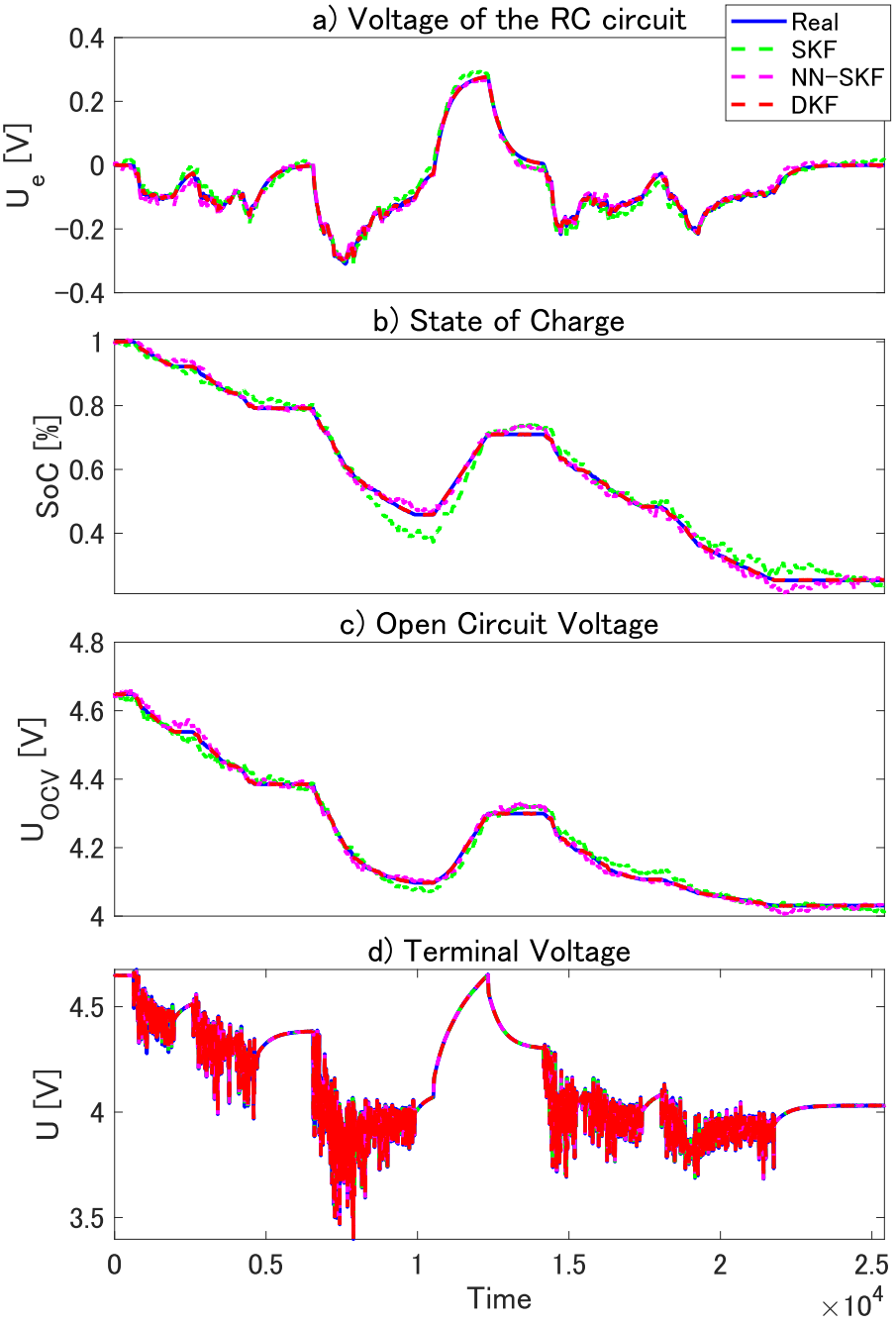

FIGURE 3.18: Generated testing set: a) voltage of the RC circuit; b) state of charge; c) open circuit voltage; d) terminal voltage.

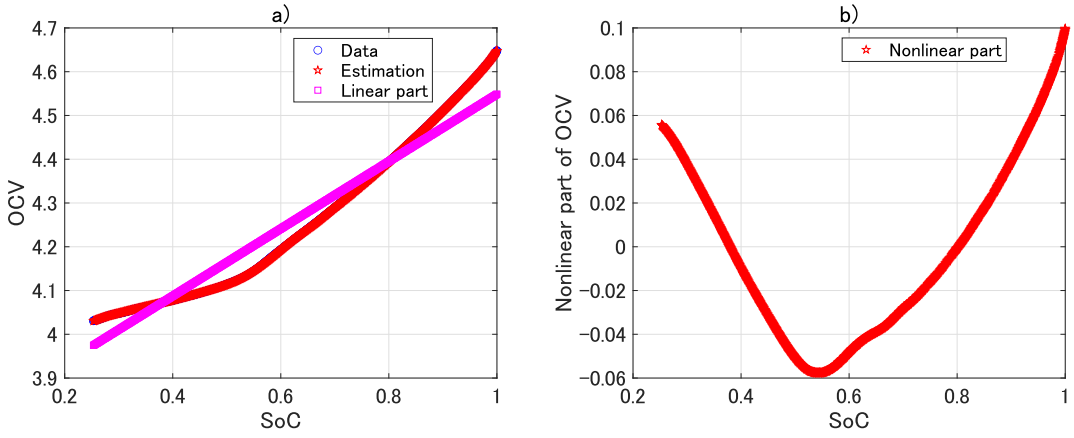

FIGURE 3.19: Fitting performance of open circuit voltage.

Chapter 4

Residual Convergence in Approximating Chance Constrained Optimization

4.1 Background and Motivation

Uncertain optimization are optimization problems involved uncertainties in constraints or objectives [*Ben-Tal and Nemirovski*, 1999]. Denote $u \in \mathbb{R}^{n_u}$ for the vector of decision variable. Denote $\delta \in \Delta \subseteq \mathbb{R}^{n_\delta}$ for the vector of uncertain variable. The cost function to be optimized is written as $J(u)$. "optimizing" means "minimizing". The constraints of the vector of decision variable is influenced by δ. Thus, we can generally denote $\mathcal{U}_\delta \subset \mathbb{R}^{n_\delta}$ as the feasible region of u for a given δ. Uncertain optimization can be written as

$$\min_{u \in \mathcal{U}_\delta} J(u). \tag{4.1}$$

However, the optimization problem is not well defined because (4.2) does not describe how the uncertain variable δ should be accounted for when we consider about the constraint $\mathcal{U}_\delta$. Since δ can be any one from the sample space Δ, a natural way to pose the problem is along a worst-case approach:

$$\min_{u \in \bigcap_{\delta \in \Delta} \mathcal{U}_\delta} J(u). \tag{4.2}$$

This is essentially robust optimization problem. *Ben-Tal and Nemirovski* [2002] and the book *Ben-Tal et al.* [2009] have had a significant role in promoting the worst-case approach in optimization.

Chance Constrained Optimization (CCO) or Probabilistic Constrained Optimization (PCO) are mathematical programs involved random variables in constraints which is required to be satisfied in a given probability level [*Charnes and Cooper*, 1959, 1970]. In this study, the appellations of chance constrained optimization and probabilistic constrained optimization are both adopted and they are exactly expressing the same meaning. Chance constrained optimization can be regarded as a relaxation of the robust optimization. The motivation of considering chance constrained optimization is that robust optimization considers the worst case in which the feasible set might be a null set. Instead, chance constrained optimization is posed the following constraints:

$$\mathcal{U}_f = \bigcup_{\Delta_s \in \mathcal{F}} \bigcap_{\delta \in \Delta_s} \mathcal{U}_\delta. \tag{4.3}$$

Here, $\mathcal{F}$ is a subset of the σ-algebra of Δ, $\mathcal{D}$:

$$\mathcal{F} = \{\Delta_s \in \mathcal{D} | \Pr\{\Delta_s\} \geq 1 - \alpha\}. \tag{4.4}$$

Here, $\alpha \in [0, 1)$ denotes a given probability level. Thus, for chance constrained optimization, we do not have to satisfy the constraints posed by all possible δ. The constraint only have to be satisfied by a desired probability. We can regard α as the level of robustness. If α is 0, the level of robustness is complete.

Chance constrained optimization has been widely applied in various industrial fields, especially for robust modeling, robust control, robust decision-making. For example, *Schildbach et al.* [2014] has proposed a stochastic model predictive control with bounds on closed-loop constraint violations based on the idea of solving a chance constrained optimization problem to obtain the constrained optimal input series for a stochastic system. *Guo and Zavlanos* [2018] addresses a motion plan problem with uncertainty. A hard constraint that rejects all the uncertainty would result a null set for the input. Thus, the chance constraint is introduced to relax the hard constraints to soft constraints. The motion plan problem becomes a chance constrained optimization problem (or probabilistic constrained problem). In *Gautam et al.* [2020], chance constrained optimization is applied to model the probabilistic energy storage in order to quantify market constrained reliability. In *Shen and Shen* [2018], *Shen et al.* [2017], the combustion control problem is reformulated as a chance constrained optimization problem, in which the engine combustion efficiency is optimized with constraint on probability on abnormal combustion. Another interesting application of chance constrained optimization in

automotive industry is introduced in *Moser et al.* [2018]. The adaptive cruise control problem is formulated as a chance constrained optimization problem The hard spacing constraint is relaxed to a flexible spacing constraint. The distance is just required to be within an interval with a probability level. Then, the energy efficiency of the vehicle can be dramatically improved.

Recently, chance constrained optimization has also been introduced to solve problems in machine learning field, both theory and applications. *Shen et al.* [2020] proposes a novel method to calculate the probabilistic bound of the predictive state trajectory of an uncertain system by reformulating the bound calculation problem into a chance constrained optimization problem. In *Campi* [2010], a novel classification method which can guarantees the probability of classification error is proposed. The bound for classification is essentially obtained by solving a chance constrained problem. Chance constrained optimization can also be used to improve the robustness of machine learning system towards the adversarial perturbations. For example, *Madry et al.* [2018] proposes a robust optimization-based method to train a neural network. Although it is called robust optimization, it has been finally relaxed to a probabilistic constrained optimization to escape from the null solution issue. Besides, *Goodwin et al.* [2020] decreases the computation burden of solving a robust optimization problem for obtaining a robust neural network. In reinforcement learning, recent research has been focused on how to achieve reinforcement learning with safe constraints [*Achiam et al.*, 2017]. Chance constrained optimization can definitely contribute to improve reinforcement learning to safe reinforcement learning. More interesting applications of chance constrained optimization has been summarized in *Campi and Garatti* [2019].

The above introduction about the wide application of chance constrained in various fields emphasises the importance of chance constrained optimization. However, chance constrained optimization is NP-hard due to the chance constraints and cannot be solved directly. The challenge of solving chance constrained program directly motivates the development of approximation approach to solve it. Unfortunately, the bias obtained by the approximation causes the problem that we do not have confidence in the solution. Besides, sample-based approximation always brings non-smoothness in the approximate problems which makes the optimization problem solver hard to find the solution. The above two points really constraints the application of chance constrained optimization. Thus, it is really important to discuss how to give better approximation of the chance constrained optimization problem.

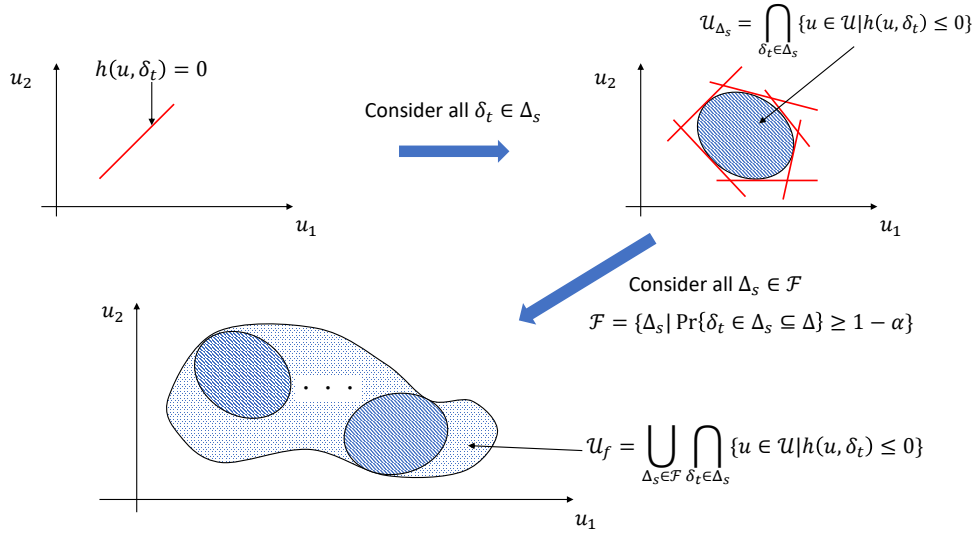

FIGURE 4.1: Formulation of probabilistic feasible domain.

4.2 Problem Description

In this section, we give a problem description of chance constrained optimization which is with more details and is more formally written than the compact one in the previous section.

Chance constrained optimization can be generally written as:

$$\begin{aligned} &\min_{u \in \mathcal{U}} \; J(u) \\ &s.t. \quad \Pr\{h(u,\delta) \le 0\} \ge 1-\alpha, \;\; \delta \in \Delta, \end{aligned} \tag{4.5}$$

where $u \in \mathcal{U} \subset \mathbb{R}^{n_u}$ is the decision variable, the decision variable domain $\mathcal{U}$ is supposed to be bounded, $\delta \in \Delta \subset \mathbb{R}^{n_\delta}$ is an uncertain parameter vector, the set Δ is the sample space of δ on which a σ-algebra $\mathcal{D}$ and a probability measure Pr are defined, $\alpha \in [0,1)$ is a given probability level for violation of chance constraints. Moreover, $J(u) : \mathcal{U} \to \mathbb{R}$ and $\forall \delta \in \Delta, h(u,\delta) : \mathcal{U} \times \Delta \to \mathbb{R}^l$ are continuous and differentiable in u. Chance constraints appearing in Eq. (4.5) emerge in various applications and can be regarded as a compromise of hard constraints which require to be satisfied for all values $\delta \in \Delta$. The feasible solution set under hard constraints could be too conservative and sometimes it could be a null set.

The feasible decision domain can be denoted as $\mathcal{U}_f$. However, it is difficult to obtain the exact expression of $\mathcal{U}_f$. Fig. 4.1 gives illustration on the formulation of probabilistic feasible domain in a 2-dimension space. Denoting $\Delta_s \subset \Delta$ and the probability measure

of Δ_s satisfies

$$\Pr\{\Delta_s\} \geq 1 - \alpha. \tag{4.6}$$

The feasible domain for $\delta_t \in \Delta_s$ is defined as

$$\mathcal{U}_{\delta_t} = \{u \in \mathcal{U} | h(u, \delta_t) \leq 0\}. \tag{4.7}$$

Then, the feasible domain for Δ_s is intersection of $\mathcal{U}_{\delta_t}$ for all $\delta_t \in \Delta_s$, which is written as

$$\mathcal{U}_{\Delta_s} = \bigcap_{\delta_t \in \Delta_s} \mathcal{U}_{\delta_t}. \tag{4.8}$$

Considering a family of Δ_s denoted as

$$\mathcal{F} = \{\Delta_s \subset \Delta | \Pr\{\Delta_s\} \geq 1 - \alpha\}, \tag{4.9}$$

the feasible decision domain for Eq. (4.5) can be defined as:

$$\mathcal{U}_f = \bigcup_{\Delta_s \in \mathcal{F}} \mathcal{U}_{\Delta_s} = \bigcup_{\Delta_s \in \mathcal{F}} \bigcap_{\delta_t \in \Delta_s} \mathcal{U}_{\delta_t}. \tag{4.10}$$

Obviously, even if $\mathcal{U}_{\delta_t}$ is known, it is impossible to obtain explicit expression or domain of $\mathcal{U}_f$ due to infinite times' operation of intersection and union. Thus, program (4.5) is NP hard due to the chance constraint. To address program (4.5), the following issues should be considered:

- How to approximate chance constraints, namely approximate the probabilistic feasible domain of decision variable;
- How to approximate the optimizer in the probabilistic feasible domain.

4.3 Related Works

The challenge of solving chance constrained program directly motivates the development of approximation approach to solve it. In recent 20 years, the main stream has converged to scenario approach [*Calafiore and Campi*, 2006] in which deterministic constraints imposed for finite sets of independently extracted samples of uncertain parameters are used to replace the chance constraints. Scenario approach preserves that the solution of the newly formulated deterministic program satisfies chance constraints with a determined bound of probability [*Calafiore and Campi*, 2006]. Afterwards, scenario approach with tight confidence bounds has been developed, in which a set of sampled constraints with tight confidence on violation probability is determined for approximating chance

constraints. For instance, a certain proportion of parameter samples to define a set of sampled constraints and discard the rest can be used to approximate chance constraints with tight violation probabilities for fixed sample number [*Campi and Garatti*, 2011]. However, scenario approach still has fatal drawbacks. It cannot ensure that the obtained solution is the optimal one in the probabilistic feasible domain. when the sample number becomes larger, the obtained solution becomes more conservative and finally converges to the totally robust solution which is feasible for all uncertainty realizations. Moreover, the solution depends highly on the chosen samples of uncertain parameters, which might be guided to the wrong directions due to the bad choices of samples.

Sample average approach has been proposed in *Luedtke and Ahmed* [2008], *Pagnoncelli et al.* [2009] as a development of scenario approach. A sample average program is used as approximation of the chance constrained program in which the chance constraints are replaced by a measure to indicate the violation probability. The feasible region and optimal solution of sample average program are proved to converge to the ones of the original problem if Lipschitz continuity of cost function and constraint function is preserved. However, there is two drawback of using sample average program:

1. First, the sample average program is a mixed integer program which is very difficult to solve;
2. Second, approximating the probability directly will obtain flatness in the constraint function which makes gradient based method hard to move to the local minimum.

Thus, [*Pena-Ordieres et al.*, 2020] proposes a smooth sample average program to address the above problem.

Different from the sample-based method, *Geletu et al.* [2017] proposes an inner-outer approximate approach to approximate the chance constraint by using explicit function. However, to establish such approximate function precisely is really challenging.

Bayesian optimization framework has been applied to optimization under unknown constraints recently [*Gramacy and Lee*, 2010, *Picheny et al.*, 2016], which is essentially a data-driven approach for approximating the optimizer of the program. Statistical approach based on Gaussian processes and Bayesian learning to both approximate the unknown function and estimate the probability of meeting the constraints is developed to approximate the optimizer in unknown feasible domain. While, this can only be applied to expected constraints. Also, Gaussian process model is still not precise enough for approximating the feasible domain described by chance constraints.

4.3.1 Scenario approach

The following content about scenario approach summarizes the research about scenario approach presented in *Calafiore and Campi* [2006], *Campi and Garatti* [2008], *Campi et al.* [2018].

Here, we still adopt the notations in Section 4.2. δ is a random variable in the sample space Δ. The sample space Δ is endowed with a σ-algebra $\mathcal{D}$ and a probability measure Pr. Moreover, let $(\Delta^m, \mathcal{D}^m, \Pr^m)$ be the m-fold Cartesian product of Δ equipped with the product σ-algebra $\mathcal{D}^m$ and the product probability $\Pr^m = \Pr \times ... \times \Pr(m \text{ times})$. Namelyk a point in $(\Delta^m, \mathcal{D}^m, \Pr^m)$ is a sample $(\delta^{(1)}, ..., \delta^{(m)})$ of m elements drawn independently from Δ according to the identical probability Pr. Note that each $\delta^{(i)}$ is regarded as an observation, and it is called a "scenario". since the approach is based on extracting $\delta^{(i)}$, it is named as "scenario approach". The decision space is defined by $\mathcal{U}$. As defined in (4.7), for each $\delta^{(i)}$, we have a constraint set $\mathcal{U}_{\delta^{(i)}}$. With $(\delta^{(1)}, ..., \delta^{(m)})$, we can construct the following constrained optimization problem:

$$\begin{aligned} &\min_{u \in \mathcal{U}} J(u) \\ &s.t. \quad u \in \bigcap_{i=1}^{m} \mathcal{U}_{\delta^{(i)}}. \end{aligned} \tag{4.11}$$

Assuming that a unique solution u_m^* exists, possibly after applying a tie-break rule[*Calafiore and Campi*, 2005]. We define a map from $(\delta^{(1)}, ..., \delta^{(m)})$ to u_m^* as:

$$\mathcal{A}_m : \Delta^m \to \mathcal{U}. \tag{4.12}$$

Namely, for a given $(\delta^{(1)}, ..., \delta^{(m)}) \in \Delta^m$, if the problem (4.11) is feasible and has a unique solution u_m^*, we have $\mathcal{A}_m(\delta^{(1)}, ..., \delta^{(m)}) = u_m^*$.

The following assumption is in force through out the scenario approach:

Assumption 4.3.1. To every $\delta \in \Delta$ there is associated a constraint set $\mathcal{U}_\delta \subseteq \mathcal{U}$, which identifies the decisions that are admissible for a given δ. For all $m = 1, 2, ...$ and for any sample $\delta^{(1)}, ..., \delta^{(m)}$, it holds that $\mathcal{A}_m(\delta^{(1)}, ..., \delta^{(m)}) \in \bigcap_{i=1}^{m} \mathcal{U}_{\delta^{(i)}}$.

We henceforth call N the actual, fixed, number of scenarios that we observe in a given application. The remained question is whether $u_N^* = \mathcal{A}_N(\delta^{(1)}, ..., \delta^{(N)})$ satisfies the chance constraint or not. This is important to certify how "robust" u_N^* is against the uncertainty of δ.

Definition 4.3.1. The violation probability of a given decision $u \in \mathcal{U}$ is defined as

$$V(u) := \Pr\{\delta \in \Delta : u \notin \mathcal{U}_\delta\}. \tag{4.13}$$

For chance constrained optimization, the feasible decision should satisfy that $V(u) < \alpha$.

Note that the element $(\delta^{(1)}, ..., \delta^{(N)})$ is random variable in Δ^N since each scenario $\delta^{(i)}$ is randomly sampled from Δ according to Pr. $u_N^* = \mathcal{A}_N(\delta^{(1)}, ..., \delta^{(N)})$ is determined by $(\delta^{(1)}, ..., \delta^{(N)})$. Thus, the violation probability of the scenario decision $V(u_N^*)$ is random variable defined over Δ^N.

The distribution of $V(u_N^*)$ has been the object of intense study for the case when u_N^* is obtained as the solution of a convex optimization [*Calafiore and Campi*, 2005, 2006, *Campi and Garatti*, 2008]. The deepest result is established in *Campi and Garatti* [2008], where it is shown that the distribution of $V(u_N^*)$ is dominated by a Beta distribution, namely,

$$\Pr^N\{V(u_N^*) > \alpha\} \leq \beta, \tag{4.14}$$

where

$$\beta = \sum_{i=0}^{n_u - 1} \binom{N}{i} \alpha^i (1-\alpha)^{N-i}. \tag{4.15}$$

This result is tight in that (4.14) holds with euqality for a whole class of convex optimization problems which are called fully supported problems in *Campi and Garatti* [2008].

In *Grammatico et al.* [2016], the bounds described by (4.14) and (4.15) have been extended to the case with a nonconvex cost function and convex constraints. The feasibility domain is restricted to a region that is obtained as the convex hull of few points to enable the application of the results of *Campi and Garatti* [2008]]. *Campi et al.* [2018] gave the generalization result for a nonconvex program in which both $J(u)$ and $\mathcal{U}_\delta$ need not to be convex.

Definition 4.3.2. Given a sample $(\delta^{(1)}, ..., \delta^{(N)}) \in \Delta^N$, a support subsample S for $(\delta^{(1)}, ..., \delta^{(N)})$ is $k-$tuple of elements extracted from $(\delta^{(1)}, ..., \delta^{(N)})$, i.e., $S = (\delta^{(i_1)}, ..., \delta^{(i_k)})$ with $i_1 < ... < i_k$, which gives the same solution as the original sample, that is,

$$\mathcal{A}_k(\delta^{(i_1)}, ..., \delta^{(i_k)}) = \mathcal{A}_N(\delta^{(1)}, ..., \delta^{(N)}). \tag{4.16}$$

A support subsample $S = (\delta^{(i_1)}, ..., \delta^{(i_k)})$ is said to be irreducible if no element can be further removed from S leaving the solution unchanged. According to the results in *Calafiore and Campi* [2005, 2006], *Campi and Garatti* [2008], if $\mathcal{U}_\delta$ is convex for any $\delta \in \Delta$, the maximal number of support subsample is n_u for any $N > n_u$. Thus, we have the results in (4.14) and (4.15).

We denote a function $\mathcal{B}_N : (\delta^{(1)}, ..., \delta^{(N)}) \to \{i_1, ..., i_k\}, i_1 < ... < i_k$ such that $(\delta^{(i_1)}, ..., \delta^{(i_k)})$ is a support subsample. Let $s_N^* = |\mathcal{B}_N(\delta^{(1)}, ..., \delta^{(N)})|$.

For nonconvex cases, the result is stated by the following Theorem:

Theorem 4.3.1. Suppose that Assumption 4.3.1 holds true, and set a value $\beta \in (0, 1)$ as confidence parameter. Let $\varepsilon : \{0, ..., N\} \to [0, 1]$ be a function such that

$$\varepsilon(N) = 1 \tag{4.17}$$

$$\sum_{i=1}^{N-1} \binom{N}{i} (1 - \varepsilon(i))^{N-i} = \beta. \tag{4.18}$$

Then, for any $\mathcal{A}_N, \mathcal{B}_N$, and probability Pr, it holds that

$$\Pr^N\{V(u_N^*) > \varepsilon(s_N^*)\} \leq \beta. \tag{4.19}$$

Due to the above summary of scenario approach, it is clear that scenario approach does not focus on how to approximate the original problem by approximating the feasible region. It only cares about the robustness of the approximate solution. With small sample size, there is a chance to get a good feasible solution while the risk of getting a solution which is not feasible is very high. On the other hand, if we use large sample size, the risk of getting a infeasible solution is small. However, the solution will be too conservative to the chance constrained optimization. We will state this point in the introduction for sample average approach.

4.3.2 Sample average approach

The following content about sample average approach summarized the results presented in *Luedtke and Ahmed* [2008], *Pena-Ordieres et al.* [2020].

Sample average approach also extracts samples from the sample space Δ and builds an approximate problem of the original chance constrained optimization problem. We adopt the same notations used in the introduction for scenario approach. Let $(\delta^{(1)}, ..., \delta^{(N)})$ be an independent Monte Carlo sample set of the random vector δ. Then, for fixed $\epsilon \in [0, 1)$ and $\gamma > 0$, the sample average approximation problem is defined to be

$$\begin{aligned} &\min_{u \in \mathcal{U}} \ J(u) \\ &s.t. \quad u \in \mathcal{U}_{\epsilon,\gamma}^N \end{aligned} \tag{4.20}$$

where $\mathcal{U}_{\epsilon,\gamma}^N$ is defined as

$$\mathcal{U}_{\epsilon,\gamma}^N = \{u \in \mathcal{U} | \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(h(u, \delta^{(i)} + \gamma) \leq 0) \leq 1 - \epsilon\}. \tag{4.21}$$

where $\mathbb{I}(z)$ is the indicator function which takes values one when z is true and zero otherwise.

There is an assumption for the constraint function $h(\cdot)$:

Assumption 4.3.2. There exists $L > 0$ such that

$$|h(u,\delta) - h(u',\delta)| \leq L\|u - u'\|_\infty,\ \forall u, u' \in \mathcal{U} \text{ and } \forall \delta \in \Delta. \tag{4.22}$$

Theorem 4.3.2. Suppose $\mathcal{U}$ is bounded with diameter D. Namely, we have $D > 0$ such that $D = \sup\{\|u - u'\|_\infty : u, u' \in \mathcal{U}\}$ be the diameter of $\mathcal{U}$. Besides, assume that Assumption 4.3.2 holds. Let $\epsilon \in [0, \alpha), \eta \in (0, \alpha - \epsilon)$ and $\gamma > 0$. Then,

$$\Pr\{\mathcal{U}^N_{\epsilon,\gamma} \subseteq \mathcal{U}_f\} \leq 1 - \lceil\frac{1}{\eta}\rceil\lceil\frac{2LD}{\gamma}\rceil^{n_u}\exp\{-2N(\alpha - \epsilon - \eta)^2\}. \tag{4.23}$$

For scenario approach, the bound for the required samples is less since it only cares about the feasibility of the optimal solution. However, when the number of sample increases, according the Theorem 4.3.2, the whole feasible set of scenario approach will be a subset of the feasible domain of chance constrained optimization. Since scenario approach choses $\epsilon = 0$ essentially, the feasible domain of scenario problem will be much more conservative than the original chance constrained optimization. It is feasible but not absolutely optimal.

4.4 Proposed Method

4.4.1 Problem reformulation

There are several difficulties to address problem described by (4.5):

- The structural properties of the feasible domain defined by $h(u,\delta) \leq 0$ may not succeed to the domain defined by the constraints $\Pr\{h(u,\delta) \leq 0\} \geq 1 - \alpha$. For instance, even if h are all linear in u, the chance constraint may not define a convex domain;
- Instead of knowing the distribution of δ, only the samples of δ are available;
- The tractable analytical function of chance constraint does not exist even with the knowledge of the distribution of δ.

This study is to find out a tractable analytical function $H(u)$ to define a feasible domain $\tilde{\mathcal{U}}_f = \{u \in \mathcal{U} | H(u) \leq 0\}$. The feasible domain $\tilde{\mathcal{U}}_f$ equals to the feasible domain $\mathcal{U}_f =$

$\{u \in \mathcal{U} | \Pr\{h(u,\delta) \leq 0\} \geq 1-\alpha\}$ with probability as 1. Then, the problem with deterministic constraints

$$\begin{aligned} &\min_{u \in \mathscr{U}} \ J(u) \\ &s.t. \quad H(u) \leq 0 \end{aligned} \tag{4.24}$$

is the equivalent problem of (4.5). By solving (4.24), the optimal solution of (4.5) can be approximated.

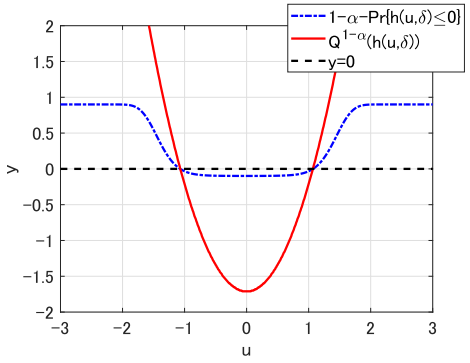

FIGURE 4.2: Comparison of $1-\alpha-\Pr\{h(u,\delta) \leq 0\}$ and $Q^{1-\alpha}(h(u,\delta))$ where $h(u,\delta) = 1.5u^2 - 3 + \delta$ and $\delta \sim N(0,1)(\alpha = 0.1)$.

The cumulative probability function of a random variable X is denote as

$$F(x) = \Pr\{X \leq x\}. \tag{4.25}$$

While, the $1-\alpha$ level quantile of a random variable X is defined as [*Steinbrecher and Shaw*, 2008]

$$Q^{1-\alpha}(X) = \inf\{x \in \mathbb{R} | \Pr\{X \leq x\} \geq 1-\alpha\}. \tag{4.26}$$

For the case $n_h = 1$, $h : \mathbb{R}^{n_x} \times \mathbb{R}^{n_\delta} \to \mathbb{R}$ is a real valued function and $h(u,\delta)$ is a scalar random variable. Thus, the cumulative probability function of $h(u,\delta)$ can be defined as

$$F(\gamma, u) = \Pr\{h(u,\delta) \leq \gamma\}, \gamma \in \mathbb{R}. \tag{4.27}$$

The quantile $Q^{1-\alpha}(h(u,\delta)) \leq 0$ is equivalent to $F(0,u) \geq 1-\alpha$. For the case $n_h > 1$, $Q^{1-\alpha}(h(u,\delta)) \leq 0$ cannot be well defined since $h(u,\delta)$ is not a scalar random variable any more. Instead of using $h(u,\delta)$, $\bar{h}(u,\delta) = \max_{j=1,\ldots,n_h} h_j(u,\delta)$ is used. Then, the cumulative probability function $F(\gamma,u)$ notes the same style as Eq. (4.27) only replacing $h(\cdot,\cdot)$ by $\bar{h}(\cdot,\cdot)$. Consequently, the quantile is written as $Q^{1-\alpha}(\bar{h}(u,\delta) \leq 0$. Without losing the generality, $Q^{1-\alpha}(\bar{h}(u,\delta) \leq 0$ can be used for $n_h = 1$ as well. Then, the

following reformulation of problem (4.5) is written as

$$\begin{aligned} &\min_{u\in\mathscr{U}} J(u) \\ &s.t. \quad Q^{1-\alpha}(\bar{h}(u,\delta)) \leq 0,\ \delta \in \Delta, \end{aligned} \tag{4.28}$$

where $\alpha \in (0,1)$ is a given probability level.

The advantage of using the quantile is that the quantile function is much less flat compared to the probability function $\Pr\{h(u,\delta) \leq 0\}$. The comparison example of the quantile function and probability function is shown in Fig. 4.2. By the quantile function-based reformulation, the feasible region is measured in the image of $\bar{h}(u,\delta)$ instead of the bounded image $[0,1]$ of the probability function.

The sample set $\Delta^N = \{\delta_1, ..., \delta_N\}$ is obtained by extracting samples independently from Δ according to the identical distribution $p_\delta(\delta)$. For a given u, the empirical CDF of $\mathcal{H}^N = \{h(u,\delta_1), ..., h(u,\delta_N)\}$ is written as

$$\tilde{F}^N(t,u) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(\bar{h}(u,\delta_i) \leq t) \tag{4.29}$$

where $\mathbb{I}(h(u,\delta_i) \leq t)$ denotes the indicator function written as

$$\mathbb{I}(\bar{h}(u,\delta_i) \leq t) = \begin{cases} 0, & \text{if } \bar{h}(u,\delta_i) > t \\ 1, & \text{if } \bar{h}(u,\delta_i) \leq t. \end{cases} \tag{4.30}$$

The $(1-\alpha)$-empirical quantile at u can be obtained from a value t such that $\tilde{F}^N(t,u) \approx 1-\alpha$, which is defined as

$$\tilde{Q}^{1-\alpha}(\bar{h}(u,\delta)) = \inf\{y | \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(\bar{h}(u,\delta_i) \leq y) \geq 1-\alpha\}, \tag{4.31}$$

equivalently written as

$$\tilde{Q}^{1-\alpha}(\bar{h}(u,\delta))) = \bar{h}_{\lceil M \rceil}(u) \tag{4.32}$$

where $M = \lceil (1-\alpha)N \rceil$ and $\bar{h}_{\lceil M \rceil}(u)$ denotes the $M-$th smallest observation of the values $\mathcal{H}^N$ for a given u.

We can use the $(1-\alpha)-$empirical quantile as an approximation of the chance constraints and form the following approximate problem

$$\begin{aligned} &\min_{u\in\mathscr{U}} J(u) \\ &s.t. \quad \tilde{Q}^{1-\alpha}(\bar{h}(u,\delta)) \leq 0,\ \delta \in \Delta. \end{aligned} \tag{4.33}$$

Using the $(1-\alpha)$-empirical quantile as an approximation of the chance constraints has two drawbacks:

- $\tilde{Q}^{1-\alpha}(\bar{h}(u,\delta)))$ is usually not differentiable. M changes for different u. Thus, even if the constraint $h(u,\delta)$ is smooth for a fixed value of δ, $\tilde{Q}^{1-\alpha}(\bar{h}(u,\delta)))$ does not have to be smooth;
- For small N, the uncertainty of $\tilde{Q}^{1-\alpha}(\bar{h}(u,\delta)))$ is large and the inward kinks of the feasible boundary will be very non-smooth. Large N can bring some smoothness back, however, this also increases the computation burden.

Thus, it is necessary to look for the smooth approximation of $(1-\alpha)$-empirical quantile.

Given the sample set Δ_N, the $(1-\alpha)$-empirical quantile $\tilde{Q}^{1-\alpha}(\bar{h}(u,\delta)))$ is essentially a function of u. $Q^{1-\alpha}(\bar{h}(u,\delta)))$ is also a function of u. By using $\tilde{Q}^{1-\alpha}(\bar{h}(u,\delta)))$ as an approximation of $Q^{1-\alpha}(\bar{h}(u,\delta)))$, a smooth function can be used to approximate the $(1-\alpha)$ quantile $Q^{1-\alpha}(\bar{h}(u,\delta)))$. In this study, single layer neural network model is used to approximate the $(1-\alpha)$ quantile $Q^{1-\alpha}(\bar{h}(u,\delta)))$. Using N independently extracted samples of δ, the neural approximation of $Q^{1-\alpha}(\bar{h}(u,\delta)))$ with S hidden nodes and activation function $g(\cdot)$ is defined as

$$\hat{H}_S(u) = \sum_{i=1}^{S} \beta_i g(u, a_i, b_i), \tag{4.34}$$

where β_i denotes the weight vector connecting the i-th hidden node and the output nodes, $a_i = [a_{i,1}, ..., a_{i,k}]$ represents the weight vector towards u, and b_i is the scalar threshold of the i-th hidden node. Here, the activation function adopts the sigmoid function expressed as

$$g(u, a_i, b_i) = \frac{1}{1 + e^{-a_i^T u + b_i}}. \tag{4.35}$$

The purpose is to achieve

$$\hat{H}_S(u) \approx Q^{1-\alpha}(\bar{h}(u,\delta))). \tag{4.36}$$

Then, $\hat{H}_S(\cdot)$ maps the decision variable $u \in \mathbb{R}^k$ into the image of $Q^{1-\alpha}(\bar{h}(u,\delta)))$ which is $\mathbb{R}$. In order to find solutions for (4.5) and (4.28), it is equivalent to solve the following problem

$$\begin{aligned} &\min_{u \in \mathscr{U}} \; J(u) \\ &s.t. \quad \hat{H}_S(u) \leq 0. \end{aligned} \tag{4.37}$$

The above formulation is a sample-based neural approximation of the original probabilistic constrained problem which is obtained by a two-layer approximation: sample

approximation and neural approximation. The convergence and feasibility of the two-layer approximation is analyzed in the next subsection.

4.4.2 Convergence and feasibility analysis

We make the following assumptions.

Assumption 4.4.1. $\bar{h}(u,\delta)$ is a Carathéodory function. For every fixed $u \in \mathcal{U}$, $\bar{h}(u,\delta)$ is a continuous random variable which is measurable and has a continuous distribution. For every fixed $\delta \in \Delta$, $\bar{h}(u,\delta)$ is a continuous function of u. Besides, for every $u \in \mathcal{U}$, $F_{\bar{h}}(\bar{h}(u,\delta))$, the cumulative distribution function of $\bar{h}(u,\delta)$, is continuously differentiable and it has strictly positive derivative of $\bar{h}(u,\delta)$ (strictly monotonically increasing), $f_{\bar{h}}(\bar{h}(u,\delta))$, over the domain of $\bar{h}(u,\delta)$: $(-\infty,+\infty)$.

Remark 4.4.1. The cumulative distribution function $F_{\bar{h}}(\bar{h}(u,\delta))$ is continuously and strictly monotonically increasing over $(-\infty,+\infty)$. Thus, the quantile function $Q(\bar{h}(u,\delta)) : [0,1] \to D_{\bar{h}(u,\delta)}$ is the inverse of $F_{\bar{h}}$ and thus continuous on $[0,1]$. Moreover, for a fixed $\delta \in \Delta$, $\bar{h}(\cdot,\delta)$ is continuous on $\mathcal{U}$. Thus, the values of the cumulative distribution function $F_{\bar{h}}(\bar{h}(u,\delta))$ and the quantile function $Q(\bar{h}(u,\delta))$ vary continuously on $\mathcal{U}$ due to the fact that continuous functions' function composition leads to continuous function. Thus, Assumption 4.4.1 implies that the $1-\alpha$ level quantile $Q^{1-\alpha}(\bar{h}(u,\delta))$ is continuous function of u on $\mathcal{U}$.

Assumption 4.4.2. The problem expressed by (4.5) has a globally optimal solution u^*, such that for any ϵ_u there is $u \in \mathcal{U}$ such that $\|u-u^*\| \leq \epsilon_u$ and $F(0,u) > 1-\alpha$ (or equivalently $Q^{1-\alpha}(\bar{h}(u,\delta)) < 0$).

Remark 4.4.2. Assumption 4.4.2 implies that there exists a sequence $\{u_k\}_{k=1}^{\infty} \subseteq \mathcal{U}$ that converges to an optimal solution u^* such that $F(0,u_k) > 1-\alpha$ (or equivalently $Q^{1-\alpha}(\bar{h}(u_k,\delta)) < 0$) for all $k \in \mathbb{N}$.

By applying Corollary 21.5 of [*van der Vaart*, 1998], we have the following lemma about the convergence of $\tilde{Q}^{1-\alpha}(\bar{h}(u,\delta))$:

Lemma 4.4.1. Suppose Assumption 4.4.1 holds. For any fixed $\alpha \in (0,1)$ and any fixed $u \in \mathcal{U}$, if N samples of δ, $\delta_i, i = 1,...,N$, are independently extracted, then

$$\tilde{Q}^{1-\alpha} - Q^{1-\alpha} = -\frac{1}{N}\sum_{i=1}^{N} \frac{\mathbb{I}(\bar{h}_i \leq Q^{1-\alpha}) - (1-\alpha)}{f(Q^{1-\alpha})} + o(1). \tag{4.38}$$

Here, $\tilde{Q}^{1-\alpha}$, $Q^{1-\alpha}$, and $\bar{h}_i$ are shorts for $\tilde{Q}^{1-\alpha}(\bar{h}(u,\delta))$, $Q^{1-\alpha}(\bar{h}(u,\delta))$ and $\bar{h}(u,\delta_i)$, respectively.

Consequently, the sequence $\{\tilde{Q}^{1-\alpha} - Q^{1-\alpha}\}$ is asymptotically normal with mean 0 and variance $\frac{\alpha(1-\alpha)}{N \cdot f^2(Q^{1-\alpha})}$. As N increases to ∞, the variance also vanishes to zeros.

For the convergence of $\hat{H}_S(u)$ to $Q^{1-\alpha}(\bar{h}(u,\delta))$, we have the following theorem:

Theorem 4.4.1. Suppose Assumption 4.4.1 holds, as $N \to \infty$, $\forall \epsilon_H > 0$ there exists S, a, b, δ such that

$$\sup_{u \in \mathcal{U}_c} \|\hat{H}_S(u) - Q^{1-\alpha}(\bar{h}(u,\delta))\| < \epsilon_H, \quad w.p.1. \tag{4.39}$$

Here, $\mathcal{U}_c \subseteq \mathcal{U}$ represents any compact set inside the feasible area.

Proof. (Theorem 4.4.1) Due to Assumption 4.4.1 and Remark 4.4.1, $Q^{1-\alpha}(\bar{h}(u,\delta))$ is continuous function of u. Then, according to the universal approximation theorem [*Cybenko*, 1989, *Hassoun*, 1995], $\forall \epsilon_H > 0, \forall Q^{1-\alpha}(\bar{h}(u,\delta))$, $\exists S \in \mathbb{N}^+, \beta_i, b_i \in \mathbb{R}^S, a_i \in \mathbb{R}^{S \times k}$ in Eq. (4.34) and (4.35) such that

$$\|\hat{H}_S(u) - Q^{1-\alpha}(\bar{h}(u,\delta))\| < \epsilon_H, \tag{4.40}$$

for all $u \in \mathcal{U}$. However, $Q^{1-\alpha}(\bar{h}(u,\delta))$ is not used directly for fitting the quantile function. The used one is the approximate value $\tilde{Q}^{1-\alpha}(\bar{h}(u,\delta))$ which has bias compared to $Q^{1-\alpha}(\bar{h}(u,\delta))$.

The bias is defined by Eq. (4.38). Denote $dQ^{1-\alpha} = \tilde{Q}^{1-\alpha} - Q^{1-\alpha}$. According to Lemma 4.4.1, $dQ^{1-\alpha}$ is normal with mean 0 and variance $\frac{\alpha(1-\alpha)}{N \cdot f^2(Q^{1-\alpha})}$. As $N \to \infty$, $\frac{\alpha(1-\alpha)}{N \cdot f^2(Q^{1-\alpha})} \to 0$. Namely, $\|dQ^{1-\alpha}\| \to 0$ with probability 1.

Since we can only obtain $\tilde{Q}^{1-\alpha}(\bar{h}(u,\delta))$, instead of Eq. (4.40) which needs $Q^{1-\alpha}(\bar{h}(u,\delta))$, the following inequality

$$\|\hat{H}_S(u) - \tilde{Q}^{1-\alpha}(\bar{h}(u,\delta))\| < \epsilon_H, \tag{4.41}$$

can be obtained $\forall \epsilon_H > 0, \forall Q^{1-\alpha}(\bar{h}(u,\delta))$, $\exists S \in \mathbb{N}^+, \beta, b \in \mathbb{R}^S, a \in \mathbb{R}^{S \times k}$. Since $\|\hat{H}_S(u) - Q^{1-\alpha}(\bar{h}(u,\delta))\| = \|\hat{H}_S(u) - \tilde{Q}^{1-\alpha}(\bar{h}(u,\delta)) + \tilde{Q}^{1-\alpha}(\bar{h}(u,\delta)) - Q^{1-\alpha}(\bar{h}(u,\delta))\| \leq \|\hat{H}_S(u) - \tilde{Q}^{1-\alpha}(\bar{h}(u,\delta))\| + \|dQ^{1-\alpha}\|$, we have

$$\|\hat{H}_S(u) - Q^{1-\alpha}(\bar{h}(u,\delta))\| < \epsilon_H + \|dQ^{1-\alpha}\|. \tag{4.42}$$

Namely, for any $\epsilon_H > 0$, we have parameters to satisfy Eq. (4.42). Notice that $\|dQ^{1-\alpha}\|$ becomes 0 with probability 1 as $N \to \infty$. Besides, for any N there exist S, a, b, δ to satisfy Eq. (4.42) for any $\epsilon_H > 0$. Thus, Eq. (4.39) is proved. □

□

Let A and A_N^S denote the sets of optimal solutions for problems (4.5) and (4.37) respectively. Notice that problem (4.28) share the same set of optimal solutions as problem (4.5). Besides, denote J^* and J_N^S the optimal values of cost functions in problems (4.5) and (4.37) respectively. The convergences of J_N^S and A_N^S as N and S increase are summarized in the following:

Theorem 4.4.2. Suppose that $\mathcal{U}$ is compact, the function $J(\cdot)$ is continuous, and Assumptions 4.4.1 and 4.4.2 hold. Then, $J_N^S \to J^*$ and $\mathbb{D}(A_N^S, A) \to 0$ w.p.1.

Proof. (Theorem 4.4.2) Due to Assumption 4.4.2, the set A is nonempty and there exists $u \in \mathcal{U}$ such that $Q^{1-\alpha}(\bar{h}(u,\delta)) < 0$. According to Theorem 4.4.1, $\hat{H}_S(u)$ converges to $Q^{1-\alpha}(\bar{h}(u,\delta)) < 0$, and thus we can find $S_0, a_0, b_0, \beta_0, N_0$ such that $\hat{H}_S(u) - \epsilon_H - \|dQ^{1-\alpha}\| \leq 0$. ($\epsilon_H$ can be any small value to 0 and $\|dQ^{1-\alpha}\|$ decreases to 0 with probability 1 as $N \to \infty$.) Because $\hat{H}_S(u)$ is continuous in u and $\mathcal{U}$ is compact, the feasible set of the approximation problem (4.37) is compact as well. Besides, A_N^S is not empty w.p.1 for all $N \leq N_0$ and all S, a, b, β, N for smaller error bound (with higher probability) $\epsilon_H + \|dQ^{1-\alpha}\|$ of Eq. (4.42). Here, we denote $\hat{H}_{S,0}(u)$ for $\hat{H}_S(u)$ with parameters $\{S_0, a_0, b_0, \beta_0, N_0\}$ and input u. Analogously, $\hat{H}_{S,k}(u_k)$ is for $\hat{H}_S(u)$ with parameters $\{S_k, a_k, b_k, \beta_k, N_k\}$ and input u_k. With parameters $\{S_k, a_k, b_k, \beta_k, N_k\}$, the corresponding notations for the optimal solution set and optimal cost value are generally defined as $A_{N,k}^{S,k}$ and $J_{N,k}^{S,k}$.

Let $\{N_k\}_{k=1}^{\infty} \geq N_0$ and $\{S_k, a_k, b_k, \beta_k, N_k\}$ be two sequences such that the corresponding $\epsilon_{s,k}$ as in Eq. (4.42) decreases to 0. Let $\hat{u}_k \in A_{N,k}^{S,k}$ which means that $\hat{u}_k \in \mathcal{U}$, $\hat{H}_{S,k}(\hat{u}_k) \leq 0$ and $J_{N,k}^{S,k} = J(\hat{u}_k)$. Let $\hat{u} \in \mathcal{U}$ be any cluster point of $\{\hat{u}_k\}_{k=1}^{\infty}$. Define $\{\hat{u}_l\}_{l=1}^{\infty}$ be a subsequence converging to $\hat{u}$. Since $\hat{H}_{S,l}(u)$ defined by $\{S_l, a_l, b_l, \beta_l, N_l\}$ is continuous and converges uniformly to $Q^{1-\alpha}(\bar{h}(u,\delta))$ on $\mathcal{U}$ w.p.1, we have that $Q^{1-\alpha}(\bar{h}(\hat{u},\delta)) = \lim_{l\to\infty} \hat{H}_{S,l}(\hat{u}_l)$ w.p.1. Therefore, $Q^{1-\alpha}(\bar{h}(\hat{u},\delta)) \leq 0$ and $\hat{u}$ is feasible for the true problem, and $J(\hat{u}) \geq J^*$. Moreover, $J(\hat{u}_l) \to J(\hat{u})$ w.p.1, which means that $\lim_{l\to\infty} J_{N,l}^{S,l} \geq J^*$. The above is true for any cluster point of $\{\hat{u}_k\}_{k=1}^{\infty}$ in the compact set $\mathcal{U}$, we have

$$\liminf_{k\to\infty} J_{N,k}^{S,k} \geq J^*, \quad w.p.1. \tag{4.43}$$

Now, by Assumption 4.4.2 and Remark 4.4.2, there exists an optimal solution u^* and a sequence $\{\hat{u}_l\}_{l=1}^{\infty}$ converging to u^* with $Q^{1-\alpha}(\bar{h}(\hat{u}_l,\delta)) < 0$. Note that $\hat{H}_{S,l}(\hat{u}_l)$ converges to $Q^{1-\alpha}(\bar{h}(\hat{u}_l,\delta))$ w.p.1, and thus there exist $K(l)$ such that $\hat{H}_{S,k}(\hat{u}_l) \leq 0$ for every $k \geq K(l)$ and every l, w.p.1. Assume that $K(l) < K(l+1)$ for every l without loss of generality and define the sequence $\{\bar{u}_k\}_{k=K(1)}^{\infty}$ by setting $\bar{u}_k = u_l$ for all k and l with $K(l) \leq k < K(l+1)$. We then have $\hat{H}_{S,k}(\bar{u}_k) \leq 0$, which implies $J_{N,k}^{S,k} \leq J(\bar{u}_k)$ for all

$k \geq K(1)$. Since f is continuous and $\bar{u}_k$ also converges to u^*, we have

$$\lim_{k\to\infty} \sup J_{N,k}^{S,k} \leq J(u^*) = J^*, \quad w.p.1. \tag{4.44}$$

Thus, $J_{N,k}^{S,k} \to J(u^*)$ w.p.1 when $k \to \infty$. Namely, $J_N^S \to J^*$.

For the proof of $\mathbb{D}(A_N^S, A) \to 0$ w.p.1, it can refer to Theorem 5.3 of *Shapiro et al.* [2014].
□ □

For the finite sample feasibility analysis of the approximation problem solutions, we will make use of Hoeffiding's inequality [*Hoeffding*, 1963, *Shapiro et al.*, 2014]:

Theorem 4.4.3. Denote $Z_1, ..., Z_N$ for independent random variables with bounded sample spaces, namely $\Pr\{Z_i \in [z_{i,min}, z_{i,max}]\} = 1, \forall i \in \{1, ..., N\}$. Then, if $s > 0$,

$$\Pr\{\sum_{i=1}^{N}(Z_i - \mathbb{E}\{Z_i\}) \geq sN\} \leq e^{-\frac{2N^2s^2}{\sum_{i=1}^{N}(z_{i,max}-z_{i,min})^2}}. \tag{4.45}$$

Based on Hoeffding's inequality, probabilistic feasibility guarantee of the sample approximation method is proved in *Luedtke and Ahmed* [2008], *Pena-Ordieres et al.* [2020] and summarized here as:

Theorem 4.4.4. Let $u \in \mathcal{U}$ be such that $u \notin \mathcal{U}_f$. Then,

$$\Pr\{\tilde{F}^N(-\gamma, u) \geq 1 - \beta\} \leq e^{-2N\tau_u^2} \tag{4.46}$$

where $\gamma > 0$, $\beta \in [0, \alpha]$, and $\tau_u > 0$ is written as

$$\tau_u = F(0, u) - F(-\gamma, u) + (\alpha - \beta). \tag{4.47}$$

Theorem 4.4.4 shows an important property of approximating the cumulative probability function by samples. Set $\gamma \approx 0$ and $\beta \approx \alpha$, if the sample number goes to ∞, $\Pr\{\tilde{F}^N(-\gamma, u) \geq 1 - \beta\}$ goes to 0. Namely, the sample-based neural approximation problem has the solution which satisfies the original chance constraint in a higher probability with more sample numbers.

4.4.3 Proposed algorithms for solving chance constrained optimization

Two algorithms are used to solve chance constrained optimization. First, sample-based algorithm is summarized to train the deterministic constraints which has a neural network form. Then, a normal algorithm for nonconvex program can be applied to solve the

deterministic program obtained by the neural approximation approach. The algorithm for nonconvex program can be either the Sequential Quadratic Program (SQP) method for regular nonlinear programs or simulated annealing algorithm. More details for SQP method and simulated annealing algorithm can be found in *Nocedal and Wright* [2006] and *Brooks and Morgan* [1995], *Dekkers and Aarts* [1991], *Ingber* [1993], *Kirkpatrick et al.* [1983], *Szu and Hartley* [1987], respectively.

Due to the discussion in the previous subsections, the algorithm for training $\hat{H}_S(u)$ is summarized as followings:

1. Generate the sample set of uncertain parameter vector $\Delta^N = \{\delta_1, ..., \delta_N\}$ by extracting samples independently from sample space Δ according to the identical distribution $p_\delta(\delta)$;
2. Generate the sample set of decision variable $\mathcal{U}^K = \{u_1, ..., u_K\}$ by extracting samples independently from feasible domain $\mathcal{U}$ according to uniform distribution;
3. For all $u_k \in \mathcal{U}^K$, calculate the $(1-\alpha)$-empirical quantile at u_k, $\tilde{Q}^{1-\alpha}(\bar{h}(u_k, \delta))$, using Eq. (4.31) and obtain the output sequence

$$Y^K = \{\tilde{Q}^{1-\alpha}(\bar{h}(u_1, \delta)), ..., \tilde{Q}^{1-\alpha}(\bar{h}(u_K, \delta))\}; \tag{4.48}$$

4. Train β, a, b, c in Eq. (4.34) based on $\mathcal{U}^K$ and Y^K by Extreme Learning Machine (ELM) algorithm which is introduced in *Huang et al.* [2006] and summarized in Appendix A.

After using neural approximation approach, we obtain the original problem with deterministic constraints. The deterministic constraints has the neural-network formulation.

SQP-based Algorithm. Assume that $\mathcal{U}$ is constrained by $g(u) \leq 0$ where $g : \mathbb{R}^n \to \mathbb{R}^c$. We denote the Lagrangian for the approximate problem as:

$$\mathcal{L}(u, \lambda_g, \lambda_H) = J(u) + \lambda_g g(u) + \lambda_H \hat{H}_S(u). \tag{4.49}$$

The flow of the algorithm is summarized as

1. Initialize a decision variable $u^0, \lambda_g^0, \lambda_H^0$ and set $k = 0$;
2. Calculate $J(u^k)$, $\bigtriangledown J(u^k)$, $\bigtriangledown g(u^k)$, $\bigtriangledown \hat{H}_S(u^k)$ and $\bigtriangledown^2_{uu}\mathcal{L}(u^k, \lambda_g^k, \lambda_H^k)$;

3. Solve the following problem and obtain a search direction d^k:

$$\begin{aligned} \min_d \; & J(u^k) + \bigtriangledown J(u^k)^T d + \frac{1}{2} d^T \bigtriangledown^2_{uu} \mathcal{L}(u^k, \lambda_g^k, \lambda_H^k) d \\ s.t. \quad & \hat{H}_S(u^k) + \bigtriangledown \hat{H}_S(u^k)^T d \leq 0, \\ & g(u^k) + \bigtriangledown g(u^k)^T d \leq 0; \end{aligned} \tag{4.50}$$

4. Check whether $\|d^k\| \leq \epsilon_d$ where ϵ_d is a sufficiently small positive number. If $\|d^k\| \leq \epsilon_d$, stop the algorithm and output $u^* = u^k$. Otherwise, update $u^{k+1} = u^k + d^k, k = k + 1$ and go back to step 2.

Simulated Annealing-based Algorithm. The original edition of simulated annealing algorithm is not presented here which can be found in [*Brooks and Morgan*, 1995, *Szu and Hartley*, 1987]. After minor revision on the original algorithm, it can be used to solve the deterministic constrained problem. The algorithm is summarized as

1. Initialize a temperature T_0, a decision value $u^* \in \mathcal{U}$ and calculate the cost value as

$$E^* = J(u^*) + C(\hat{H}_S(u^*)), \tag{4.51}$$

   where

$$C(u^*) = \begin{cases} 0, & \text{if } \hat{H}_S(u^*) \leq 0 \\ \bar{C}, & \text{if } \hat{H}_S(u^*) > 0. \end{cases} \tag{4.52}$$

   Here, $\bar{C}$ should be chosen as a very larger value;

2. For iteration $m = 1, 2, ..., M$, do the following step iteratively

   a) Randomly select another point $u_m \in \mathcal{V}_r(u^*) = \{u_m \in \mathcal{U} | \|u_m - u^*\| \leq r\}$. $\mathcal{V}_r(u^*)$ a neighbourhood of the previous point and r is the radius. Calculate the corresponding cost value E_m similarly as Eq. (4.52);

   b) Calculate

$$\bigtriangledown E_m = E_m - E^*, \tag{4.53}$$

   c) Move to the new point by setting $u_m = u^*$ if a random variable μ distributed uniformly over (0,1), satisfies

$$\mu \leq e^{-\frac{\bigtriangledown E_m}{T_{m-1}}} \tag{4.54}$$

   or equivalently

$$\bigtriangledown E_m \leq -T_{m-1} \log \mu; \tag{4.55}$$

d) Update the temperature as

$$T_m = \begin{cases} T_{m-1}, & \text{if (4.54) holds} \\ \rho \cdot T_{m-1}, & \text{if (4.54) doesn't hold.} \end{cases} \tag{4.56}$$

Here, $\rho \in (T_{min}/T_0, 1)$.

e) Terminate the algorithm if $T_m < T_{min}$ where T_{min} is the lower boundary for the temperature or $m = M$.

4.5 Numerical Example

This section presents numerical simulation for verifying the proposed method. The following points are addressed in the verification:

- The comparison of using cumulative probability and quantile function instead of the chance constraint;
- The comparison of using different algorithms to solve the approximate problem.

4.5.1 Simulation model

The targeted problem in the numerical simulation is a non-convex program with chance constraints. The decision domain is $\mathcal{U} = [-6, 6]^2$. The cost function is

$$J(u) = \frac{\sum_{i=1}^{2}(u_i + 0.5)^4 - 30u_i^2 - 20u_i}{100}. \tag{4.57}$$

The constrained function is

$$h(u, \delta) = (\sum_{i=1}^{2} 0.125 * (u_i - 2\delta)^4 - 3.5 * (u_i - 2\delta)^2) - (8 - 0.1\delta)^2, \tag{4.58}$$

where δ is random variable which obeys normal distribution $\mathcal{N}(0, 1)$. Moreover, the violation probability level is $\alpha = 0.05$.

The image of the objective function and bound defined by violation probability is plotted in Fig. 4.3.

4.5.2 Simulation results

We compared the following methods in the simulation validation:

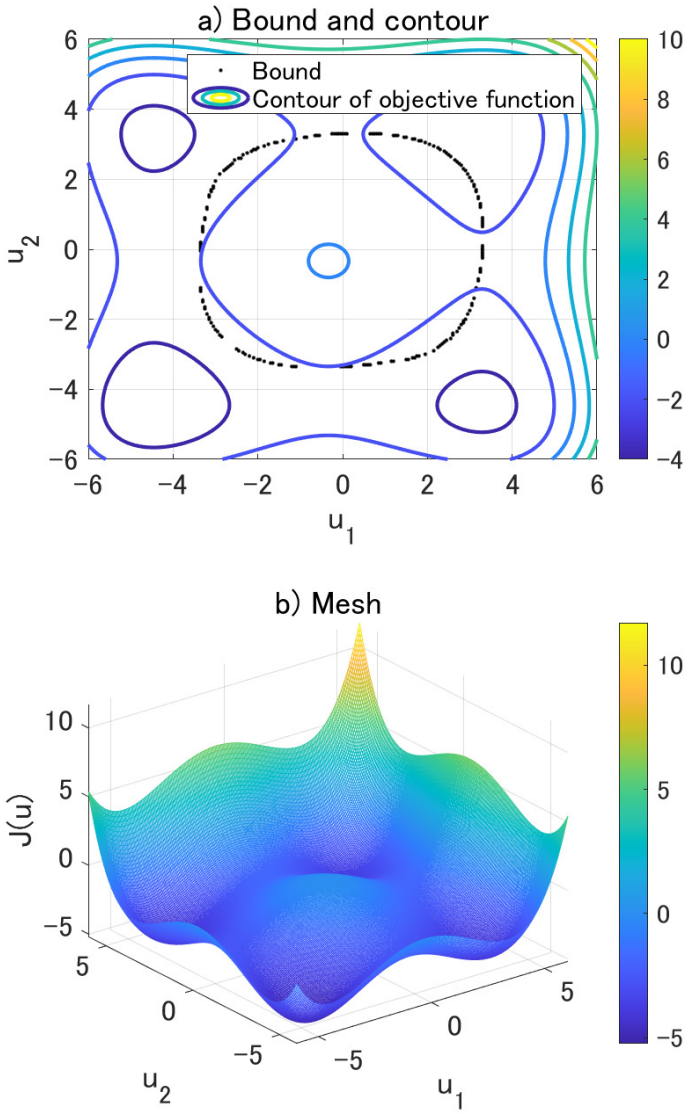

FIGURE 4.3: Objective function and bound of the simulation model: a) Bound and contour.

- S.A.(c): the algorithm for solving the nonlinear program adopts simulated annealing-based algorithm. The approximate constraint for chance constraint uses cumulative probability function;
- S.A.(q): the algorithm for solving the nonlinear program adopts simulated annealing-based algorithm. The approximate constraint for chance constraint uses quantile function;
- SQP(c): the algorithm for solving the nonlinear program adopts SQP algorithm. The approximate constraint for chance constraint uses cumulative probability function;

- SQP(q): the algorithm for solving the nonlinear program adopts SQP algorithm. The approximate constraint for chance constraint uses quantile function.

In the simulation, for each method, we consider different sample numbers of δ: 500, 1000, 2000, 3000, 4000, 5000. For each method under a given sample number of δ, the calculation of optimal solution is repeated for 200 times. The samples were randomly chosen in each time.

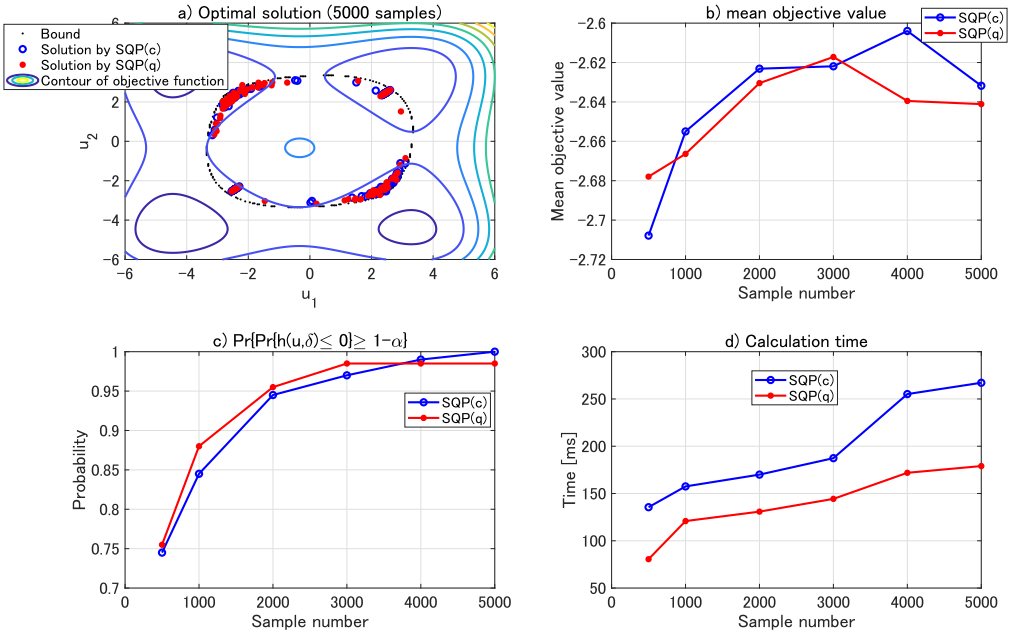

FIGURE 4.4: Simulation results of SQP method.

Fig. 4.4 shows the comparison between SQP(c) and SQP(p). Fig. 4.4 a) gives the plots of the optimal solutions given by SQP(c) and SQP(p) when 5000 samples of δ were chosen. The solutions are similarly distributed around the bound. The mean objective values shown in Fig. 4.4 b) and the violation probability shown in Fig. 4.4 c) also validates that using the cumulative probability function and quantile function does not influence the final results in this example. However, using the cumulative probability function increases the calculation time which is shown in Fig. 4.4 d).

Fig. 4.5 shows the comparison between S.A.(c) and S.A.(p). Fig. 4.5 a) gives the plots of the optimal solutions given by S.A.(c) and S.A.(p) when 5000 samples of δ were chosen. The solutions are similarly distributed around the bound. The mean objective values shown in Fig. 4.5 b) and the violation probability shown in Fig. 4.5 c) also validates that using the cumulative probability function and quantile function does not influence the final results in this example. Different from the case of using SQP algorithm, using

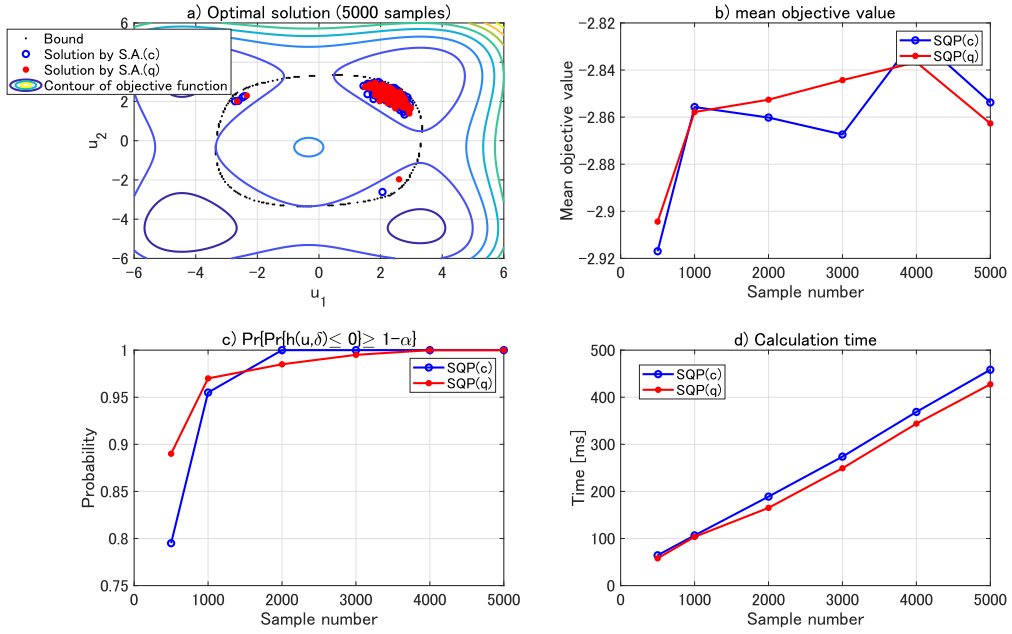

FIGURE 4.5: Simulation results of simulated annealing method.

the cumulative probability function does not increase the calculation time when using simulated annealling which is shown in Fig. 4.5 d).

Compared to SQP method, using simulated annealing method costs more time to calculate the optimal solution. While, the objective value is improved and the solution locates near the optimal position with high probability.

4.6 Application to Interval Predictor Model of Wind Power

Predictor model of wind power is used to predict the wind power based on wind speed measurement [*Ouyang et al.*, 2017]. Fig. 4.6 illustrates the power curve constructed from the industrial data. The data comes from a large wind farm located in Jiugongshan, Hubei, China. The data set was collected at turbines at a sampling interval of 10 min. In total 56, 618 data points were collected from March 17, 2009, to April 17, 2009. The unit of the active wind power is kW, and the value of power is normalized for air density of 1.18 kg/m^3.

Piecewise models are often used to improve the prediction accuracy [*Khalfallah and Koliub*, 2007, *Lydia et al.*, 2013]. The wind speed range is discretized into intervals, and the corresponding wind speed and power data make the partitions. Supposing the cut-out speed is v_{co}, the speed range $[0, v_{co}]$ is divided into N_w equal length intervals.

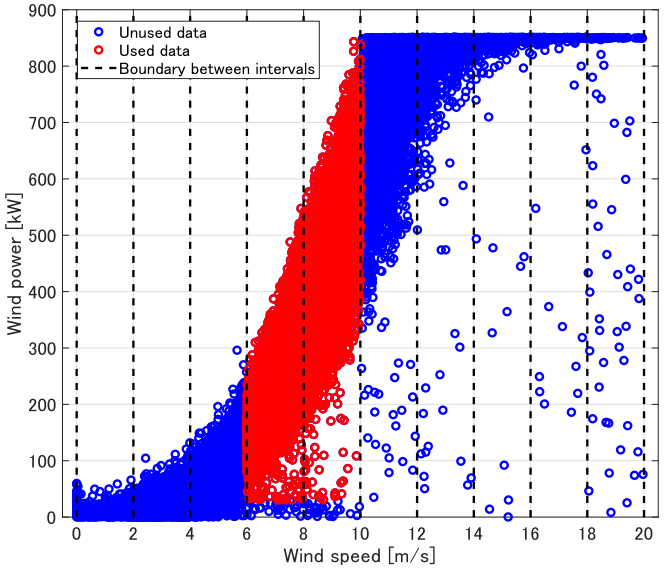

FIGURE 4.6: Power curve of an industrial wind turbine.

The data points in each interval are defined as

$$D_i = \{(v, p(v)) \in \mathbb{R}^2 | v \in [v_i, v_{i+1})\}, i = 1, ..., N_w \tag{4.59}$$

where D_i represents the points set of the $i-$th partion, v represents wind speed and $p(v)$ is the corresponding wind power, and $v_i = \frac{i-1}{N_w} v_{co}$ is the demarcation speed between the $i-$th and $(i-1)$-th partition. In this study, v_{co} is chosen as 20, and N_w is 10, as shown in Fig. 4.6. Besides, We do not establish models for all partitions in this study. As an example to demonstrate the proposed method. The data in partition 4 and 5 is used as an unity, which are marked red circles in Fig. 4.6. Notice that some under-power points or stopping points are wiped out by the data pre-process method introduced in Section 3 of *Ouyang et al.* [2017]. For convenience, denote the used dataset as D.

Consider a system with the mechanism or model from the input to output written as

$$y = f(\varphi, \delta_y) \tag{4.60}$$

where $y \in \mathbb{R}$ denotes the system output and $\varphi \in \mathbb{R}^{n_\varphi}$ is a regression vector containing input variables $u \in \mathbb{R}^{n_u}$ or decision variables on which the system output y depends. δ_y denotes the uncertain parameter vector and $\delta_y \in \Delta_y \subseteq \mathbb{R}^{n_{\delta_y}}$. Besides, assume that probability measure is well defined on the the sigma algebra of Δ_y, $\mathcal{B}_{\Delta_y}$. Due to the existence of δ_y, for a fixed φ, y is supposed to be located in a sample space with well defined sigma algebra and probability measure. Let $\mathcal{Y}(\varphi)$ be the sample space and $p_y(y|\varphi)$ be the probability density function conditioned on φ. Standard predictor models

can only give a specific value of y for a φ. However, the probabilistic interval of y conditioned on φ is important for various applications.

Differently from standard predictor models, interval predictor models return a prediction interval instead of a single prediction value [*Campi et al.*, 2009]. As defined in *Campi et al.* [2009], an interval predictor model is a set-valued map

$$I : \varphi \to I(\varphi) \subseteq Y \subset \mathbb{R} \tag{4.61}$$

where $\varphi \in \mathbb{R}^{n_\varphi}$ is a regression vector containing input variables $u \in \mathbb{R}^{n_u}$ or decision variables on which the system output y depends. Given an observed φ, $I(\varphi)$ is an informative interval or the prediction interval, containing y with a given guaranteed probability $1 - \alpha, \alpha \in [0, 1)$. Output intervals are obtained by considering the span of parametric families of functions. Here, we consider the family of linear regression functions defined by

$$\mathcal{M} = \{y = \theta^T \varphi + e, \theta \in \Theta \subseteq \mathbb{R}^{n_\theta}, \|e\| \leq \varepsilon \in \mathbb{R}\}. \tag{4.62}$$

Then, a parametric (Interval Predictor Model) IPM is obtained by associating to each φ the set of all possible outputs given by $\theta^T \varphi + e$ as θ varies over Θ and e varies over $\mathcal{E} = \{e \in \mathbb{R} | \|e\| \leq \varepsilon\}$, is defined as

$$I(\varphi) = \{y : y = \theta^T \varphi + e, \forall \theta \in \Theta, \forall e \in \mathcal{E}\}. \tag{4.63}$$

A possible choice for the set Θ is a ball with center c and radius r:

$$\Theta = \mathcal{B}_{c,r} = \{\theta \in \mathbb{R}^{n_\theta} : \|\theta - c\| \leq r\}. \tag{4.64}$$

Then, the interval output of the IPM can be explicitly computed as

$$I(\varphi) = [c^T \varphi - (r\|\varphi\| + \varepsilon), c^T \varphi + (r\|\varphi\| + \varepsilon)]. \tag{4.65}$$

For the wind power prediction problem, the input is wind speed and output is wind power as $u = v$ and $y = p$. Besides, φ is chosen as $\varphi = [v\ v^2]^T$ in the wind power prediction problem for example. More advanced model of φ can be chosen for more accurate performance. For the simplicity of explanation, we use a relatively simple model for example. Identifying the IPM is essentially identifying (c, r, ε). Without loss of the generality, we use $I_{(c,r,\varepsilon)}(v)$ for $I(\varphi)$ in the later part for IPM identification problem of wind power prediction.

Due to the above discussions, the IPM identification problem of wind power prediction is to find IPM $I_{(c,r,\varepsilon)}(v)$ (or find (c, r, ε))such that

$$\min_{(c,r,\varepsilon)} wr + \varepsilon \tag{4.66}$$

subject to

$$\begin{aligned} r, \varepsilon \geq 0, \\ \Pr\{p \in I_{c,r,\varepsilon}(v)\} \geq 1 - \alpha, \alpha \in [0, 1). \end{aligned} \tag{4.67}$$

Notice that w is a weight and often chosen as $w = \mathbb{E}\{\|\varphi\|\}$ [*Campi et al.*, 2009]. In this study, w is chosen as 20.

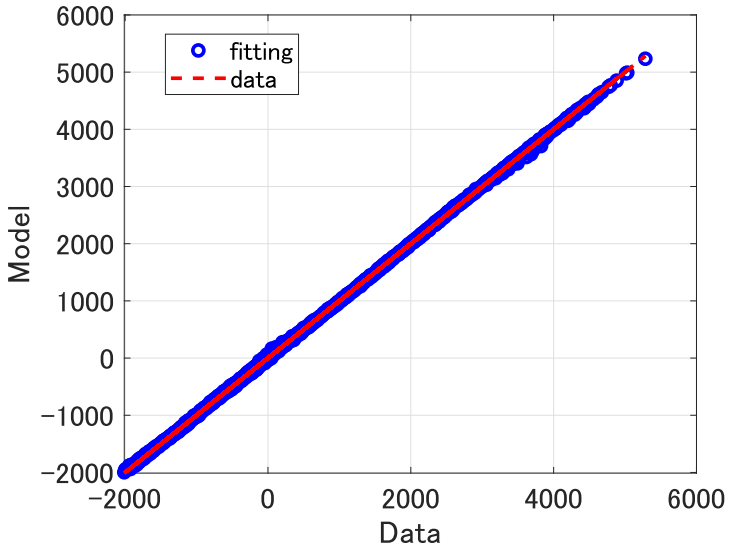

FIGURE 4.7: Estimation results of quantile function by using neural networks.

4.6.1 Results and Discussions

Fig. 4.7 shows one example of the estimation results of quantile function by using neural networks. 5000 different combinations of (c, r, ε) are considered. The data set was divided into a train set and a test set. The blue circles shows the quantile function value given by the trained fitting model. The red dotted line shows the result of the empirical quantile function value calculated by using the data of the test set. The mean value of error is -0.0421 and the mean of the abstract value of mean is 12.5882.

We compare the proposed method with scenario approach which is proposed in **?**. The required α is set as 0.05. Fig. 4.8 and Fig. 4.9 shows the results of interval predictions by scenario approach and the proposed method, respectively. The number of used data samples is 150, 750 or 7500. As the number of samples increases, the scenario approach

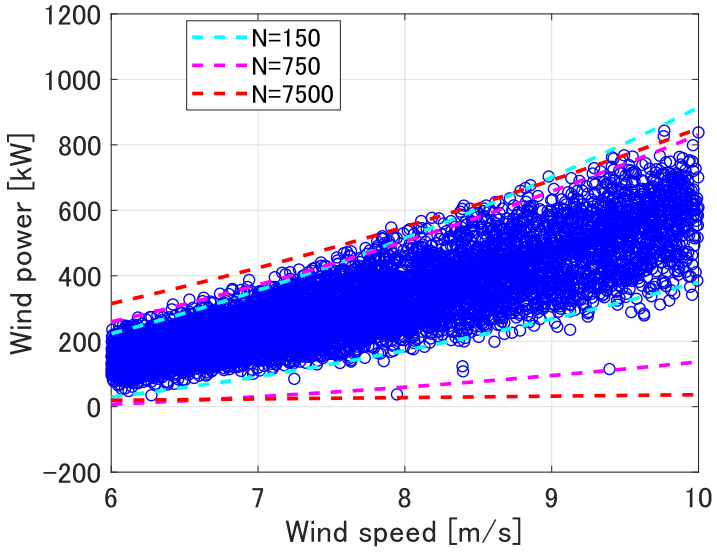

FIGURE 4.8: Results of interval predictions by scenario approach using different choices of sample number.

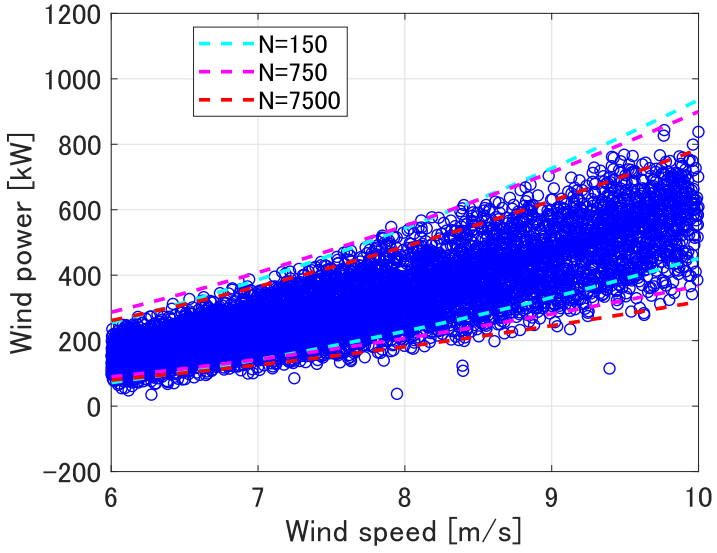

FIGURE 4.9: Results of interval predictions by neural approximation approach using different choices of sample number. The limitation of violation probability is $\alpha = 0.05$.

gives more conservative results while the proposed method performs with better robust on sample numbers.

A more comprehensive validation was conducted through the posterior Monte Carlo analysis. In the validation, the number of data sample was increased from 150 to 7500. For each number of data samples, 500 times sampling and calculation of c, r, ε processes were done. For instance, if we set the number of data samples as 150. Then, 500 times of extracting 150 samples of (v, p) from the training set were done. For each extracted 150 samples, the corresponding c, r, ε was calculated by scenario approach or

TABLE 4.1: The probability that constraint failure probability $\alpha > 0.05$: $\Pr\{\alpha > 0.05\}$

N	150	375	750	1500	3750	7500
SA	0.224	0.004	0	0	0	0
Proposed	0.156	0.07	0.006	0	0	0

TABLE 4.2: The mean value of cost function

N	150	375	750	1500	3750	7500
SA	61.56	70.47	74.72	77.58	79.95	80.93
Proposed	50.25	50.45	50.69	50.97	51.33	51.72

proposed method. The statistical analysis results are summarized in Tab. 1 and Tab. 2. Apparently, the proposed method can achieve a better trade-off between probability constraint and cost value.

Chapter 5

Predictive Probabilistic Bounds on State Trajectories for Uncertain Nonlinear Systems

The precise bounds for the evolution of state variables in uncertain systems are useful for the applications related to uncertain dynamical systems such as anomaly detection, safety assessment and robust control design. However, the computation of the tight system state bounds for uncertain nonlinear system is a challenging task due to the NP-hard issue. This chapter proposes a sample average method to approximately compute the probabilistic ellipsoidal bounds for uncertain nonlinear systems, which is an improvement of the scenario approach-based method. The probabilistic ellipsoidal bound computation of discrete-time nonlinear systems is formulated as a probabilistic constrained problem. Then, the approximate solution of the probabilistic constrained problem is obtained by solving a sample average-based approximate optimization problems with N samples of uncertain variables from the sample space. The proposed method is compared with the scenario approach-based method in a numerical simulation. The results show that the proposed method can obtained a tighter bound than the scenario approach-based method. The conservatism issue of the scenario approach-based method has been dramatically improved.

5.1 Introduction

5.1.1 Motivations

Probabilistic bounds on the evolution of the states of uncertain dynamical systems play a crucial role in many application problems such as anomaly detection, safety assessment and robust control design[*Blanke et al.*, 2006, *Nagy and Braatz*, 2003]. For example, *Liu et al.* [2020] used the probabilistic region of the trajectory to detect the fault in the industrial wireless sensor networks. In *Yu and Chen* [2019], Markov boundary outliers is formulated for anomaly detection, which is essentially the outer bounds of the hidden state. *Esfahani and Lygeros* [2016] showed that the probabilistic outer bound of state trajectory can improve the accuracy of fault detection in uncertain nonlinear dynamical systems. *Rostampour et al.* [2017] discussed that the optimal fault detection requires the information of probabilistic bound of system trajectory. In the robust fault estimation proposed in *Wan et al.* [2016, 2020], probabilistic bounds of the state trajectories are used. Another general examples is the stochastic model predictive control [*Farina et al.*, 2016, *Mesbah*, 2016, *Moser et al.*, 2018]. The performances of model predictive control on both objective and constraints' satisfaction will be improved if precise probabilistic evolution of the state trajectories.

However, the computation of probabilistic bounds on the state is challenging due to the NP hard issue. Efficient algorithm to solve the problem approximately is required.

5.1.2 Background and related works

In recent years, computation of the probabilistic bounds of state evolution in uncertain dynamic systems has attracted a lot of attentions. For instance, *Alamo et al.* [2008], *Witsenhausen* [1968] proposed set-based approaches to roughly approximate the bounds. *Prajna* [2006] used barrier certificate approach to address the computation. *Kishida and Braatz* [2015] applied linear matrix inequalities method to design the algorithm of computing the probabilistic bounds in linear systems. For polynomial systems, *Kishida et al.* [2014] proposed algorithms based on skewed structured singular value approach. These methods achieve good performance in linear systems or polynomial systems. However, the above methods cannot be extended to general nonlinear systems.

Esfahani and Lygeros [2016] applied randomized optimization to address the outer bound estimation in the nonlinear system. *Shen et al.* [2020] proposed scenario approach-based algorithm to compute probabilistic bound of state trajectories for general uncertain nonlinear systems. However, these two methods still cannot give the exact probabilistic

bound since the robust optimization is used and the computed bounds converge to the completely robust bound not the given probability level.

5.1.3 Key contributions of this chapter

This chapter proposes a new approach for computing probabilistic ellipsoidal bounds for uncertain nonlinear discrete-time systems, which gives the exact probabilistic bound instead of the completely robust bound. The main contributions are summarized as

- A probabilistic constrained optimization problem is formulated to calculate the ellipsoidal bounds for future system states in general nonlinear discrete-time systems;
- The formulated problem is NP-hard. To solve the NP-hard problem, the sample average approach (proposed in *Luedtke and Ahmed* [2008]) is applied to obtain a deterministic optimization problem through sample extraction of the uncertain parameters, disturbances, and initial states. We extend the results in *Luedtke and Ahmed* [2008] to the problem of obtaining probabilistic ellipsoidal bounds for future system states;
- An algorithm is proposed to computing probabilistic ellipsoidal bounds for nonlinear discrete-time systems;
- The feasibility and optimality of the proposed method are validated in a numerical example with a comparison with scenario approach-based method proposed in *Shen et al.* [2020].

5.2 Problem Description

Consider a discrete-time system expressed as

$$\begin{aligned} &x_k = f(x_{k-1}, \omega_{k-1}), \\ &\omega_{k-1} \in \Delta_\omega \subset \mathbb{R}^{n_\omega}, \\ &x_0 \in \Delta_{x_0} \subset \mathbb{R}^{n_x} \end{aligned} \tag{5.1}$$

where x_0 and ω_{k-1} denote uncertain initial state vector and uncertain parameters vector separately, the unbounded sample space of x_0 is denoted as Δ_{x_0} with probability measure $\Pr_{x_0}$ on it, Δ_ω denotes the unbounded sample space for ω_{k-1} for any $k \in \{1, 2, 3, ...\}$

on which there is a probability measure $\Pr_\omega$, the system state vector at time step $k \in \{1, 2, 3, ...\}$ is denoted as x_k, and $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_\omega} \to \mathbb{R}^{n_x}$ is convex, continuous and differentiable on $\mathbb{R}^{n_x} \times \mathbb{R}^{n_\omega}$.

Obviously, due to the uncertain initial state x_0 and uncertain parameter vectors $\omega_0, ..., \omega_{k-1}$, the state vector at k-step x_k is uncertain state as well. The all possible instances for x_k form the sample space of x_k denoted as $\Delta_{x_k} \subset \mathbb{R}^{n_x}$.

Problem 5.2.1. Given a system of the form described in (5.1), find the tightest possible outer bounds on the trajectory x_k for any $k \in \{1, 2, 3, ...\}$ to make sure that the probability that x_k locates in the bounds is larger than a given probability level. The problem can be formulated as

$$\begin{aligned} &\min_{U_k \in \mathcal{U}_k} \det M_k^{-1} \\ &s.t. \quad x_k = f(x_{k-1}, \omega_{k-1}), \\ &\qquad x_0 \in \Delta_{x_0} \subset \mathbb{R}^{n_x}, \\ &\qquad \omega_{k-1} \in \Delta_\omega \subset \mathbb{R}^{n_\omega}, \\ &\qquad \Pr\{(x_k - C_k)^T M_k (x_k - C_k) \leq 1\} \geq 1 - \alpha \end{aligned} \tag{5.2}$$

where M_k is a positive definite matrix with n_x columns and n_x raws, C_k is a n_x-dimension vector, and $U_k = \{C_k, M_k\}$ is the input variable for the problem.

Problem 5.2.1 is an NP-hard problem due to the existence of chance constraints. This work addresses Problem 5.2.1 with the scenario approach. Besides, the measure of violation of the constraint for a given U_k is defined by

$$V(U_k) \doteq \Pr\{(x_k - C_k)^T M_k (x_k - C_k) > 1\}. \tag{5.3}$$

The chance constraint can be expressed as $V(U_k) \leq \alpha$. The feasible domain of problem 5.2.1 is written as

$$\mathcal{U}_{f,k} = \{U_k \in \mathcal{U}_k : V(U_k) \leq \alpha\}. \tag{5.4}$$

5.3 Scenario Approach-based Method

5.3.1 Mathematical Preliminaries

Consider an uncertain convex optimization problem with chance constraint as

$$\begin{aligned} &\min_{u\in\mathscr{U}} J(u) \\ &s.t. \quad \Pr\{h(u,\delta)\leq 0\}\geq 1-\alpha,\ \delta\in\Delta,\ \alpha\in(0,1) \end{aligned} \tag{5.5}$$

where $u\in\mathscr{U}\subset\mathbb{R}^{n_u}$ is the control variable or the decision variable, the decision variable set $\mathscr{U}$ is convex and closed, $\delta\in\Delta\subset\mathbb{R}^{n_\delta}$ is an uncertain parameter, the set Δ is the sample space of δ and Pr is a probability measure on Δ, α is a given probability level, moreover, for any fixed value $\delta\in\Delta$, both $J(u):\mathscr{U}\ \rightarrow\ \mathbb{R}$ and $h(u,\delta):\mathscr{U}\times\Delta\ \rightarrow\ \mathbb{R}$ are continuous and convex in u. The constraint of program (5.5) is a chance constraint.

Definition 5.3.1. The measure of violation of the constraint in (5.5) for a given u is defined by

$$V(u)\doteq\Pr\{\delta\in\Delta : h(u,\delta)>0\}. \tag{5.6}$$

Then, the chance constraint can also be expressed as

$$V(u)\leq\alpha. \tag{5.7}$$

Obviously, due to the chance constraint, the program (5.5) is NP hard *Campi and Garatti* [2008].

If independent samples $\delta^{(i)}, i=1,...,N$ is identically extracted from Δ according to probability measure Pr, a deterministic convex optimization problem, or called Robust Convex Program with N constraints(RCP(N)), can be formed as

$$\begin{aligned} &\min_{u\in\mathscr{U}} J(u) \\ &s.t. \quad h(u,\delta^{(i)})\leq 0,\ i=1,...,N \end{aligned} \tag{5.8}$$

which is a standard convex finite optimization problem and a solution can be found at low computational cost by available solvers when N is not too large **?**. The optimal solution $\hat{u}_N$ of the program (5.8) is called as the scenario solution for program (5.5) generally. Moreover, since the extractions $\delta^{(i)}, i=1,...,N$ is randomly dawed, the optimal solution $\hat{u}_N$ is random variable. On the other hand, the degree of robustness and optimality of $\hat{u}_N$ should be investigated, precisely, the measure that $\hat{u}_N$ satisfies the chance constraint described by (5.7). The following theorem shows that $V(\hat{u}_N)$ tends rapidly to zero as N increases and the optimality of $\hat{u}_N$ is also assured in a probabilistic level.

Theorem 5.3.1. Let $\alpha\in(0,1),\beta\in(0,1)$ be two given probability levels. If

$$N\geq\frac{2}{\alpha}\ln\frac{1}{\beta}+2n_u+\frac{2n_u}{\alpha}\ln\frac{2}{\alpha}, \tag{5.9}$$

then, it holds that

$$\Pr\{(\delta^{(1)}, ..., \delta^{(N)}) \in \Delta^N : V(\hat{u}_N) \leq \alpha\} \geq 1 - \beta. \tag{5.10}$$

Besides, let $J_{\mathrm{PCP}(\alpha)}$ denote the optimal objective value of the probabilistic constrained problem PCP(α) in (5.5) when it is feasible (i.e., $J_{\mathrm{PCP}(\alpha)} = \inf_{u \in \mathcal{U}} J(u)$ subject to $V(u) \leq \alpha$) and let $J_{\mathrm{RCP}(N)} = J(\hat{u}_N)$ be the optimal objective value of the deterministic convex optimization problem RCP(N) in (5.8) when it is feasible(notice that $J_{\mathrm{RCP}(N)}$ is a random variable due to the randomness of $\hat{u}_N$, while $J_{\mathrm{PCP}(\alpha)}$ is a deterministic value), with N any number satisfying (5.9). Then,

1) with probability at least $1 - \beta$, if RCP(N) is feasible it holds that

$$J_{\mathrm{RCP}(N)} \geq J_{\mathrm{PCP}(\alpha)}; \tag{5.11}$$

2) assume PCP(α_1) is feasible, where $\alpha_1 = 1 - (1 - \beta)^{1/N}$. With probability at least $1 - \beta$, it holds that

$$J_{\mathrm{RCP}(N)} \leq J_{\mathrm{PCP}(\alpha_1)}. \tag{5.12}$$

The proof of Theorem 5.3.1 can be referred to the proofs of Theorem 1 and Theorem 2 in *Calafiore and Campi* [2006]. For the feasibility of scenario approach, Theorem 5.3.1 indicates that the scenario approach cannot be robust against all situations while the level of robustness can be retained. Note that β is an important factor and choosing $\beta = 0$ makes $N = \infty$. However, for practical purposes, β has very limited importance due to its appearance under the sign of logarithm: although β is selected as 10^{-10} which is near zero in practical purpose, N does not grow significantly. On the other hand, Theorem 5.3.1 also gives the optimality of the obtained approximate solution for original problem by scenario approach. In a probabilistic level, the optimal objective value by using the approximate solution is bounded by (5.11) and (5.12) since the approximate solution and the optimal objective value are random variable due to the samples are randomly chosen.

5.3.2 Main result

In this section, the approach to solve the Problem 5.2.1 at each time instant is proposed based on the scenario approach. The main result of this study is summarized in the following theorem.

Theorem 5.3.2. Denote $x_0^{(i)}, \omega^{(i)}, i = 1, ..., N$ for the samples of x_0, ω extracted from Δ_{x_0} and Δ_ω independently and identically. Calculate the samples of $x_k^{(i)}, k = 1, 2, 3, ..., i \in$

$\{1, ..., N\}$ recursively through

$$\begin{aligned} &\forall i \in 1, ..., N \\ &x_1^{(i)} = f(x_0^{(i)}, \omega^{(i)}), \\ &..... \\ &x_k^{(i)} = f(x_{k-1}^{(i)}, \omega^{(i)}), \\ &..... \end{aligned} \tag{5.13}$$

Then, if

$$N \geq \frac{2}{\alpha} \ln \frac{1}{\beta} + 2(n_x^2 + n_x) + \frac{2(n_x^2 + n_x)}{\alpha} \ln \frac{2}{\alpha}, \tag{5.14}$$

$\forall k = 1, 2, 3, ...,$ the optimal solution $\hat{U}_k(N) = \{\hat{M}_k(N), \hat{C}_k(N)\}$ of the deterministic problem

$$\begin{aligned} &\min_{U_k} \det M_k^{-1} \\ &s.t. \quad \forall i \in \{1, ..., N\}, \\ &\qquad (x_k^{(i)} - C_k)^T M_k (x_k^{(i)} - C_k) \leq 1. \end{aligned} \tag{5.15}$$

satisfies

$$\Pr\{V(\hat{U}_k(N)) \leq \alpha\} \geq 1 - \beta, \tag{5.16}$$

$$\Pr\{\det \hat{M}_k^{-1}(N) \geq m_{min}^k\} \geq 1 - \beta, \tag{5.17}$$

$$\Pr\{\det \hat{M}_k^{-1}(N) \leq m_{max}^k\} \geq 1 - \beta. \tag{5.18}$$

Here, denote the solution of (5.2) as $U_k(\alpha) = \{M_k(\alpha), C_k(\alpha)\}$ for α and $U_k(\alpha_1) = \{M_k(\alpha_1), C_k(\alpha_1)\}$ for $\alpha_1 = 1-(1-\beta)^{1/N}$ $m_{min}^k = \det M_k^{-1}(\alpha)$ and $m_{max}^k = \det M_k^{-1}(\alpha_1)$.

The approach starts calculating x_1 from an uncertain initial state x_0 and is repeatedly performed to calculate the probabilistic ellipsoidal bound of the system states' future evolution. The obtained solution is with "β"-sense robustness and optimality for the original Problem 5.2.1.

5.3.3 Proof for the main result

First, the property of $x_k^{(i)}$ calculated by (5.26) should be addressed.

Lemma 5.3.1. $x_k^{(i)}$ calculated by (5.26) can be regarded as samples randomly extracted from Δ_{x_k}.

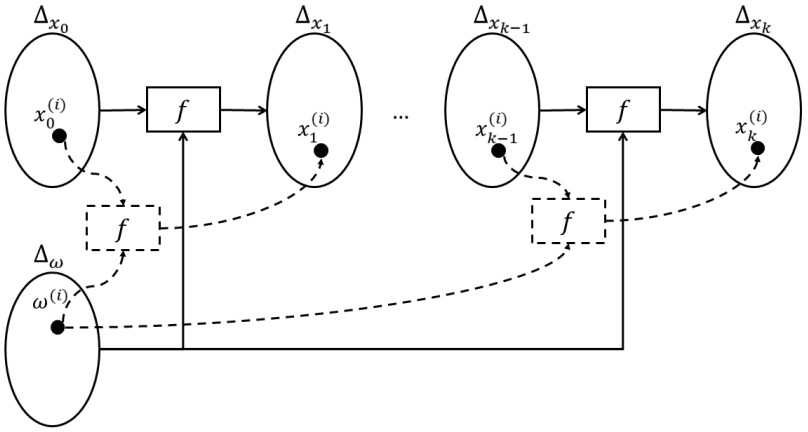

FIGURE 5.1: Intuitive illustration of Lemma 5.3.1.

Proof. (Lemma 5.3.1). When $k = 1$, for any $i \in 1, .., N$, $x_1^{(i)}$ are calculated from

$$x_1^{(i)} = f(x_0^{(i)}, \omega^{(i)}). \tag{5.19}$$

Note that $x_0^{(i)} \in \Delta_{x_0}, \omega^{(i)} \in \Delta_\omega, \forall i \in \{1, ..., N\}$ holds and $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_\omega} \to \mathbb{R}^{n_x}$ is a function from $\Delta_{x_0} \times \Delta_\omega$ to the space of x_1 denoted as Δ_{x_1}. Thus, $x_1^{(i)} \in \Delta_{x_1}$ holds. Moreover, $x_0^{(i)} \in \Delta_{x_0}, \omega^{(i)} \in \Delta_\omega$ are randomly selected and $x_1^{(i)}$ is consequently determined, which implies that $x_1^{(i)}$ is randomly selected from Δ_{x_1}. Therefore, Lemma 5.3.1 stands for $k = 1$. Assume that, Lemma 5.3.1 stands for $k = s > 1$. Then, as $k = s + 1$, $x_s^{(i)}$ can be calculated as

$$x_s^{(i)} = f(x_{s-1}^{(i)}, \omega^{(i)}),\ \omega^{(i)} \in \Delta_\omega. \tag{5.20}$$

Since $x_{s-1}^{(i)} \in \Delta_{x_{s-1}}, \omega^{(i)} \in \Delta_\omega, \forall i \in \{1, ..., N\}$, $x_s^{(i)} \in \Delta_{x_s}$ holds consequently due to f is a function to the space of x_s denoted as Δ_{x_s} from $\Delta_{x_s} \times \Delta_\omega$. Moreover, $x_{s-1}^{(i)} \in \Delta_{x_{s-1}}, \omega_{s-1}^{(i)} \in \Delta_\omega$ are randomly selected and $x_s^{(i)}$ is consequently calculated, which implies that $x_s^{(i)}$ is randomly selected from Δ_{x_s}. The assumption also holds for s. Then, $\forall k = 1, 2, 3, ...$, Lemma 5.3.1 holds and the proof ends. □

An intuitive illustration of Lemma 5.3.1 is given by Fig. 5.1.

With Lemma 5.3.1, the proof of Theorem 5.3.2 is summarized as following:

Proof. (Theorem 5.3.2). The input variables for the case of k are U_k which consists of C_k and M_k. The dimension for C_k is equal to the state vector's dimension n_x. Since M_k is a $n_x \times n_x$ matrix, the number of parameters to be decided is also n_x^2. Thus, the dimension of the input variables is $n_x^2 + n_x$. Besides, $x_k^i, i \in \{1, ..., N\}$ are randomly extracted.

Due to the above discussions, conclusion of Theorem 5.3.2 is proved according to Theorem 5.3.1. □

5.3.4 Proposed algorithm

Due to the above discussions, the proposed algorithm is summarized as

1. Generate $x_0^{(i)}$, $\omega^{(i)}$, $i = 1, ..., N$ from Δ_{x_0}. Generate $\omega^{(i)}, i = 1, ..., N$ from Δ_ω. Set $k = 0$;
2. Set $k = k + 1$, Calculate $x_k^{(i)}$ though $x_k^{(i)} = f(x_{k-1}^{(i)}, \omega^{(i)})$;
3. Solve problem (5.15) and obtain C_k, M_k;
4. Check whether $k = K$. If $k = K$, end and output $C_1, M_1, ..., C_K, M_K$. Otherwise, go back to step 2.

5.4 Sample Average Approach-based Method

5.4.1 Notations and problem reformulation

For given $\gamma > 0$, the cumulative probability function of $H(x_k, C_k, M_k, \gamma) := (x_k - C_k)^T M_k (x_k - C_k) - 1 + \gamma$ is denoted as

$$F(\gamma, U_k) = \Pr\{H(x_k, C_k, M_k, \gamma) \leq 0\}. \tag{5.21}$$

The sample set $\Delta_{x_k}^N = \{x_k^{(1)}, ..., x_k^{(N)}\}$ is obtained by extracting samples independently from the sample space of x_k: $\Delta_{x_k} = \{x_k \in \mathbb{R}^{n_x} | x_k = f(x_{k-1}, \omega_{k-1}), \forall x_{k-1} \in \Delta_{x_{k-1}}, \forall \omega_{k-1} \in \Delta_\omega\}$. Note that Δ_{x_k} is generated recursively from $k = 0$. For a given U_k, the empirical cumulative probability function of

$$\mathcal{H}_k^N = \{H(x_k^{(1)}, C_k, M_k), ..., H(x_k^{(N)}, C_k, M_k)\} \tag{5.22}$$

is written as

$$\tilde{F}(\gamma, U_k) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}(H(x_k^{(i)}, C_k, M_k, \gamma) \leq 0) \tag{5.23}$$

where $\mathbb{I}(\cdot)$ denotes the indicator function. $\mathbb{I}(\cdot) = 1$ if $\cdot$ is true, otherwise, $\mathbb{I}(\cdot) = 0$.

The empirical cumulative probability function can be used as the approximation of chance constraint and form the following approximate problem

$$\begin{aligned} &\min_{U_k \in \mathcal{U}_k} \det M_k^{-1} \\ &s.t. \quad x_k^{(i)} = f(x_{k-1}^{(i)}, \omega_{k-1}),\ \forall i \in \{1, ..., N\} \\ &\qquad x_0^{(i)} \in \Delta_{x_0} \subset \mathbb{R}^{n_x},\ \forall i \in \{1, ..., N\} \\ &\qquad \omega_{k-1} \in \Delta_\omega \subset \mathbb{R}^{n_\omega}, \\ &\qquad \tilde{F}(\gamma, U_k) \geq 1 - \epsilon, \epsilon \in [0, \alpha],\ \alpha \in [0, 1). \end{aligned} \tag{5.24}$$

The problem defined by (5.24) is the sample average approximation problem of problem 5.2.1. We can solve (5.24) to obtain the approximate solution of problem 5.2.1.

Define the feasible domain of (5.24) as

$$\mathcal{U}_{\epsilon,\gamma,k}^N = \{U_k \in \mathcal{U}_k : \tilde{F}(\gamma, U_k) \geq 1 - \epsilon\}. \tag{5.25}$$

5.4.2 Main result

The approach to solve the Problem 5.2.1 at each time instant is proposed based on the sample average method. The main result of this study is summarized in the following theorem.

Theorem 5.4.1. Suppose $\mathcal{U}_k$ is bounded with diameter D for every k. Namely, we have $D > 0$ such that $D = \sup\{\|U_k - U_k^+\|_\infty : U_k, U_k^* \in \mathcal{U}_k\}$. Lete $\epsilon \in [0, \alpha), \eta \in (0, \alpha - \epsilon)$ and $\gamma > 0$. Denote $x_0^{(i)}, \omega^{(i)}, i = 1, ..., N$ for the samples of x_0, ω extracted from Δ_{x_0} and Δ_ω independently and identically. Calculate the samples of $x_k^{(i)}, k = 1, 2, 3, ..., i \in \{1, ..., N\}$ recursively through

$$\begin{aligned} &\forall i \in 1, ..., N \\ &x_1^{(i)} = f(x_0^{(i)}, \omega^{(i)}), \\ &..... \\ &x_k^{(i)} = f(x_{k-1}^{(i)}, \omega^{(i)}), \\ &..... \end{aligned} \tag{5.26}$$

Then, we have

$$\Pr\{\mathcal{U}_{\epsilon,\gamma,k}^N \subseteq \mathcal{U}_{f,k}\} \leq 1 - \lceil \frac{1}{\eta} \rceil \lceil \frac{2LD}{\gamma} \rceil^{|U_k|} \exp\{-2N(\alpha - \epsilon - \eta)^2\}. \tag{5.27}$$

$\forall k = 1, 2, 3, ...$

With Lemma 5.3.1, the proof of Theorem 5.4.1 is summarized as following:

Proof. (Theorem 5.4.1). The objective function of (5.2) is continuous.

Besides, $(x_k - C_k)^T M_k (x_k - C_k)$ is continuous function of U_k for any $x_k \in \Delta_{x_k}$. $\mathcal{U}_k$ is compact. Thus, $(x_k - C_k)^T M_k (x_k - C_k)$ is Lipschitz continuous on $\mathcal{U}_k$ for any $x_k \in \Delta_{x_k}$, which means that Assumption 4.3.2 is satisfied.

The input variables for the case of k are U_k which consists of C_k and M_k. Besides, $x_k^i, i \in \{1, ..., N\}$ can be regarded as random variables from Δ_{x_k}. Due to the assumption that x_0 and ω have continuous distribution in Problem 5.2.1, x_k also has continuous distribution since it is recursively calculated from x_0 and ω_{k-1} via continuous function f. $(x_k - C_k)^T M_k (x_k - C_k)$ is also continuous function of x_k for any fixed U_k. The distribution of $(x_k - C_k)^T M_k (x_k - C_k)$ is continuous for any fixed U_k since x_k has continuous distribution.

Note that the probabilistic constraint of Problem 5.2.1 is equivalent to

$$F(0, U_k) \geq 1 - \alpha. \tag{5.28}$$

The value of $F(0, U_k)$ varies continuously on $\mathcal{U}_k$ due to the fact that continuous functions' function composition leads to continuous function. Assumption **??** holds.

Therefore, Problem 5.2.1 essentially has a continuous function as constraint. Since $\mathcal{U}_k$ is compact, $\mathcal{U}_{f,k}$ is also compact. Thus, Assumption **??** holds.

Due to the above discussions, conclusion of Theorem 5.4.1 is proved according to Theorem 4.3.2, which closes the proof. □

5.4.3 Proposed algorithm

The sample average approximate problem is essentially an integer program written as

$$\begin{aligned}
&\min_{U_k \in \mathcal{U}_k, y \in \mathbb{Z}^N} \det M_k^{-1} \\
&s.t. \quad x_k^{(i)} = f(x_{k-1}^{(i)}, \omega_{k-1}),\ \forall i \in \{1, ..., N\} \\
&\qquad x_0^{(i)} \in \Delta_{x_0} \subset \mathbb{R}^{n_x},\ \forall i \in \{1, ..., N\} \\
&\qquad \omega_{k-1} \in \Delta_\omega \subset \mathbb{R}^{n_\omega}, \\
&\qquad \frac{1}{N} \sum_{i=1}^{N} y(i) H(x_k^{(i)}, C_k, M_k, \gamma) \geq 1 - \epsilon, \epsilon \in [0, \alpha],\ \alpha \in [0, 1), \\
&\qquad y(i) \in \{0, 1\},\ \forall i \in \{1, ..., N\}.
\end{aligned} \tag{5.29}$$

Due to the above discussions, the proposed algorithm is summarized as

1. Generate $x_0^{(i)}$, $\omega^{(i)}$, $i = 1, ..., N$ from Δ_{x_0}. Generate $\omega^{(i)}, i = 1, ..., N$ from Δ_ω. Set $k = 0$;
2. Set $k = k + 1$, Calculate $x_k^{(i)}$ though $x_k^{(i)} = f(x_{k-1}^{(i)}, \omega^{(i)})$;
3. Solve problem (5.29) and obtain C_k, M_k;
4. Check whether $k = K$. If $k = K$, end and output $C_1, M_1, ..., C_K, M_K$. Otherwise, go back to step 2.

5.5 Numerical Example

In this section, an example of the implementation of the probabilistic ellipsoidal bounds calculation approach is presented.

5.5.1 System model for the numerical example

Consider the nonlinear model of air charging and engine speed dynamics in an engine powertrain system presented in *Shen et al.* [2020]. A discrete-time approximation of the continuous dynamic model is obtained by Euler discretization with a sample step of 0.02 s. Moreover, the uncertain parameters ω_1, ω_2 are denoted the parametric uncertainty. The resulting model is

$$\begin{bmatrix} x_{1,k} \\ x_{2,k} \end{bmatrix} = \begin{bmatrix} x_{1,k-1} \\ x_{2,k-1} \end{bmatrix} + 0.02 \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \tag{5.30}$$

where f_1 and f_2 are

$$f_1 = 5(\sqrt{10 - x_{1,k-1}} - \omega_{1,k-1} x_{2,k-1} x_{1,k-1}) \tag{5.31}$$

and

$$f_2 = sin(\omega_{2,k-1} t). \tag{5.32}$$

Here, $t = 0.02(k-1)$ is the current time. Moreover, the initial state satisfies Gaussian distribution with mean as

$$\begin{bmatrix} \bar{x}_{1,0} \\ \bar{x}_{2,0} \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix} \tag{5.33}$$

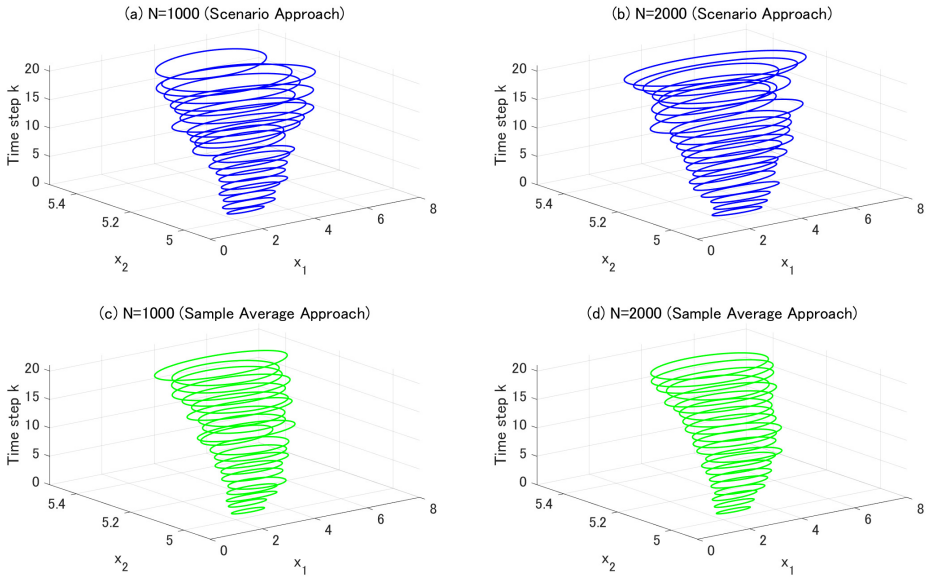

FIGURE 5.2: Time evolution of bounds computed by scenario approach and sample average approach using different sample numbers: (a) $N = 1000$ by scenario approach; b) $N = 2000$ by scenario approach; c) time step $N = 1000$ by sample average approach; d) time step $N = 2000$ by sample average approach.

and covariance matrix as

$$\Sigma_{x,0} = \begin{bmatrix} 0.16901 & 0.0001 \\ 0.0001 & 0.0001 \end{bmatrix}. \tag{5.34}$$

For the uncertain model parameters, the probability distribution is also Gaussian distribution and the mean vector and covariance matrix are written as

$$\begin{bmatrix} \bar{\omega}_1 \\ \bar{\omega}_2 \end{bmatrix} = \begin{bmatrix} 0.5 \\ \pi \end{bmatrix} \tag{5.35}$$

and

$$\Sigma_{\omega,0} = \begin{bmatrix} 0.36 & 0 \\ 0 & 0.36\pi^2 \end{bmatrix}. \tag{5.36}$$

5.5.2 Results and discussions

We compared the sample average approach and scenario approach in the numerical example. For the sample average approach, we set $\gamma = 0.01$ and $\epsilon = 0.05$ aiming to achieve an approximation of $\alpha = 0.1$. The numerical solution results on time evolution of the bound by both scenario approach-based method and sample average approach-based method with different N for step $k = 1$ to $k = 21$ are shown in Figure 5.2. The

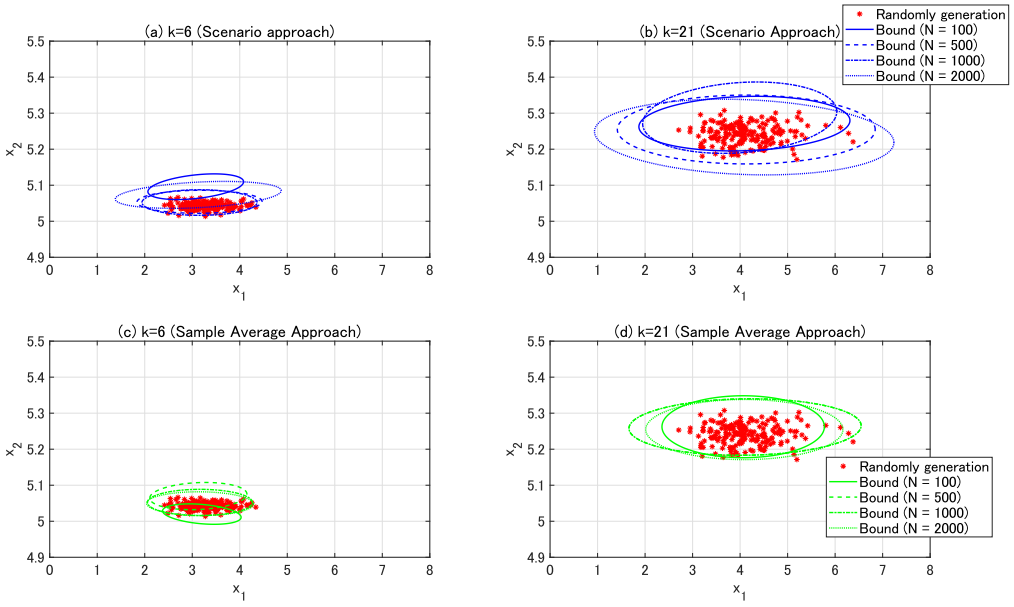

FIGURE 5.3: Bounds computed by scenario approach and sample average approach at different time steps: (a) time step $k = 6$ by scenario approach; b) time step $k = 21$ by scenario approach; c) time step $k = 6$ by sample average approach; d) time step $k = 21$ by sample average approach. The bounds are different types of blue lines and green line. The red "*" are the randomly generated points at the certain time steps.

bounds get wider as N is larger which implies that large N allows a smaller violation of the tests.

Moreover, the posterior Monte-Carlo analysis was implemented for the validation. This posterior test generated 201 trajectories of a system described by Equation (5.30) in one case. First, 501 different cases were simulated. The initial condition and uncertain parameter samples are chosen according to the distribution depends on Equation (5.33)-(5.36). In Figure 5.3, the generated points in the test of one case are put together with the computed bounds by the proposed algorithm. Obviously, for the both methods, with larger N, the bounds are larger and have a larger probability to have the test points inside the bounds. Moreover, the error gets larger when the prediction interval is further. Note that some points still locate out of the bounds even though 2000 samples are used in the sample average method since the bounds produced by sample average approach-based method is expected to ensure that the points locate inside the bound with a probability of $\alpha = 0.1$. However, for scenario approach, the bounds are larger than sample average approach and included all the points into the bound. Namely, scenario approach is going to achieve a completely robust bound rather than a probabilistic bound.

The validation results of the violation probability and the optimality are summarized in Fig. 5.4 which shows comprehensive statistical analysis results of 6000 cases of the posterior Monte-Carlor analysis. The probability that $\alpha > 0.1$ denoted as $\Pr\{\alpha > 0.1\}$

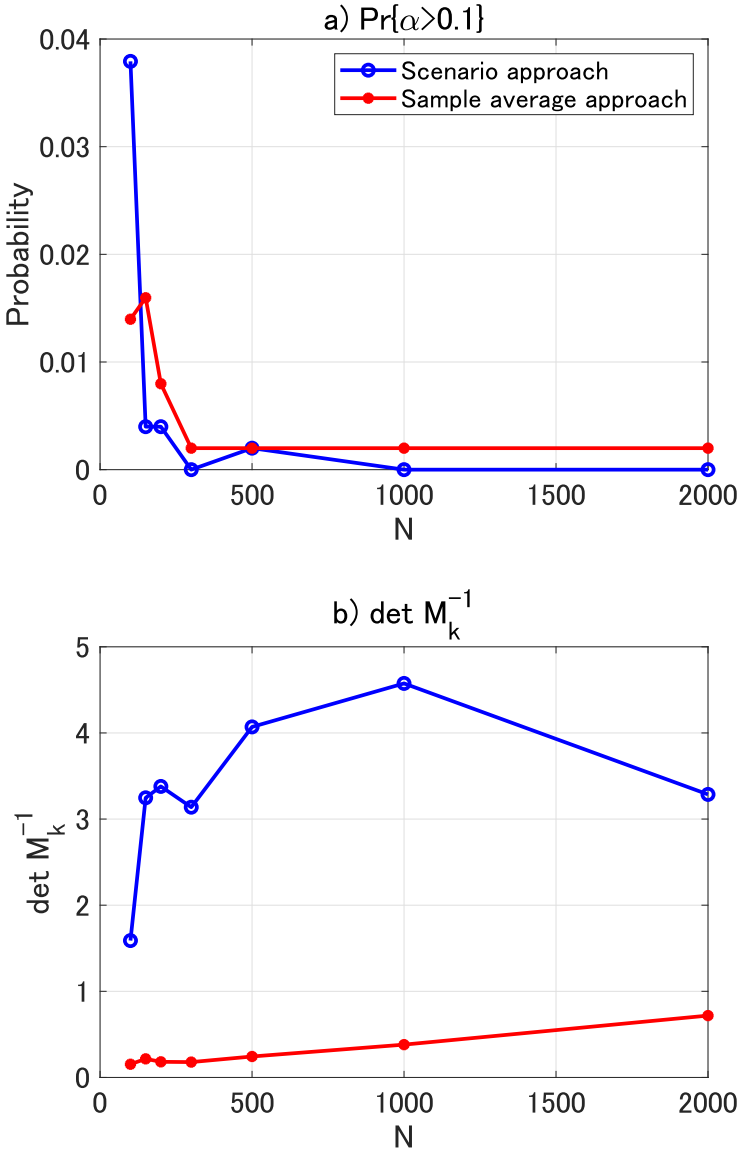

FIGURE 5.4: Comparison of scenario approach and sample average approach about the failure probability and cost function performance: a) Probability of constraints failure for cases with different sample numbers; b) Cost function performance for cases with different sample numbers.

is also calculated for every N. As N gets larger, the probability of constraints failure decreases statistically for both scenario approach and sample average approach. For scenario approach, when $N \geq 150$, $\Pr\{\alpha > 0.1\}$ is already smaller than 0.05 which means that it is potential to use smaller scenarios to ensure the required probabilistic level $\alpha = 0.1$. For sample average approach, the failure probability converges to a level slightly larger than 0.

To check the optimality, the mean value of $\det M_k^{-1}$ is also calculated for each method. The results show that the optimality goes worse with more samples for both methods. However, the sample average approach has a much better performance than the scenario approach.

5.6 Conclusion

The chance constrained optimization-based algorithms are proposed for computing the probabilistic ellipsoidal bounds of the state trajectories for nonlinear systems with uncertain initial state and parameters. The feasibility and optimality of the two proposed methods (scenario approach and sample average approach) are proved theoretically. A numerical example has also been implemented to validate the two method. Key contributions and findings are summarized as followings:

- A probabilistic constrained problem is formulated for calculating the probabilistic ellipsoidal bounds of the future trajectory of system states;
- Scenario approach and sample average approach are used to approximate the probabilistic constrained problem by formulating a deterministic problem with samples of the uncertain parameters in the system. For a given probabilistic level and upper bound of the violation probability, the least number of samples required for calculating the bound can be determined for both scenario approach and sample average approach;
- The optimality of the solutions obtained by scenario approach and sample average approach is discussed theoretically;
- The results of numerical example show that more samples will improve the feasibility by sacrificing optimality slightly for sample average approach. However the scenario approach needs more sacrifice on optimality.

The proposed method still has limitations. One of the most important limitations is that an extremely large sample size is required when the allowable probability level is a

very small number. Then, the computation burden will be not acceptable for real-time applications. Future works will be focused on

- In order to satisfy the extremely small allowable probability level in real-world problems of safety assessment, sample discarding approach will be investigated to accomplish the small allowable probability level with fewer samples;
- Further application case studies using real industrial data will be addressed.

Chapter 6

Discussion and Future Work

6.1 Model Improvement for State-Space Models

This thesis has proposed residual analysis-based model improvement for state-space models with nonlinear response. The future work will be focused on extending the method to nonlinear state-space models.

As a simple example, we consider the following stochastic nonlinear dynamical system

$$x(t+1) = 0.9x(t) + u(t) - \frac{5\sin(x(t))}{x(t)} + Q_{\text{Num}}\mathcal{N}(0,1), \tag{6.1}$$

$$y(t) = x(t) + \frac{5\sin(x(t))}{x(t)} + R_{\text{Num}}\mathcal{N}(0,1), \tag{6.2}$$

$$u(t) = 3\cos(0.2t). \tag{6.3}$$

In this example, Q_{Num} is fixed as 0.1. R_{Num} takes value of 0.1.

There are nonlinear parts in state equation and observation equation. We use linear model to fit the data generated by the above nonlinear system and obtain the predictive state residual and the predictive observation residual which are plotted in Fig. 6.1 and 6.2, respectively. In this case, the predictive state residual has the similar shape of the real nonlinear part in the state equation. However, the predictive observation residual goes far away from the nonlinear part in the observation equation. For a fixed value of x, the predictive observation residual has bifurcations which cannot be fitted by a

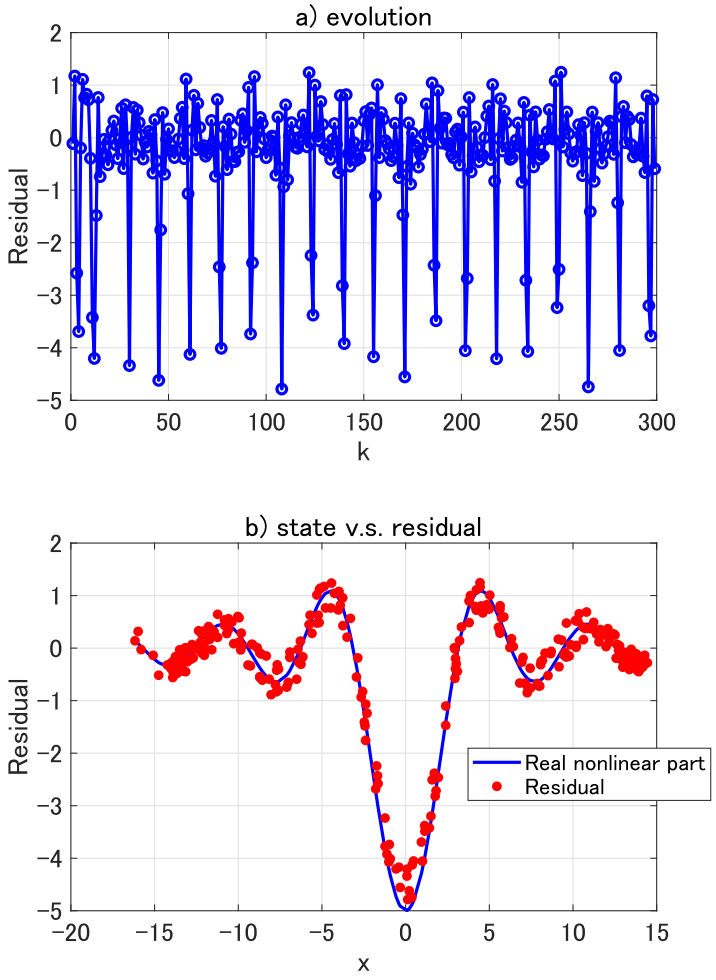

FIGURE 6.1: Predictive state residual of linear model for the nonlinear state-space model: a) evolution; b) state v.s. residual.

function. Thus, our decimation residual-based algorithm should be modified to address this issue.

Another aspect of the future work is to introduce deep learning model and generative model to fit the residual. In some applications, we will meet extremely complex nonlinear part in the state space model. The single-layer neural network cannot provide sufficient fitting accuracy for the complex nonlinear part. We have to use deep learning model to improve the model fit. Beside, if the map from state variable and input variable to the residual cannot describe by a function, we have to use generative model in which the map from state variable and input variable to the distribution of residual is established. Then, we could build a generative state-space model with better fitting performance.

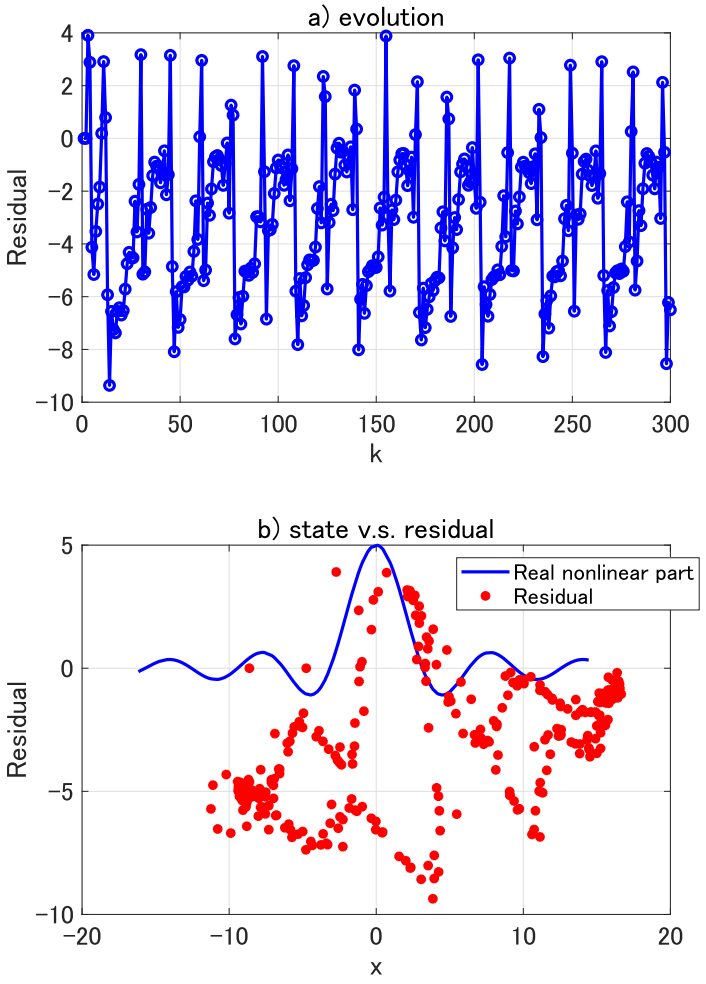

FIGURE 6.2: Predictive observation residual of linear model for the nonlinear state-space model: a) evolution; b) state v.s. residual.

6.2 Random Set Theory for Chance Constrained Optimization

In this paper, we proposed a sample-based neural approximation method to approximate the chance constrained optimization. The residual in this approximation is the distance between two sets, the optimal solution set of sample-based neural approximate problem and the optimal solution set of the original chance constrained optimization problem. We have proved that the residual vanishes with probability 1 if we choose infinite samples of random variables and use infinite activation functions. However, the following issues have not been touched:

- We need the assumption that both objective function and constrained function are

Lipschitz continuous. Besides, the distribution of constrained function for every input must be continuous. Can we extend our conclusion to more general case?

- We make sure that the convergence attains with probability 1 if the numbers of sample and activation functions become infinite. However, we still have to figure out how the residual will be for fixed numbers of random variables and activation functions.

To address the above issues, we can dig some things from random set theory. Notice that the optimal solution set of the approximate problem and the feasible domain of the approximate problem are essentially random set since they are determined by the randomly extracted samples from the sample space of the random variables. Thus, in the future, we would like to investigate whether we could find some theoretic results of random set theory to help us to solve the above issues.

Appendix A

Extreme Learning Machine

Extreme Learning Machine (ELM) is essentially an algorithm to train the parameters in Single Layer Feedforward Neural-networks (SLFNs). SLFNs is with a number of hidden nodes and with almost any nonlinear activation function, as an approximation of standard multilayer feedforward neural networks. For N arbitrary distinct samples (x_i, y_i), where $x_i = [x_{i,1}, ..., x_{i,m}]^T \in \mathbb{R}^n$ denotes the plant input and $y_i = [y_{i,1}, ..., y_{i,m}]^T \in \mathbb{R}^m$ the plant output, standard SLFNs with $\bar{N}$ hidden nodes and activation function $g(x)$ models the input-to-output relationship as

$$\hat{t}_i = \sum_{j=1}^{\bar{N}} \beta_j g_j(x_i) = \sum_{j=1}^{\bar{N}} \beta_j g(\omega_j^T x_i + b_j), \tag{A.1}$$

where $i = 1, ..., N$ is the sample index, $\omega_j = [\omega_{j,1}, ..., \omega_{j,n}]^T$ represent the weight vector connecting the j-th hidden node and the input nodes, $\beta_j = [\beta_{j,1}, ..., \beta_{j,m}]^T$ denotes the weight vector connecting the j-th hidden node and the output nodes, and b_j is the threshold of the j-th hidden node. $\omega_j^T x_i$ denotes the inner product of ω_j and x_i. The output nodes are linear in this SLFNs.

The standard SLFNs with $\bar{N}$ hidden nodes with activation function $g(x)$ is able to approximate these N samples with zero error means that

$$\sum_{i=1}^{N} \|e_i\| = \sum_{i=1}^{N} \left\| t_i - \hat{t}_i \right\| = 0, \tag{A.2}$$

i.e., according to the result in [*Huang*, 2003], there exists $\bar{N} \leq N, \beta_j, \omega_j$ and b_j such that

$$t_i = \sum_{j=1}^{\bar{N}} \beta_j g(\omega_j^T x_i + b_j), \tag{A.3}$$

where $j = 1, ..., N$ is sample index. More generally, by defining

$$H = \begin{bmatrix} g(\omega_1^T x_1 + b_1) & ... & g(\omega_{\bar{N}}^T x_1 + b_{\bar{N}}) \\ \vdots & ... & \vdots \\ g(\omega_{\bar{N}}^T x_N + b_1) & ... & g(\omega_{\bar{N}}^T x_N + b_{\bar{N}}) \end{bmatrix}, \tag{A.4}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_{\bar{N}}^T \end{bmatrix}, \tag{A.5}$$

and

$$T = \begin{bmatrix} t_1^T \\ \vdots \\ t_{\bar{N}}^T \end{bmatrix}, \tag{A.6}$$

the above N equations can be written compactly as

$$T = H\beta. \tag{A.7}$$

As introduced in [*Huang et al.*, 2006], H is called the hidden layer output matrix of the neural network; the i-th column of H is the i-th hidden node output with respect to inputs $x_1, x_2, ..., x_N$. The gradient-based algorithm can be used to train the value of $\beta, \omega_1, ..., \omega_{\bar{N}}, b_1, ..., b_{\bar{N}}$. Compare to the gradient-based algorithm, ELM algorithm, proposed in [*Huang et al.*, 2006], is more simple and efficient.

Regarding $X = [x_1, x_2, ..., x_N]$ as input, and T as output, the ELM-based SLFNs model training algorithm can be summarized as:

Batch ELM for training the SLFNs models

Input: $X, T, \bar{N}$

Output: $\beta, \omega_j, b_j, j = 1, 2, ..., \bar{N}$

1: Randomly assign ω_j and b_j, $j = 1, ..., \bar{N}$;

2: Calculate the hidden layer output matrix H;

3: Calculate the β as $\beta = H^M T$.

H^M is the Moore-Penrose generalized inverse of matrix H [*Rao and Mitra*, 1972], which can be derived by

$$H^M = (H^T H)^{-1} H^T. \tag{A.8}$$

Then, the estimation of β can be calculated as

$$\beta = (H^T H)^{-1} H^T T. \tag{A.9}$$

The basis of algorithm (Batch ELM for training the SLFNs models) is the results presented in [*Tamura and Tateishi*, 1997], if the activation function g is infinitely differentiable the hidden layer output matrix H is invertibel and $\|H\beta - T\| = 0$. The sequential implementation of Eq. A.9 can be derived and referred as the recursive least squares (RLS) algorithm. The proof of the RLS algorithm can be found in [*Chong and Zak*, 2001]. The sequential ELM algorithm is derived based on RLS algorithm and summarized as:

Algorithm 1 Sequential ELM for training the SLFNs models

1: Step 1: Give β_0 according to algorithm 1
2: Step 2: Calculate the hidden layer output matrix h_{k+1} based on further coming observation (x_{k+1}, t_{k+1}) according to Eq.A.4, $k = 0, 1, 2, ..., i, ...$
3: Step 3: Calculate the β_{k+1} as

$$\beta_{k+1} = \beta_k + M_{k+1} h_{k+1} (t_{k+1}^T - h_{k+1}^T \beta_k) \tag{A.10}$$

where M_{k+1} is calculated as

$$M_{k+1} = M_k - \frac{M_k h_{k+1} h_{k+1}^T M_k}{1 + h_{k+1}^T M_k h_{k+1}}. \tag{A.11}$$

4: Step 4: Set $k = k + 1$

Bibliography

Achiam, J., D. Held, A. Tamar, and P. Abbeel (2017), Constrained policy optimization., *Proceedings of the 34th International Conference on Machine Learning, PMLR*, *70*, 22–31.

Alamo, T., J. M. Bravo, M. J. Redondo, and E. F. Camacho (2008), A set-membership state estimation algorithm based on dc programming., *Automatic*, *44*, 216–224.

Baddeley, A., R. Turner, J. Moller, and M. Hazelton (2005), Residual analysis for spatial point processes, *Journal of the Royal Statistical Society, Series B*, *67*, 617–666.

Ben-Tal, A., and A. Nemirovski (1999), Robust solutions of uncertain linear programs., *Operations Research Letters*, *25*(1), 1–13.

Ben-Tal, A., and A. Nemirovski (2002), Robust optimization methodology and applications., *Mathematical Programming*, *92*, 453–480–13.

Ben-Tal, A., L. E. Ghaoui, and A. Nemirovski (2009), *Robust Optimization*, Princeton University Press, Princeton, NJ.

Blanke, M., M. Kinnaert, J. Lunze, and M. Staroswiecki (2006), *Diagnosis and Fault-Tolerant Control, 2nd ed.*, New York, NY, USA: Springer.

Boyd, S., and L. Vandenberghe (2004), *Convex Optimization*, Cambridge University Press.

Bray, A. K., and F. P. Schoenberg (2013), Assessment of point process models for earthquake forecasting, *Statistical Science*, *28*(4), 510–520.

Bray, A. K., K. Wong, C. D. Barr, and F. P. Schoenberg (2014), Voronoi residual analysis of spatial point process models with applications to california earthquake processe, *The Annals of Applied Statistics*, *8*(4), 2247–2267.

Brooks, S. P., and B. J. T. Morgan (1995), Optimization using simulated annealing., *Journal of the Royal Statistical Society. Series D (The Statistician)*, *44*(2), 241–257.

Buckby, J., T. Wang, J. Zhuang, and K. Obara (2020), Model checking for hidden markov models, *Journal of Computational and Graphical Statistics*, *29*(4), 859–874.

Calafiore, G., and M. C. Campi (2005), Uncertain convex programs: randomized solutions and confidence levels., *Math. Program.*, *102*(1), 25–46.

Calafiore, G., and M. C. Campi (2006), The scenario approach torobust control design., *IEEE Trans. Automatic Control*, *51*(5), 743–753.

Calvet, L., V. Czellar, and E. Ronchetti (2015), Robust filtering., *Journal of the American Statistical Association*, *110*(520), 1591–1606.

Campi, M. C. (2010), Classification with guaranteed probability of error., *Machine Learning*, *80*(1), 63–84.

Campi, M. C., and S. Garatti (2008), The exact feasibility of randomized solutions of uncertain convex programs., *SIAM Journal on Optimization*, *19*(3), 1222–1230.

Campi, M. C., and S. Garatti (2011), A sampling-and-discarding approach to chance-constrained optimization: feasibility and optimality., *Journal of Optimization Theory and Applications*, *148*(2), 257–280.

Campi, M. C., and S. Garatti (2019), *Introduction to the scenario approach*, MOS-SIAM Series on Optimization, Philadelphia.

Campi, M. C., G. Calafiore, and S. Garatti (2009), Interval predictor models: identification and reliability., *Automatica*, *45*(2), 382–392.

Campi, M. C., S. Garatti, and F. A. Ramponi (2018), A general scenario theory for non-convex optimization and decision making., *IEEE Transactions on Automatic Control*, *63*(12), 4067–4078.

Charnes, A., and W. W. Cooper (1959), Chance constrained programming., *Management Science*, *6*(1), 73–79.

Charnes, A., and W. W. Cooper (1970), On probabilistic constrained programming., *in Proceedings of the Princeton Symposium on Mathematical Programming, Princeton University Press, Princeton, NJ*, pp. 113–138.

Cheng, R., and J. Liu (2000), Mixture kalman filters., *Journal of the Royal Statistical Society Series B*, *62*(3), 493–508.

Chong, E., and S. Zak (2001), *An introduction to optimization*, New York: John Wiley.

Clements, R. A., F. P. Schoenberg, and A. Veen (2012), Evaluation of space-time point process models using super-thinning, *Environmetrics*, *23*, 606–616.

Cybenko, G. (1989), Approximation by superpositions of a sigmoidal function, *Mathematics of Control, Signals and Systems*, *2*, 303–314.

Deisenroth, M., R. Turner, M. Huber, U. Hanebeck, and C. Rasmussen (2012), Robust filtering and smoothing with gaussian process., *IEEE Transactions on Automatic Control*, *57*(7), 1865–1871.

Dekkers, A., and E. Aarts (1991), Global optimization and simulated annealing., *Mathematical Programming*, *50*, 367–393.

Durbin, J., and S. J. Koopman (2000), Time series analysis of non-gaussian observations based on stae space models from both classical and bayesian perspectives, *Journal of the Royal Statistical Society Series B*, *62*(1), 3–56.

Esfahani, P., and J. Lygeros (2016), A tractable fault detection and isolation approach for nonlinear systems with probabilistic performance., *IEEE Transactions on Automatic Control*, *61*(3), 633–647.

Farina, M., L. Giulioni, and R. Scattolini (2016), Stochastic linear model predictive control with chance constraints – a review., *Journal of Process Control*, *44*, 53–67.

Freund, R. J., R. W. Vail, and C. W. Clunies-Ross (1961), Residual analysis, *Journal of the American Statistical Association*, *56*(293), 98–104.

Gautam, P., R. Karki, and P. Piya (2020), Probabilistic modeling of energy storage to quantify market constrained reliability value to active distribution systems., *IEEE Transactions on Sustainable Energy*, *11*(2), 1043–1053.

Geletu, A., A. Hoffmann, M. Kloppel, and P. Li (2017), An inner-outer approximation approach to chance constrained optimization., *SIAM Journal on Optimization*, *27*(3), 1834–1857.

Ghahramani, Z., and S. Roweis (1999), Learning nonlinear dynamical systems using an em algorithm, *Advances in Neural Information Processing Systems*, *11*.

Goldberger, and S. Arthur (1961), Stepwise least squares: residual analysis and specification of error, *Journal of the American Statistical Association*, *56*(296), 998–1000.

Goodwin, J., O. Brown, and V. Helus (2020), Fast training of deep neural networks robust to adversarial perturbations., *2020 IEEE High Performance Extreme Computing Conference (HPEC)*, p. DOI: 10.1109/HPEC43674.2020.9286256.

Gramacy, R. B., and H. K. H. Lee (2010), Optimization under unknown constraints., *arXiv:1004.4027 [stat.ME]*.

Grammatico, S., X. Zhang, K. Margellos, P. J. Goulart, and J. Lygeros (2016), A scenario approach for non-convex control design., *IEEE Transactions on Automatic Control*, *61*(2), 334–345.

Guarniero, P., A. Johansen, and A. Lee (2017), The iterated auxiliary particle filter., *Journal of the American Statistical Association*, *112*(520), 1636–1647.

Guo, M., and M. M. Zavlanos (2018), Probabilistic motion planning under temporal tasks and soft constraints., *IEEE Transactions on Automatic Control*, *63*(12), 4051–4066.

Hassoun, M. (1995), *Fundamentals of Artificial Neural Networks*, The MIT Press.

Hoeffding, W. (1963), Probability inequalities for sums of bounded random variables., *Journal of the American Statistical Association*, *58*(301), 13–30.

Huang, G. (2003), Learning capability and storage capacity of two-hidden-layer feedforward networks., *IEEE Trans. Neural Networks*, *14*(2), 274–281.

Huang, G., Q. Zhu, and C. Siew (2006), Extreme learning machine theory and applications., *Neurocomputing*, *70*, 489–501.

Ingber, L. (1993), Simulated annealing: practice versus theory., *Math. Comput. Modeling*, *18*(11), 29–57.

Kalman, R. E. (1960), A new approach to linear filtering and prediction problems, *Transactions of the ASME, Journal of Basic Engineering, Series D*, *82*, 35–45.

Katzfuss, M., J. Stroud, and C. K. Wikle (2020), Ensemble kalman methods for high-dimensional hierachical dynamic space-time models, *Journal of the American Statistical Association*, *115*(530), 866–885.

Khalfallah, M. G., and A. M. Koliub (2007), Suggestions for improving wind turbines power curves., *Desalination*, *209*(1), 221–229.

Kirkpatrick, S., J. C. D. Gelett, and M. Vecchi (1983), Optimization by simulated annealing., *Science*, *220*(4598), 671–680.

Kishida, M., and R. B. Braatz (2015), Ellipsoidal bounds on state trajectories for discrete-time systems with linear fractional uncertainties., *Optim. Eng.*, *16*, 695–711.

Kishida, M., P. rumschinski, R. Findeisen, and R. B. Braatz (2014), Efficient polynomial-time outer bounds on state trajectories for uncertain polynomial systems using skewed structured singular values., *IEEE Transactions on Automatic Control*, *59*(11), 3063–3068.

Kitagawa, G. (1987), Non-gaussian state-space modeling of nonstationary time series, *Journal of the American Statistical Association*, *82*, 1032–1063.

Kitagawa, G. (1996), Monte carlo filter and smoother for non-gaussian nonlinear state space models, *Journal of Computational and Graphical Statistics*, *5*, 1–25.

Koyama, S., L. Perez-Bolde, C. Shalizi, and R. Kass (2010), Approximate methods for state-space models., *Journal of the American Statistical Association*, *105*(489), 170–180.

Kreuzer, A., and C. Czado (2020), Efficient bayesian inference for nonlinear state space models with univariate autoregressive state equation, *Journal of Computational and Graphical Statistics*, *29*(3), 523–534.

Lange, K. (2013), *Optimization*, Springer-Verlag New York.

Liu, L., G. Han, Y. He, and J. Jiang (2020), Fault-tolerant event region detection on trajectory pattern extraction for industrial wireless sensor networks, *IEEE Transactions on Industrial Informatics*, *16*(3), 2072–2080.

Luedtke, J., and S. Ahmed (2008), A sample approximation approach for optimization with probabilistic constraints., *SIAM Journal on Optimization*, *19*(2), 674–699.

Luzi, M., M. Paschero, A. Rizzi, E. Maiorino, F. Massimo, and F. Mscioli (2019), A novel neural networks ensemble approach for modeling electrochemical cells, *IEEE Transactions on Neural Networks and Learning Systems*, *30*(2), 343–354.

Luzi, M., F. Massimo, F. Mscioli, M. Paschero, and A. Rizzi (2020), A white-box equivalent neural network circuit model for soc estimation of electrochemical cells, *IEEE Transactions on Neural Networks and Learning Systems*, *31*(2), 371–382.

Lydia, M., S. S. Kumar, and G. E. P. Kumar (2013), Advanced algorithms for wind turbine power curve modeling., *IEEE Trans. Sustain. Energy*, *4*(3), 827–835.

Madry, A., A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu (2018), Classification with guaranteed probability of error., *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*.

Mesbah, A. (2016), Stochastic model predictive control: An overview and perspectives for future research., *IEEE Transactions on Control Systems Magazine*, *36*(6), 30–44.

Moser, D., R. Schmied, H. Waschl, and L. del Re (2018), Flexible spacing adaptive cruise control using stochastic model predictive control., *IEEE Trans. Control Systems Technology*, *26*(1), 274–281.

Nagy, Z. K., and R. D. Braatz (2003), Warst-case and distributional robustness anlysis of finite-time control trajectories for nonlinear distributed parameter systems., *IEEE Transactions on Control Syst. T.*, *11*, 494–504.

Niemi, J., and M. West (2010), Adaptive mixture modeling metropolis methods for bayesian analysis of nonlinear state-space models., *Journal of Computational and Graphic Statistics*, *19*(2), 260–280.

Nocedal, J., and S. J. Wright (2006), *Numerical Optimization*, Springer, New York.

Ogata, Y. (1988), Statistical models for earthquake occurrences and residual analysis for point processes, *Journal of the American Statistical Association*, *83*, 9–27.

Ouyang, T., A. Kusiak, and Y. He (2017), Modeling wind-turbine power curve: a data partitioning and mining approach., *Renewable Energy*, *102*(Part A), 1–8.

Pagnoncelli, B. K., S. Ahamed, and A. Shapiro (2009), Sample average approximation method for chance constrained programming: theory and applications., *Journal of Optimization Theory and Applications*, *142*, 399–416.

Paschero, M., G. L. Storti, A. Rizzi, and F. M. F. M. amd G. Rizzoni (2016), A novel mechanical analogy-based battery model for soc estimation using a multicell ekf, *IEEE Transactions on Sustainable Energy*, *7*(4), 1695–1702.

Pattipati, B., B. Balasingam, G. Avvari, K. Pattipati, and Y. Bar-Shalom (2014), Open circuit voltage characterization of lithium-ion batteries, *Journal of Power Sources*, *264*, 317–333.

Pena-Ordieres, A., J. Luedtke, and A. Wachter (2020), Solving chance-constrained problems via a smooth sample-based nonlinear approximation., *SIAM Journal on Optimization*, *30*(3), 2221–2250.

Picheny, V., R. B. Gramacy, S. M. Wild, and S. L. Digabel (2016), Bayesian optimization under mixed constraints with a slack-variable augmented lagrangian., *arXiv:1605.09466 [stat.CO]*.

Plett, G. (2006), Sigma-point kalman filtering for battery management systems of lipb-based hev battery packs: part 1: introduction and state estimation., *Journal of Power Sources*, *161*, 1356–1384.

Prajna, S. (2006), Barrier certificates for nonlinear model validation., *Automatica*, *42*, 117–126.

Rao, C., and S. Mitra (1972), *Generalized inverse of matrices and its application*, Wiley, New York.

Rostampour, V., R. Ferrari, and T. Keviczky (2017), A set based probabilistic approach to threshold design for optimal fault detection., *in Proceedings of American Control Conference, Seattle WA*.

Schildbach, G., L. Fagiano, C. Frei, and M. Morari (2014), The scenario approach for stochastic model predictive control with bounds on closed-loop constraint violations., *Automatica*, *50*(12), 3009–3018.

Schoenberg, F. P. (2003), Multidimensional residual analysis of point process models for earthquake occurrences, *Journal of the American Statistical Association*, *98*, 789–795.

Selmic, R., and F. L. Lewis (2002), Neural network approximation of piecewise continuous functions: application to friction compensation, *IEEE Transactions on Neural Networks*, *13*(2), 745–751.

Shapiro, A., D. Dentcheva, and A. Ruszczynski (2014), *Lectures on Stochastic Programming: Modeling and Theory 2nd ed.*, SIAM.

Shen, X., and T. Shen (2018), Chance-constrained optimization for torque tracking control with improving fuel economy in spark-ignition engines., *SICE Journal of Control, Measurement, and System Integration*, *11*(4), 365–371.

Shen, X., and J. Zhuang (2021), Decimating nonlinear response in state-space models, *Manuscript*.

Shen, X., Y. Zhang, T. Shen, and C. Khajorntraidet (2017), Spark advance self-optimization with knock probability threshold for lean-burn operation mode of si engine., *Energy*, *122*(1), 1–10.

Shen, X., J. Zhuang, and X. Zhang (2019), Approximate uncertain program, *IEEE Access*, *7*, 182,357–182,365.

Shen, X., T. Ouyang, Y. Zhang, and X. Zhang (2020), Computing probabilistic bounds on state trajectories for uncertain systems, *IEEE Transactions on Emerging Topics in Computational Intelligence*, *Early Access*, DOI: 10.1109/TETCI.2020.3019,040.

Shen, X., T. Ouyang, N. Yang, and J. Zhuang (2021), Sample-based neural approximation approach for probabilistic constrained programs, *IEEE Transactions on Neural Networks and Learning Systems*, *Early Access*, DOI:10.1109/TNNLS.2021.3102,323.

Sorenson, H. W. (1982), Parameter and state estimation: introduction and interrelation, *IFAC Proceedings Volumes*, *15*(4), 85–99.

Steinbrecher, G., and W. T. Shaw (2008), Quantile mechanics., *European Journal of Applied Mathematics*, *19*(2), 87–112.

Szu, H., and R. Hartley (1987), Fast simulated annealing., *Physics Letters A.*, *122*, 157–162.

Tamura, S., and M. Tateishi (1997), Capabilities of a four-layered feedforward neural network: four layers versus three., *IEEE Trans. Neural Networks*, *8*(2), 251–255.

Tanizaki, H. (2009), *Nonlinear Filters: Estimation and Applications*, Springer-Verlag Berlin.

Tsai, C. L., Z. Cai, and X. Wu (1998), The examination of residual plots, *Statistica Sinica*, *8*, 445–465.

van der Vaart, A. W. (1998), *Asymptotic Statistics (Cambridge Series in Statistical and Probabilistic Mathematics)*, Cambridge University Press.

Wan, Y., T. Keviczky, M. Verhaegen, and F. Gustafsson (2016), Data-driven robust receding horizon fault estimation., *Automatic*, *71*, 210–221.

Wan, Y., V. Puig, C. Ocampo-Martinez, Y. Wang, E. Harinath, and R. D. Braatz (2020), Fault detection for uncertain lpv systems using probabilistic set-membership parity relation., *Journal of Process Control*, *87*, 27–36.

Wang, Y., T. Wang, and J. Zhuang (2018), Modeling continuous time series with many zeros and an application to earthquakes, *Environmetrics*, *29*(4).

Watanabe, N. (1985), Notes on the kalman filter with estimated parameters, *Journal of Time Series*, *6*(4), 269–278.

Watson, M. W., and R. F. Engle (1983), Alternative algorithms for the estimation of dynamic factor, mimic and varying coefficient regression models, *Journal of Econometrics*, *23*, 385–400.

Witsenhausen, M. H. (1968), Sets of posible states of linear systems given perturbed observations., *IEEE Transactions on Automatic Control*, *13*(5), 556–558.

Yang, T., P. Mehta, and S. Meyn (2013), Feedback particle filter., *IEEE Transactions on Automatic Control*, *58*(10), 2465–2480.

Yu, K., and H. Chen (2019), Markov boundary-based outlier mining., *IEEE Transactions on Neural Networks and Learning Systems*, *30*(4), 1259–1264.

Zheng, F., Y. Xing, J. Jiang, B. Sun, J. Kim, and M. Pecht (2016), Influence of different open circuit voltage tests on state of charge online estimation for lithium-ion batteries, *Applied Energy*, *183*, 513–525.

Zhu, H., and H. Wu (2007), Estimation of smooth time-varying parameters in state space models, *Journal of Computational and Graphical Statistics*, *16*(4), 813–832.

Zhuang, J. (2006), Second-order residual analysis of spatiotemporal point processes and applications in model evaluation, *Journal of the Royal Statistical Society, Series B*, *68*(4), 635–653.

Zhuang, J. (2015), Weighted likelihood estimators for point processes, *Spatial Statistics*, *14*, 166–178.

Zhuang, J., Y. Ogata, and D. Vere-Jones (2002), Stochastic declustering of space-time earthquake occurrence., *Journal of the American Statistical Association*, *97*(458), 369–380.

Zucchini, W., and I. MacDonald (2009), *Hidden Markov Models for Time Series: An Introduction Using R.*, CRC Press.