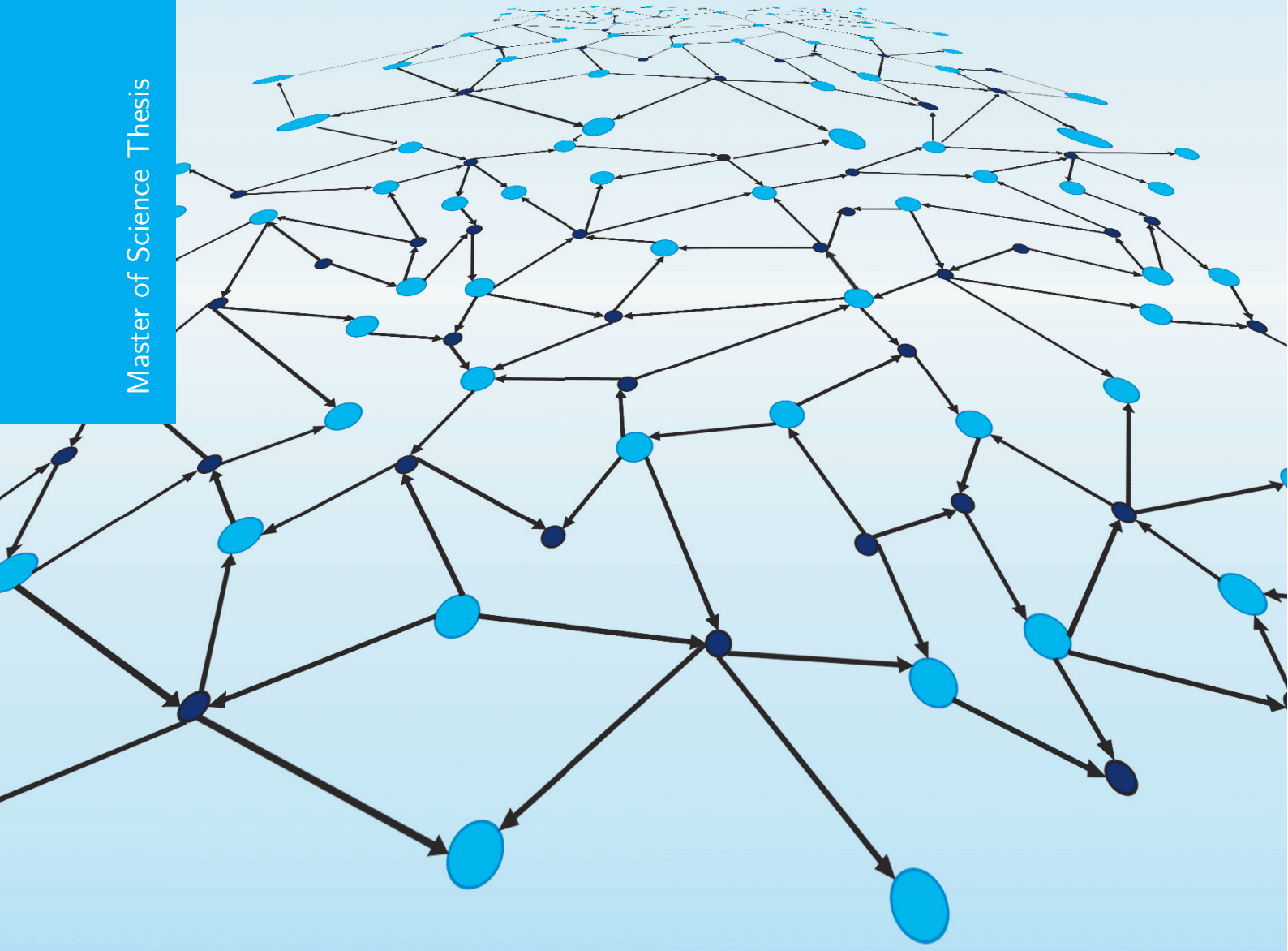
Uncertainty Propagation in Bilinear and Polynomial System for Probabilistic Threshold Detection

Tanay Milind Naik

Master of Science Thesis



TU Delft Delft University of Technology

Delft Center for Systems and Control

Uncertainty Propagation in Bilinear and Polynomial System for Probabilistic Threshold Detection

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Tanay Milind Naik

October 16, 2021

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

Uncertainty can be defined as imperfect or unknown information arising in a stochastic environment. Due to the very limited knowledge, it is difficult to propagate and quantify various uncertainties affecting the system to its next step. As a result, it has been a challenge to consider multiple uncertainties affecting the system in various fields, such as Fault Detection and Diagnosis. Fault detection has been an essential part of a large industrial and manufacturing system to take a proper corrective measure into account in a case of unexpected behavior. However, determining a robust threshold bound for fault detection is a big challenge. Several uncertainties affect the system (such as parametric uncertainties, experimental uncertainties, process noise, measurement noise, etc.). Ignoring the effects of various uncertainties (i.e., Deterministic Bound) can lead to a false alarm.

Therefore, to design a robust threshold probabilistic-based technique is used where all unknown parameters are taken into account. However, the major problem lies in propagating these unknown parameters into the next time step with their limited knowledge. The aim is to determine a probabilistic-based threshold by taking the effect of various uncertainty affecting into account. A novel algorithm (Message Passage Bilinear Uncertainty Propagation) is being proposed, which is used to quantify and propagate various uncertainties affecting the dynamical system into the next time step. Therefore, through this Master of Science Thesis, primary research is to design and validate the proposed algorithm and to check the algorithm can be used to determine a probabilistic threshold.

To answer this question, this report discusses firstly various other algorithms used to propagate uncertainty in Fault Detection then followed by the proposed novel algorithm. In the report, a detailed explanation of the algorithm for a trivial example is presented. The algorithm is then developed and implemented in MATLAB. Next, validation of the output generated through the algorithm is performed using Monte Carlo simulation. Finally, various analyses based on the MC simulation are discussed to support the results generated through the algorithm. An innovative approach is also discussed to extend the polynomial system algorithm as the proposed algorithm is limited to the bilinear system.

Further, a detailed explanation is given on applying the algorithm on a general state-space

model in terms of mathematics involved, which is further extended on applying the algorithm on a real-time application such as the Four Tank System. After successfully implementing the algorithm on the Four Tank System, the algorithm is used to propagate uncertainty into the dynamical system to determine the robust threshold. A robust threshold is found considering the effect of various uncertainty. The threshold found using the proposed algorithm is dynamic as it evolves based upon the state dynamics and satisfies the required condition. As a result presented algorithm satisfy all the requirement and can also be used in other applications. The algorithm was analyzed based upon various criteria to conclude this thesis, and a comparative study was conducted.

Table of Contents

List of Figures

List of Tables

Preface

Since my childhood, I have always heard about many tragic instances in industries or chemical factories due to negligence of proper fault diagnosis. These incidents always made me questioned how fault can be detected and how these accidents can be prevented. During my master's study in TU Delft in System and Control, I did an elective subject on Fault Diagnosis and Fault-Tolerant Control. During the course, I learned about how Fault Detection and Diagnosis are carried out in industries, and it answered many questions which, I had in the back of my mind. These all instances drove me towards taking my graduation in the Fault Detection and Diagnosis domain.

In September 2020, during my discussion with my professor Dr.R.Ferrari, he suggested an algorithm he proposed three years back but never tested. As I read more in-depth about the algorithm, it surprised me how uncertainty plays a vital role in a dynamic system and how drastically it can affect its performance. Another thing that fascinated me most was how this algorithm is not just limited to be applied for fault detection; instead, it can be used in various other domains where uncertainty plays a significant role. These factors motivated me to take this algorithm as my thesis topic.

During my masters, I learned in-depth about various controllers, modeling the dynamical system, graph theory, probability, and statistics. These all topics helped during various stages of my thesis. I consider myself lucky to get the opportunity to work under Professor Dr.R.Ferrari. Without him, the thesis wouldn't be possible.

I hope this thesis gives the reader a clear understanding of applying the proposed algorithm (i.e., Message Passing Bilinear Uncertainty Propagation) for uncertainty propagation in the dynamical system. Also hope that this algorithm is further investigated and applied in the various domain where uncertainty quantification and propagation is difficult.

Acknowledgements

After a lot of ups and downs, the two-year journey of my MSc System and Control at TU Delft comes to an end with this last milestone. Reaching this stage wouldn't have been possible without the support of many people involved, to whom I would like to express my sincere gratitude.

Firstly I would like to thank my supervisor Dr.R.Ferrari Assistant Professor in Fault-Tolerant Control, TU Delft, for his assistance and guidance during the whole process of my thesis. I am immensely grateful for his continuous support and the encouragement to push my boundaries. Furthermore, his immense knowledge and experience have always helped me understand my research area in various aspects.

I would like to thank my daily supervisor Ir.Z.Feng, who helped me at all various stages of my thesis. I want to thank her for all her valuable guidance throughout my thesis. It was due to her that I was able to tackle my day-to-day problems in my thesis. I would also like to thank Dr.D.Boskos for taking their time out for my thesis defense.

I will be indebted to my family for providing me with this wonderful opportunity. I am grateful for their constant love and encouragement, which kept me motivated and confident throughout my master's degree. My accomplishment and success are because they believed in me. I would also like to thank all my friends for their unfailing support and continuous encouragement. No amount of word would be enough to thank Pranav for being there with me during difficult times. I would also like to thank Mahalakshmi, Akshat, Pieter, and Prajwal for helping me survive my master's degree. A special thanks to Ishita for assisting me during my thesis report.

I perceive my Master's Degree as a big milestone in my career and development. I will continue to work on being a better engineer and helping out our global society.

Delft, University of Technology — Tanay Milind Naik
October 16, 2021

Vidyā Prasasyate Lokaih Vidyā Sarvatra Gauravā |

Vidyayā Labhate Sarvaṁ Vidvāna Sarvatra Pūjyate ‖

Knowledge is extolled by everyone, Knowledge is considered great everywhere. One can attain everything with the help of knowledge and person is respected everywhere

—*Chanakya Niti*

I dedicate this work to Mummy and Papa for all their sacrifice and their constant love.

.

Chapter 1

Introduction

Since the advent of the industrial revolution, there has been an increase in the demand for the use of sophisticated and complex processes to meet up with the higher system performance, product quality, and lower service cost [27]. To meet this increasing demand, the degree of automation in the technical manufacturing process is continuously growing. This calls for the use of a safer and more reliable based approach [16].

The use of the more and more automatic system has lead to an increase in the probability of occurrence of the fault. As a result, one of the critical issues in an automated system is reliability. The traditional way to reduce the fault in any system is to enhance the component's quality and robustness or use a hardware redundant-based approach. Even so, a fault-free system cannot be guaranteed as there will always be a presence of external influence such as noise or parametric uncertainty. Therefore, from 1970's fault detection and diagnosis have become an essential part of any control or automated system. Since then, enormous research has been going on in developing, theory, and implementing the Model-based fault detection technique. Today, model-based fault detection has become quite popular in various industrial and manufacturing sectors due to its easy implementation, higher reliability, accuracy, etc. [11].

Currently, there are a lot of model-based fault diagnosis techniques used. All these approaches differ in the type of process model, data set, applied algorithm, etc. Among the existing schemes, Probability-based techniques are pretty popular because of their efficiency and reliability for reconstructing the process variable. The main problem in this approach is taking into account the effect of various uncertainty affecting the system and designing a threshold for the proper fault detection and diagnosis. Not taking the effect of uncertainty concerning the system into consideration can lead to False Alarm Rate (FAR). Therefore there is a proper need to design an algorithm to propagate uncertainty affecting the system. Thus the main focus of the thesis is on designing and implementing an algorithm to propagate the uncertainty acting on the system for designing a correct threshold bound.

1-1 Motivation

Fault Detection and Diagnosis (FDD) are an essential part of the process control in any industrial system. These schemes reduce the probability of fault occurrence, which can ultimately lead to complete failure in the system. The unexpected failure can have a significant impact on the safety, economy, and environment. Throughout our history, various human tragedies could have been reduced if the proper corrective measures had been taken into account, such as:

- A Boeing 747-200F lost both of his engines while taking off from the Schiphol Airport, Amsterdam, ultimately crew lost its control and plane crashed into the building. There were a significant number of deaths. Maciejowski [30] suggested that this could have been avoided if there was an alarm to reconfigure the controller.
- A nuclear accident caused at Chernobyl Nuclear Power Plant. The main cause of the tragedy was the use of outdated instruments and the lack of fault handling instruments to detect the deviation from the normal operation.
- Bhopal gas tragedy caused due to leakage of water in Methyl Isocyanate tank causing an unwanted chemical reaction. This tragedy could have been prevented if there had been a proper diagnostic tool to detect and reduce the water leakage in the tank.

These tragedies in humankind could have been prevented if proper action had been taken into account correctly. Moreover, It is quite natural for a complex system to become vulnerable to these faults. Therefore, the FDD tool is necessary to maintain to detect the fault and minimize the impact of it acting on the system.

As discussed in the previous section, for the past 50 years, there has been much research in developing various algorithms to detect the faults and take corrective measures. However, the most widely preferred approach is the Model-Based Approach. In a Model-based approach, an observer is designed, which replicates the actual process of the system. Then, the comparison is made between the actual model, and an observer output and difference(i.e., residual) is generated. Finally, the residual is used to determine the threshold and fault in the system [32].

Model-based fault detection can be widely divided into two categories *Probabilistic Based Approach* and *Deterministic Based Approach*, based on how the threshold is defined. In a deterministic-based approach, the threshold is more or like constant values. These constant acts as an upper and lower limit. Values under these limits are used to signify a healthy system. However, the deterministic-based approach is generally not the preferred approach in industries because the threshold defined in this scheme is quite conservative and tightly bounded [36]. Besides, the possibility of uncertainty affecting the system is not taken into account. Uncertainty plays a significant role in any mathematical modeling of the system. Usually, the system's parameters have defects present, or a disturbance or noise is acting on the system due to inaccuracy in the model's design, sensor error, etc. However, these quantities are not known precisely. As a result, there is always a mismatch between the actual measurement and mathematical model even when the system is healthy [8]. This sometimes causes the system to take some rare values, which are not within the range and result in a false alarm.

To compensate for these drawbacks, a probabilistic based approach is used. In this approach, the threshold bound is more relaxed and flexible. It is defined in terms of a set, and the probability of a false alarm can be determined using a tunable parameter. It also minimizes the detection rate [36]. In this approach, parametric uncertainty is also taken into account. The main problem in this approach is how to propagate uncertainty into the system to calculate the threshold bound. Uncertainty quantities propagating into the system are not deterministic but rather stochastic quantities. Uncertainty being a stochastic (i.e., random) quantity, there is not much information other than its distribution (i.e., mean and covariance). Without much knowledge about the random variables, it is difficult to propagate it through the various time steps to calculate the threshold bound in the process. Therefore the main research area of my thesis lies in how to propagate uncertainties in the system to calculate the threshold bound in a better and efficient way.

1-2 Research Question

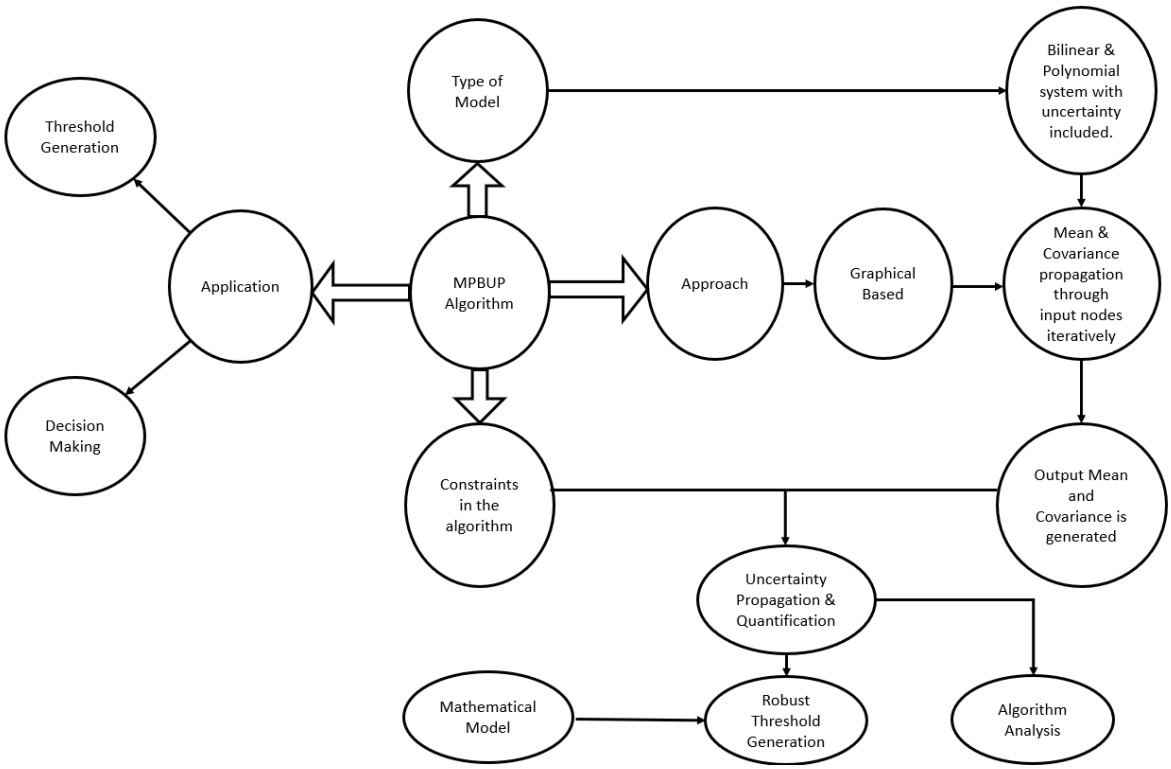


Figure 1.1: Mind Map for Formulating Objective of the thesis.

As discussed in the previous section, uncertainty plays a significant role in fault detection and can lead to FAR if the threshold bound is not correctly designed. However, propagating the stochastic uncertainty in a one-time step ahead in a dynamical system is quite challenging. As it is a random variable, and only mean and covariance is known. To evaluate the overall effect of the uncertainties, each parameter's mean and covariance value has to be propagated in a one-time step considering the other parameters' uncertainties. Therefore the main area of my research is to develop an algorithm to quantify the overall effect of uncertainty in any

dynamical model (i.e., linear and polynomial), perform its validation, and use the developed algorithm to find a robust threshold. As uncertainty propagation is complex, a graphically based algorithm is developed to overcome this issue and find the overall effect. The objective of the research is to study and find the answers to the following questions. Figure 1.1 shows the memory map, which helped me in formulating my research questions.

Research Question To design and develop an algorithm to automatically propagate parametric uncertainty, process noise, and measurement noise through the system and to use an algorithm to derive the robust threshold for fault detection?

Further, we lay out our research sub-questions as guidelines in answering our final research question.

Research Sub Questions

1. How to validate our algorithm and the output generated through our algorithm?
2. Can the developed algorithm for the uncertainty propagation be applied to both the bilinear and polynomial system? If not, what are the modification needed to be carried out?
3. How can the developed algorithm be applied to the state-space model, and what are the assumptions to consider?
4. How can the developed algorithm can be used successfully to evaluate a robust threshold?
5. What are constraint and bottleneck of the algorithm which affects its numerical efficiency and speed?

1-3 Research Contribution

As discussed before, from the past 50 years, much research has been carried out to develop various algorithms for fault detection. However, all the developed algorithm faces a significant difficulty in propagating the uncertainty present in the system model. As a result, parametric uncertainty and noise will always be present in the model, resulting in over conservative threshold bound.

To take the effect of the uncertainty into account, there are not many algorithms developed focusing on uncertainty propagation. Therefore, a graphical-based algorithm is proposed, which is used to quantify the effect of various uncertainty affecting the dynamical model. In the algorithm system's uncertainty is modeled as a bipartite graph. Uncertainty propagated at each time step through the algorithm. Besides developing the algorithm, my research also included applying the algorithm on the state-space model to derive the robust threshold and perform various algorithm analyses. The proposed algorithm does not find its application just as a part of probabilistic-based fault detection. However, it can be used in various other applications such as other fields also such as weather forecast, stock market prediction, etc.

1-4 Thesis Outline

The report is organized as follows: Chapter 2 provides a brief introduction to Fault, FDD, the principle of model-based fault detection, problem-related to probabilistic based approach, the importance of uncertainty, and the outline of the conducted literature survey of the various algorithm for uncertainty propagation in Fault detection. Chapter 3 introduces the proposed algorithm for the uncertainty propagation, a detailed explanation algorithm, and how to apply the algorithm for the higher degrees of system. Chapter 4 discusses the algorithm's validation for bilinear and polynomial systems and further analysis done using the validation experiment. Chapter 5 outlines how to apply the algorithm on a state-space model. It also discusses how to apply the algorithm on a real-time application such as a four tank system and, consequently, find a robust threshold bound for the system. In Chapter 6, various algorithm analysis is performed in terms of complexity, numerical efficiency, etc. and the MPBUP algorithm with Probabilistic based approach is also compared with other algorithms used for FDD. Finally, Chapter 7 concludes with the thesis research and future work that can be done.

Chapter 2

Background Information

2-1 Fault: Basic Concept

Fault in a dynamical system is defined as a deviation of the system structure or the parameter from its normal situation [2]. In other words, faults hamper or disturb the regular system operation, thus causing a deterioration in the system's performance, leading to failure. The fault is different from failure as we use fault to denote malfunction rather than a catastrophe. Furthermore, failure is used to suggest a complete breakdown of the system component or function. In other words, the fault may lead to failure if no proper action is taken into account [7].

2-2 Fault Detection and Diagnosis (FDD)

FDD units are designed in any industrial and manufacturing plant. They are used to take the corrective measurements properly and prevent any catastrophe action from taking place. The overall concept of FDD can be divided into three main essential tasks such as:-

- Fault Detection: Detect the fault in the functional unit when there is an unexpected change in the behavior. The output from the Fault Detection unit is usually binary, i.e., either 1 or 0. Where 1 indicates the presence of a fault and 0 indicates the system is in a healthy state.
- Fault Isolation: This unit is used to localize or classify the different based upon various parameters such as sources.
- Fault Analysis or Identification: Determine the type, magnitude, and cause of the fault.

Since 1990′s, there has been ongoing research to design various methods for FDD. All the different algorithms work on the basic principle to detect the change in the process parameters and hence can be called a change detection algorithm.

When a fault is diagnosed in the system, a proper corrective measure is taken into account to remove the fault in the system. This subsystem is called Fault Tolerant Control (FTC). Both FDD and FTC are used together in any industrial and manufacturing to take proper corrective measures against the fault and reduce its chance of occurrence. The Fault diagnosis and tolerance based algorithms or techniques are divided into the following categories:

- Hardware Redundancy - The main principle in this technique is to use redundant components. These redundant components run in parallel with the actual component and are used to detect the change in the output. Whenever a fault is present in the actual component, it is replaced by a redundant component through a relay switch. The main advantage is that this scheme is highly liable for fault isolation, but at the same time, it is expensive. Figure 2.1 shows the block diagram of the hardware redundant-based approach. Hardware redundant component is a part of FTC.

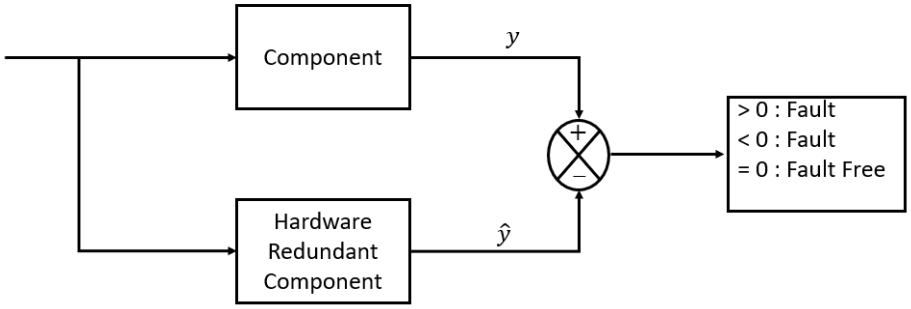


Figure 2.1: Schematic Diagram of Hardware Based Redundant Model.

- Software Redundancy - This approach is quite similar to hardware redundancy. The process model will run parallel to the process in software redundancy rather than redundant components, and the same process input will drive it. In a fault-free situation, the reconstructed process model will follow the actual process output. In contrast, there will be a deviation between the estimated model and the real-time process in the case of fault. This deviation is called residual [11]. The block diagram for software redundancy is similar to hardware redundancy, except the hardware component block is replaced by the software component block. Software redundant component can be used as a FDD and FTC, as it is used to detect the fault and in the presence of a fault replacing with another healthy software component block.

- Signal Processing Based- It works on the principle that specific signals carry information in the form of a symptom. Thus, fault can be detected by processing these signals through various ways such as spectral power density, limit check, etc. This approach is used for steady-state signal types. Figure 2.2 shows the block diagram of the signal based model.

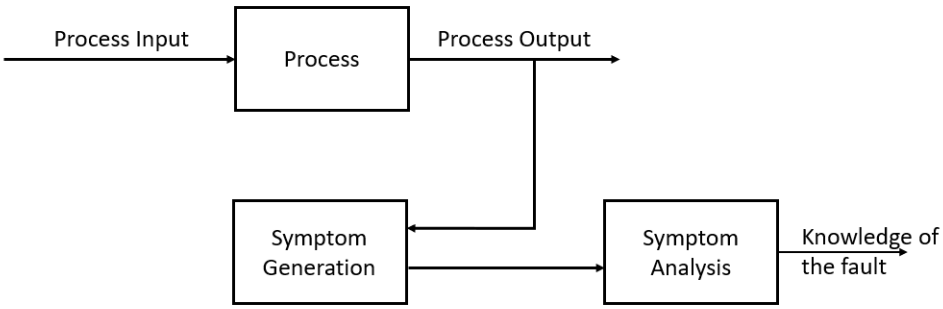


Figure 2.2: Schematic Description of Signal Based Model.

Software or analytical redundancy has been quite popular in the last two decades. The main advantage of software redundancy is that no additional hardware component is needed. It can be directly implemented on the digital computer and is comparatively cheaper. Other than this analytical-based approach uses the deep knowledge of the system to generate the model and residual. Compared to other methods, there are fewer chances of FAR and Missed Detection Rate (MDR).

2-3 Model Based Fault Diagnosis

The analytical or software-based approach is gaining more and more popularity for FDD. Because it is cheaper and more efficient than other approaches, and it can be applied online. The main principle of the analytical-based approach is that a difference signal is generated based upon the comparison between actual output and predicted output which is called a residual or symptom signal. Based upon the residual signal, the presence of fault is detected. The residual should be zero when the system has a fault-free operation and nonzero when a fault is present. This property of residual is used to determine if a fault is present in the system. The two crucial software-based technique used for FDD is *Model Based Fault Diagnosis* and *Data Based Fault Diagnosis*. Model-based fault diagnosis makes use of a mathematical model of the system to determine the fault [37]. The mathematical model of the system is based upon the prior measurement, system dynamics, etc. Data-based fault diagnosis uses a training data set (i.e., healthy data), which is used to train the system and detect the threshold. Further, this training data set is compared with the actual data to detect the fault in the system.

Model-based fault diagnosis can be mainly divided into stages such as residual generation and residual evaluation.

- Residual Generation: It generates a residual signal by comparing the actual process output with the estimated output from the mathematical model. The residual signal carries the information of the faults.
- Residual Evaluation: It is used to process the residual signal further for the possible information of the fault. It is used to extract information about the fault of interest from the residual signal employing post-processing the output signal [11].

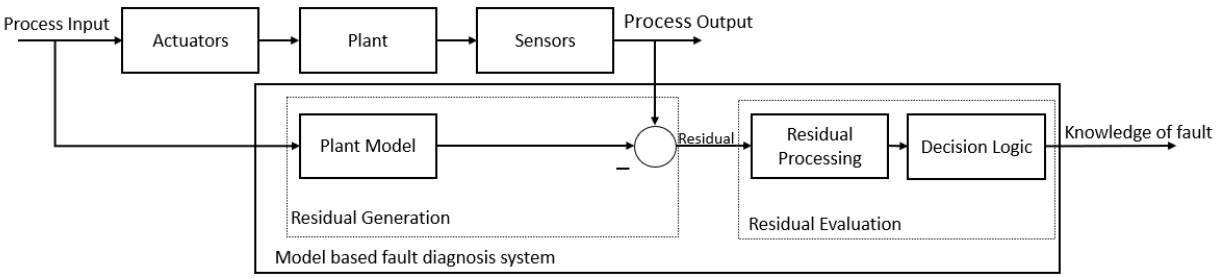


Figure 2.3: Schematic Description of Model Based Scheme.

The model-based fault diagnosis can be divided into three main categories as (1) Parity Space Approach, (2) Parameter Identification Approach, and (3) Output Observer-Based Approach.

Parity Based Approach is a general framework for Fault Detection and Identification (FDI), which includes both open loops and closed-loop strategies. In this approach, parity space signifies residual vector. A residual vector is a linear combination of sensor output and input applied over a finite time window. To detect the fault parity vector is analyzed. If the sensor output and input combination are trivial, the residual vector is zero, and the system is fault-free. On the other hand, when the system contains fault or a particular component is faulty, the subset of the residual vector will be nontrivial, indicating the presence of fault [31].

Parameter Based Approach is used when the process model is not known at all. Therefore, this method involves finding the model using the input u_k and the output y_k data through subspace identification. It is followed by finding the residual. The identification method also helps in gaining more information about the sources and how they get affected by faulty if there is a fault present in the system [16].

Observer-based techniques involve designing an observer which replicates the process and residual is obtained. In the past few decades, the output observer-based approach has been widely used for FDD. First, it has many advantages such as quick detection, no excitation input signal is required, and it can be easily implemented online and offline. Secondly, compared to other model-based techniques such as parity space or parameter estimation, it does not require any assumption or condition regarding the model.

2-4 Observer Based Approach

The observer-based technique is one of the widely applied model-based schemes for detecting a fault in the system. In the control domain, observers are used for estimating the states which are not measured, whereas, in the Fault Detection (FD), Output observers are used, i.e., these observer gives the estimates of the measurement [13].

In an observer, a process/mathematical model is developed, which is used to replicate the actual model. A residual signal is generated based upon the difference between the actual model and the estimated model. The residual signal contains a lot of information. If the residual signal is zero, it indicates that there is no model disturbance or uncertainty. Deviation

of the residual signal from zero indicates the presence of uncertainty in the model. There will always be a presence of parametric uncertainty and disturbance in the system model in an actual scenario. Therefore while designing an observer for FDD main aim is to reduce the effect of the model disturbance, and it does not affect the normal processing of the system. As a result, a threshold is designed which takes care of the model uncertainty and various disturbance present in the model. Threshold selection plays a crucial role in checking whether the change in the system parameter (i.e., residual) remains within the desired limits.

2-4-1 System Dynamics

The observer used for FD is a type of Luenbenger observer. Observer aims to generate an error (i.e., residual) signal and minimize the error between actual output and the estimated output through the state feedback mechanism. In the observer, an error is computed at each time step as the system propagates through time. So eventually the error in the system becomes zero ($e(t) \longrightarrow 0$).

The plant dynamics of any system in general can be defined as:

$$\begin{aligned} x(k+1) &= (A+\tilde{A})x(k) + (B+\tilde{B})u(k) + \chi(k) \\ y(k) &= (C+\tilde{C})x(k) + \xi(k) \end{aligned} \tag{2.1}$$

Where, $x(k)$ corresponds to the states in the system, $u(k)$ corresponds to input acting in the system, A, B and C are the state matrices, $\tilde{A}$, $\tilde{B}$ and $\tilde{C}$ are the uncertainty acting in the system in the state matrices. $\chi(k)$ corresponds to the process noise and $\xi(k)$ corresponds to the measurement noise.

The observer dynamics can be defined as:

$$\begin{aligned} \hat{x}(k+1) &= A\hat{x}(k) + Bu(k) + L(\hat{y}(k) - y(k)) \\ \hat{y}(k) &= C\hat{x}(k) \end{aligned} \tag{2.2}$$

Therefore, residual can be defined as $r_x(k) = x(k)$ - $\hat{x}(k)$

$$\begin{aligned} r_x(k+1) &= x(k+1) - \hat{x}(k+1) \\ r_x(k+1) &= [(A+\tilde{A})x(k) + (B+\tilde{B})u(k) + \chi(k)] - [A\hat{x}(k) + Bu(k) + L(\hat{y}(k) - y(k))] \\ r_x(k+1) &= A(x(k) - \hat{x}(k)) + \tilde{A}x(k) + \tilde{B}u(k) + \chi(k) + L[Cx(k) + \xi(k) - \hat{C}x(k)] \\ r_x(k+1) &= Ar_x(k) + \tilde{A}x(k) + \tilde{B}u(k) + \chi(k) + LCr_x(k) + L\xi(k) \\ r_x(k+1) &= A_0 r_x(k) + \gamma(k) \end{aligned} \tag{2.3}$$

Where, $A_0 = LC$ and $\gamma(k)$ corresponds to the total uncertainty acting in the system and r_x. It can be expressed as:

$$\gamma(k) = Ar_x(k) + \tilde{A}x(k) + \tilde{B}u(k) + \chi(k) + L\xi(k) \tag{2.4}$$

$$\begin{aligned} r_y(k+1) &= y(k) - \hat{y}(k) \\ r_y(k+1) &= [(C+\tilde{C})x(k) + \xi(k)] - [C\hat{x}(k)] \\ r_y(k+1) &= C(x(k) - \hat{x}(k)) + \xi(k) \\ r_y(k+1) &= Cr_x(k) + \xi(k) \end{aligned} \tag{2.5}$$

Residual in terms of state and output is found in the above equations 2.3 and 2.5 respectively. In the absence of fault, residual deviates from zero only due to modeling uncertainties, with the nominal value being around zero under the actual working condition. If the fault occurs, residual deviate from zero due to a change in the parameter from the fault-free condition. The role of the decision system is to determine whether the residual remains within the healthy condition bound [2]. The bound or limit is called a Threshold. There is various way to bound the system. The most common ways to bound the system are Deterministic and Probabilistic based schemes.

2-4-2 Deterministic based approach

In a Deterministic based scheme, residual is usually bounded by norm or limit-based approach. In norm-based approach, a scalar threshold τ is used to bound the residual signal r^0, whereas, in the limit based approach, a vector is found such that each j^{th} component of the threshold $\tau_{(j)}$ bounds the residual $|r^0_{(j)}|$. In this scheme, residual is bounded under the minimum and maximum value of r^0 in a fault-free condition. Mathematically for signal z, limits can be expressed as:

$$\begin{aligned} z < z_{\min} \text{ or } \quad z > z_{\max} &\implies \text{alarm, a fault is detected} \\ z_{\min} \leq z \leq z_{\max} &\implies \text{no alarm, fault-free} \end{aligned} \tag{2.6}$$

The deterministic-based approach is quite simple to design based on the knowledge of the model under the healthy condition and very simple to check the presence of a fault. However, the deterministic-based approach also has many setbacks, such as the upper bound on the uncertainty is relatively static and conservative. As a result, the inability of the traditional approach to tightly bound the arbitrarily shaped healthy region such as the nonconvex region. Also, this approach cannot take into account the possibly rare value taken by the uncertainty. The threshold value found using this approach is static; as a result, the evolution of the system dynamics is not taken into account. Due to the presence of these setbacks, a probabilistic-based approach is used [11].

Ideally, it is expected from the model-based approach to be robust concerning the unavoidable model and uncertainties measurement, thus having lower FAR and at the same time having reasonable detection rate (i.e., low MDR) [40]. However, in general, it is not possible to achieve both zero FAR and zero MDR. As a result, there is always a trade-off between FAR and MDR. To compensate for this trade-off Probabilistic based approach is used. In the Probabilistic based approach, dynamic bounds are used to bound the threshold.

2-4-3 Probabilistic based approach

In the probabilistic-based approach primary objective is to enhance the robustness of the residual generator against model uncertainties and disturbance without significant loss of the fault sensitivity. In this approach, statistical methods like Multivariate Control Charts (MCC) are widely used to detect a fault. However, the probability distribution of the observation is unknown. The algorithm aims to get a decision rule based upon the learning of data collected under normal operating conditions and a new observation that comes from normal or fault mode. In this scheme, the learning sample is commonly generated from a Gaussian distribution with a known mean and variance [41].

The bound in the probabilistic-based approach is dynamic. The residual of the system is calculated at each time step. Uncertainty, process noise, and measurement noise are assumed random variables with a known mean and covariance. These random variables are propagated through time along the nominal dynamics to calculate the threshold bound. Threshold bound in the probabilistic-based scheme is calculated using Mahalanobis Distance (MD) and Chebyshev Inequality. MD is a measure of Euclidean distance between the objects. In order words, it is a measure of the distance between a point P and distance D. It is a multi-dimensional generalization of measuring how many standard deviations away P is from the mean of D. The distance of P is zero at a mean of D, and it grows as P moves away from D along the principal component axis. MD calculates the Euclidean distance by taking the mean and covariance of the data into account [24]. Therefore MD is used to find how far is residual from the threshold.

Chebyshev Inequality is a probabilistic inequality. According to the inequality for any probability distribution with a mean (μ) and variance (σ^2) defined at least $(1 - \frac{1}{\lambda}^2)$ of the distribution are within the (λ) standard deviation of the mean. In other words, it states that the probability that an observation will be more than λ standard deviation from the mean is at most $\frac{1}{\lambda^2}$ [43]. Chebyshev inequality can be applied for a variety of distributions. Therefore using MD and Chebyshev inequality [9], the threshold can be defined as:

$$\mathcal{E}_\alpha \triangleq \left\{ r_y \in \mathbb{R}^n \mid \sqrt{(r_y - \bar{r}_y)\, \Sigma_{r_y}^{-1}\, (r_y - \bar{r}_y)} \leq \frac{n}{\alpha} \right\}, \alpha \in (0, 1] \tag{2.7}$$

Where,r_y is the output residual, $\bar{r_y}$ and Σ_{r_y} is the mean and covariance of the output residual. ε_α is the probabilistic threshold bound. α is a user defined constant, it used to guarantee against the desired probability of false alarm. In healthy condition following condition hold according to chebyshev inequality [9].

$$\text{Prob}\left[r_y \in \mathcal{E}_\alpha\right] \geq \alpha \tag{2.8}$$

The Probabilistic-based approach considers all the setbacks of the deterministic-based approach, and threshold bound minimizes the probability of the false alarm. Unfortunately, this scheme also has its setback, that the propagation of uncertainty in the system to compute the threshold bound. As uncertainty is a combination of stochastic quantities, there is no information available about the distribution. Therefore its propagation in the next time step is difficult. To overcome the drawback of the model-based approach, various uncertainty propagation algorithms are designed, such as Belief Propagation, MPBUP algorithm, etc. These algorithms are used with the probabilistic based approach to evaluate the threshold and detect the fault in the system [20].

2-5 Polynomial Chaos

Uncertainty is ubiquitous in a dynamic system or model. Another algorithm dealing with uncertainty propagation in the dynamical system for the FDD is Polynomial Chaos (PC). PC is a non-sampling-based method to determine the evolution of the parametric uncertainty in a system. It is used to quantify forward projection of the uncertainty [29], [22]. It transforms the stochastic-based equation to a deterministic-based equation and projects it into a higher dimension using orthogonal projection. PC converts the complex stochastic distribution of input variable onto measured variable to calculate the moment analytically and efficiently.

The PC-based approach has a strong mathematical basis and the ability to produce a functional representation of stochastic quantity. In PC, random input can be represented as a stochastic model based on which lower-order stochastic moment can be derived easily. PC replaces an implicit mapping between the variables/parameter and express them using an explicit parameter. It replaces the standard continuous differential equation describing the model with a series of orthogonal expansions. PC finds numerous applications in various domains such as sensitivity analysis, fluid dynamics, finite element analysis, and fault diagnosis, etc. PC is an active fault diagnostic tool. Active fault diagnosis involves designing an input signal that ensures detection or isolation of fault, but at the same time, the input signal is less intrusive to the system performance. It is used to separate the healthy model from the n faulty model based on designing a Probability Density Function (PDF) of the model [26].

2-5-1 PC as a Fault Detection Tool

PC-based active fault diagnosis can be divided into three steps. Firstly, the residual is constructed using the polynomial expansion of the variable. Then, stochastic moments of the system are computed, which is used to construct the PDF of the model having uncertain parameters. Secondly, the dissimilarity between the model output is found in the presence of probabilistic uncertainty. The similarity of the two models is directly related to the PDF of the model. The more similar are the model more is the Bayes error. The error is minimized to reduce the overlap between the output PDF of the two models. Thirdly, a nonlinear optimization is performed to determine an optimal input sequence that minimizes the overlap between input sequences of various models [39].

PC is used to discriminate the fault based on class type and severity of damage it can cause. The unknown input fault function g can be described as:

$$g = \bar{g}_i + \triangle g_i$$

Where, $\bar{g}_i$ corresponds to the constant mean value and $\triangle g_i$ corresponds to perturbation or stochastic variation around each mean value.

The threshold value for fault detection is taken constant around the mean value g_i. This constant value for fault detection in PC act like a control limit. The control limit is computed at each time step based on each operating mode's mean and variance. The mean and variance are calculated stochastically using PC-based expansion. Thus, the threshold limit is dynamic and will keep changing as the process evolves or propagates through time.

PC-based approach for fault detection can be formulated as a two-level detection problem. In the *Level* 1 *Algorithm*, for each mean value (g_i), PDF is calculated using PC expansion. Mean value remains constant in the neighborhood of the steady-state and acts as an operating mode in the system. Measurement values are calibrated around the measurement noise. The steady-state value calculated at a given time instant is compared against the predicted or calculated value. A steady-state is not effective against the transient changes. In *Level* 2 *Algorithm*, the algorithm is designed to consider transient changes. The algorithm is based on the fitting criterion of the measured variable over a moving time window. The fitting criteria are based upon either a Maximum Likelihood Function or Bayesian Inference Estimator. These fitting criteria are used to maximize the difference between the predicted and the measured value over the moving time window. It also compares it with the threshold to detect the presence of a fault in the system [5].

2-5-2 Advantages and Disadvantages

There are some advantages and disadvantages listed below for PC based approach for fault detection as follows :

Advantages

- The main advantage of using PC is that it reduces numerical complexity required in finding mean and variance of the system [26].
- Prehistoric data required by PC to model the stochastic distribution is much less compared to data needed by Monte Carlo Simulation.
- It takes into account the nonlinearity of the system by explicitly taking the first principle model [5].
- In PC based model computational time is comparatively is less. Hence, it can be applied on higher dimensional system [39].
- It can be used to calculate the multiple stochastic faults at the same time. It is also used to isolate multiple faults at the same time.

Disadvantages:

- PC is based upon a series of quadrature truncation that induces a numerical bias. These biases are very detrimental for convergence analysis in high dimensions.
- Calculation of higher moment is computationally complex.

2-6 Bayesian Networks

Bayesian Networks (BN) is a robust probabilistic tool that combines process knowledge with expert opinion. BN is directed acyclic graphs that embed the cause-effect relationship between the nodes (i.e., nodes represent various process variables). The framework of BN allows reasoning under uncertainty (i.e., conditional dependencies is represented via bipartite graph). A sequential evidence propagation computes the component failure probability through the conditional probability distribution specified for each node. BN-based approach for fault

detection method ranks the possible node based on the updated conditional probability distribution to show the contribution of each variable on the particular fault [6].

2-6-1 Bayesian Network as Fault Detection

BN-based fault detection mainly consists of four to five steps. Firstly, a BN is constructed using the bond graph, process knowledge data, etc. Then, the initial conditional probability of each node is taken into account using prior knowledge or by training the healthy data set. During the system operation, evidence is propagated through the network. External evidence (i.e., change in some parameters), it is asserted into the network through various nodes. Then, the effect of this evidence is propagated into the network. Finally, the impact of the evidence is adjusted into the network through the various nodes. When the value of the particular node is readjusted into the network, nodes that are connected to the particular node change their own belief to remain consistent with the network. Evidence propagation is an iterative process between the parent and child node until saturation is reached. Saturation is obtained when no other nodes propagate its evidence into effect.

The threshold in BN is a probabilistic limit that depends on the conditional probability of nodes calculated in healthy conditions. To identify the root cause of the fault in the system, a dynamic threshold is used. The threshold is calculated at each time. The kernel density function is used to estimate upper and lower control limits such that samples remain within the limit $[0, 1]$. These limits act as critical limits. The number of samples that exceed the critical limit is counted and normalized. The normalized value is used to determine the contribution of each variable by calculating the likelihood of the variable. It serves as an indicator for pointing out the primary cause of the fault [44].

In the BN-based approach, the system's parameters changes are propagated through the network using conditional probability. As a result, this approach can consider the various uncertainty affecting the system, such as parametric uncertainty, process noise, measurement noise, etc. Thresholds found using this approach will be robust to rare uncertainty values, and there will be fewer chances of FAR. At the same time, the algorithm also has some drawbacks. As BN constructs a directed acyclic graph and not all the industrial processes are acyclic. To model a cyclic process into BN is difficult. Therefore, a dummy or mediating variable is used. A dummy variable adds an extra node into the system and eventually increases the algorithm complexity.

2-6-2 Advantages and Disadvantages

Advantages:

- It is a graphical-based approach. It can be used to monitor the plant operation in a networked fashion so that variable interaction and casual relationships are well captured.
- It can identify the root cause of any process or fault propagation pathway through any abnormality caused in the system which cannot be identified in routine monitoring [44].
- BN has a good fault handling capacity. This approach is used to rank the multiple faults in terms of probability, hence giving an operator option to decide which fault should be given the priority [42].

Disadvantages:

- For a highly complex process as the number of variable increases, the learning algorithm become computationally difficult and enormous data is needed [6].
- BN cannot be directly applied on an online industrial process as it requires some time to update the conditional probability of its variable; hence it may causes a delay in fault detection [42].
- Message passage between two nodes can take place from multiple paths. As a result, a message can propagate indefinitely. This problem, in general, is complex and, for a large network, is computationally difficult [10].

2-7 Belief Propagation

Belief Propagation (BP) is a type of simple linear message propagating algorithm in a graphical-based structure. It is also called as *Loopy Belief Algorithm* (where the words loopy is used to signify graphical model with cycles). BP passes a message between the nodes and is iterated until convergence is reached or stopped after a finite number of steps. Each node in the graph has its own belief (i.e., posterior probability). The approach is quite different from the BN as it is a factor-based method. A factor graph is a tree. The factor graph are designed in such a way that one node is a root node, and any non-root nodes connected to other nodes are leaf nodes. In any factor graph, nodes are divided into two subsets, factor set, and variable set. There is a link connecting factor nodes and variable nodes [35].

BP is commonly used in artificial intelligence and information theory. Its use has been successfully demonstrated in various applications such as free energy approximation, fault detection, etc. The working principle of BP can be summarized that it is used to calculate the approximate factor marginal and variable marginal of the probability distribution of the factor graph. The calculation is done by message passing on a factor graph. Each node passes the message to its neighboring nodes. Variable node gives a message to its neighbor (i.e., the variable node sends a message to the factor node and vice versa). An outgoing message is a function of the incoming message at each node. This process is iterated until it reaches local convergence or fixed iteration. BP can be termed as one step extension of BN.

BP in fault detection algorithms is used for uncertainty propagation. The algorithm propagates various parametric uncertainty, measurement noise, process noise, etc., are propagated through the algorithm as an event between the neighboring nodes in a message-passing manner. As the algorithm propagates through the dynamical system, there is an exchange of information between the nodes for each time step. As a result, each node updates its own belief, and this continues until equilibrium is reached. The probabilistic threshold in BP can be computed using a healthy data set (i.e., fault-free data set), which corresponds to the mean and variance of the node-set. These moments are used to derive conditional probability. Conditional probability represents the mutual information shared among the variable. This information is used to derive the robust threshold. Threshold acts as a limit-based approach. When the system propagates through the next step, the variable's change is propagated as evidence in the network. As a result, the network updates its own belief. Updated belief is used to compute the moments and residual. Residual is compared with the threshold to detect the presence of fault [21]

BP can be seen as an interesting tool to propagate the uncertainty into the model and thus to derive a probabilistic threshold. At the same time, there are some setbacks in the algorithm, such as BP deals with undirected acyclic graphs. Moreover, it is difficult to take loops into account (as the main assumption on which message passage relies is that the states of the neighbor are uncorrelated with one another, which is only true if there are no loops present). As a result, *loopy belief propagation* is used. The main problem with loopy belief propagation is convergence, which adds a bias to the output model. There are more advantages and disadvantages related to the algorithm, which are discussed in the subsequent section.

2-7-1 Advantages and Disadvantages

Advantages

- It provides a causal relationship between the various factor and variables. Thus it is easy to determine the root cause of the fault.
- This factor-based approach helps in fault isolation because this algorithm works in a distributed or parallel-based approach.
- It reduces the computational bottleneck and improves the robustness of monitoring and diagnosis by avoiding a single point of failure.

Disadvantages

- For a dense and irregular graph, the fault signature matrix A is not sparse. It gets dominated by a lot of nonzero values. BP based approach relies on the graph sparsity to ensure efficiency and accuracy [21].
- At each step or iteration number of mixture terms required to represent the message will increase at an exponential rate. To handle this exponential rate, it has to be approximated by a small mixture [21].

2-8 Conclusion

In the previous sections, various model-based algorithms for FDD were discussed. Many of those algorithms did take uncertainty propagation in a dynamical model into account to evaluate a robust threshold for fault detection. However, at the same time, there are also many setbacks in these algorithms. These setbacks reduce their applicability to apply to various models. First, such as Polynomial Chaos, which propagates uncertainty in a dynamical model but due to its strong mathematical base (i.e., orthogonal projection, quadrature truncation), it is difficult to calculate the moments for highly complex processes. Similarly, both message-based approach Bayesian and Belief Propagation cannot be applied to a system with the loops. As a result, it limits their use to be used in any dynamical model. Secondly, due to their use of the particular type of graphical structure, the complexity increases exponentially.

As a result, there is a need for an algorithm for uncertainty quantification and propagation, which can be applied to any dynamical model considering loops. Moreover, its numerical complexity is comparatively less and can take various uncertain parameters into account simultaneously.

Chapter 3

Proposed Uncertainty Propagation Algorithm

The previous chapter discusses many algorithms for FDD in a system. In all, the algorithm discussed above bounding various types of uncertainty is complicated, which is needed for developing a robust threshold with a given performance in terms of FAR. Many of the algorithms discussed above involve various strategies to take the uncertainty of the system into account, but there is always an increase in the numerical complexity. Therefore considering all the factors, a novel algorithm *Message Passing Bilinear Uncertainty Propagation (MPBUP)* is designed, which can be used to propagate stochastic random variables through the time step and can be used to propagate uncertainty in the system [12]. MPBUP algorithm is used in correspondence with the Probabilistic Model-Based Fault Detection to propagate the uncertainty in the system at each time step to find a robust threshold for the fault detection. A detailed explanation of the algorithm is given in the subsequent section

3-1 MPBUP Algorithm

MPBUP algorithm is a message-passing algorithm designed to propagate the effect of uncertainty into the system. This algorithm was inspired by well known BP algorithm by J.Pearls.

The main idea of the algorithm is to automatically propagate the effects of uncertainty into an interconnected system at each time step and derive a threshold for fault detection. The principal working of the MPBUP algorithm is quite similar to the BP algorithm in terms of message propagation, but still, there are significant differences. In the algorithm, a bipartite graph is designed, containing two different nodes sets. First, a link connects the nodes in the node sets. Then, a message (i.e., mean and covariance) is propagated between the nodes from different node sets. Each node updates it's own mean and covariance at the end of each iteration, based upon the received message. This process continues until the message is propagated through all the nodes in the graph. Finally, the output nodes' estimates (i.e.,

mean and covariance) are used to compute the probabilistic threshold for the fault detection. A detailed explanation of the algorithm is explained in the subsequent section.

3-1-1 System Dynamics

In the MPBUP algorithm, the Equation 2.4 shows the total uncertainty affecting any system at a particular time instant. It includes parametric uncertainties due to various parameters involved in the model, measurement noise due to sensor calibration or inaccuracies, and process noise due to various external disturbances acting on the system. These uncertainties acting in the system can be quantified and expressed as the following class of bilinear equations:

$$a_l = \sum_{i=1}^{n} \theta_{l,(i)} a_i + \sum_{i=1}^{m} \kappa_{l,(i)} b_i + \sum_{i=1}^{n} \sum_{j=1}^{m} \psi_{l,(i,j)} a_i b_j + \sum_{i=1}^{m} \sum_{j=1}^{m} \omega_{l,(i,j)} b_i b_j, \forall l \in \{1, ..., n\} \tag{3.1}$$

Where $a \in \mathbb{R}^n$ are the quantity to be solved for (i.e., output), $b \in \mathbb{R}^m$ is the vector of random variable acting as a input to the system. n and m represents the number of input and output acting on the model. κ_l, $\theta_l \in \mathbb{R}^m$ and ψ_l,$\omega_l \in \mathbb{R}^{m \times m}$ are the vectors and matrices of the known coefficients in the bilinear equation. The goal of MPBUP algorithm is to iteratively compute the mean and covariance of a (i.e., output) using the knowledge of the mean ($\bar{b}$) and covariance (Σ_{b_j}) of the input nodes.

In the Equation, 3.1 input nodes (b) is constituted by all the parametric uncertainties, estimation error, process noise, and measurement noise numbered for convenience in lexicographic order, and output nodes (a_i) represents the total uncertainty propagated.

The key point in the MPBUP algorithm is to draw the bipartite graph. To draw the bipartite graph, nodes are needed to be divided into two sets. Therefore to generate two nodes sets Equation 3.1 is written in the form of the Sum of Product (SOP) expression. SOP expression helps in drawing the desired bipartite graph.

$$a_l = \sum_{h=1}^{n_f} \varphi_{(l,h)} f_h(a, b), \forall l \in \{1, ..., n\} \tag{3.2}$$

Where f_h is called as factors that which is obtained as the product of individual component of a and b. $n_f = (m+n)(m+1)$ are the possible factors, which for the convenience numbered in the lexicographic order. $\varphi \in \mathbb{R}^{n \times n_f}$ is a coefficient matrix in which coefficient elements (i.e., κ_l, θ_l, ψ_l,ω_l) are taken in right order.

The key part of the MPBUP algorithm is to propagate uncertainty in the parameter through the directed bipartite graph [$\mathcal{G} \triangleq (\mathcal{D}, \mathcal{E})$]. Where we introduced the node set [$\mathcal{D} \triangleq \mathcal{C} \cup \mathcal{F}$]. With the help of SOP, the node set can be partitioned into variable and factor set as $\mathcal{C} \triangleq \{a_1, ...a_n, b_1, ...b_n\}$ and $\mathcal{F} \triangleq \{f_1, ...f_n\}$ respectively. $\mathcal{E}$ signifies the edge set. Figure 3.1 shows a bipartite graph, where a distinction is made between factor and variable node-set in terms of their representation by using square and circle, respectively.

A vector set d is defined as concatenation of all the elements of $\mathcal{D}$ taken in lexicographic order. Finally two more sets are defined containing visited nodes $\mathcal{V}(q)$ and transversed edges $\mathcal{J}(q)$ at the end of each iteration q.

In the start, algorithm is initialized as:

$$\begin{aligned}\bar{d}(0) &\triangleq col(0_a, \bar{b}, 0_f)\\ \Sigma_d(0) &\triangleq diag(0_{\Sigma_a}, \Sigma_b, 0_{\Sigma_f})\\ \mathcal{V}(0) &= \{b_1, ...b_m\}\\ \mathcal{J}(0) &= \emptyset\end{aligned}$$

This signifies that the algorithm only has initial knowledge of the central moments of the input b at the start. Then algorithm alternate by sending message from factor $\longrightarrow$ variable and variable $\longrightarrow$ factor, until all node and edges result visited and transversed (i.e., $\mathcal{V} = \mathcal{D}$ and $\mathcal{T} = \mathcal{E}$). The message passage between the factor-variable and vice-versa are described in detail as:

Variables $\longrightarrow$ Factor: For every untransversed edge $e = (j, h) \in (\neg\mathcal{J})$ connecting a visited variable node (c_j) to the unvisited factor node (f_h). A message is transmitted to f_h containing $\bar{c}_j$ and covariance Cov $[c_j, d_i]$ between c_j and all the visited node $d_i \in \mathcal{V}$. Edge e is added to the transversed set $\mathcal{J}$. f_h is only added to the visited node set $\mathcal{V}$ if all its incident edges belong to $\mathcal{J}$.

Factor $\longrightarrow$ Variable: For every untransversed edge $e = (h, j) \in (\neg\mathcal{J})$ connecting a visited factor node (f_h) to an unvisited variable node (c_j). A message is transmitted to c_j containing $\bar{f}_h$ and covariance Cov$[f_h, d_i]$ between f_h and all the visited node d_i. Edge e is added to the transversed set $\mathcal{J}$. f_h is only added to the visited node set $\mathcal{V}$ if all its incident edges belong to $\mathcal{J}$. In factor $\longrightarrow$ variable, a variable node c_j is a linear combination of factors. So we can update mean and covariance as $\bar{d}(q) = \Phi\bar{d}(q-1)$and $\Sigma_d(q) = \Phi\Sigma_d(q-1)\Phi^T$. Φ matrix can be defined as $\Phi \in \mathbb{R}^{n_d \times n_d}$. Following condition holds such that $\Phi_{(i,i)} = 1$ for every i such that $d_i \in \mathcal{V}(q-1)$ and $\Phi_{(l,h)} = \varphi(l, h)$ for every h that belong to transversed set.

This process continues alternatively until all the nodes in the graph have been travelled .

The total uncertainty term γ can be a combination of various random variable such as $\tilde{A}$, r_x, χ, ξ, etc. To calculate the moments(i.e., mean and variance) of the product of the random variable Bohrnstedt and Goldberger formula is taken into account [3].

$$\begin{aligned}\mathbb{E}[\tilde{a}\tilde{b}] &= \mathbb{E}[\tilde{a}]\mathbb{E}[\tilde{b}] + \text{Cov}[\tilde{a}\tilde{b}]\\ \text{Var}[\tilde{a}\tilde{b}] &= \text{Cov}[\tilde{a}^2, \tilde{b}^2] + (\text{Var}[\tilde{a}] + \mathbb{E}[\tilde{a}]^2) + (\text{Var}[\tilde{b} + \mathbb{E}[\tilde{b}]^2] - (\text{Cov}[\tilde{a}, \tilde{b}] + \mathbb{E}[\tilde{a}]\mathbb{E}[\tilde{b}])^2 \quad (3.3)\\ \text{Cov}[\tilde{a}\tilde{b}, \tilde{c}] &= \mathbb{E}[\tilde{a}]\text{Cov}[\tilde{b}, \tilde{c}] + \mathbb{E}[\tilde{b}]\text{Cov}[\tilde{a}, \tilde{c}] + \mathbb{E}[\triangle\tilde{a} \triangle \tilde{b} \triangle c]\end{aligned}$$

The algorithm will iteratively compute mean $(\bar{d})$ and covariance (Σ_d), whoes values are iterated $q \in \{0, \frac{1}{2}, 1, ...\}$ by exchanging message over the directed edge.

3-2 MPBUP Algorithm Example

To get a clear understanding about the MPBUP algorithm, a simple bipartite network graph is taken in considered and the proposed algorithm can be illustrated step by step.

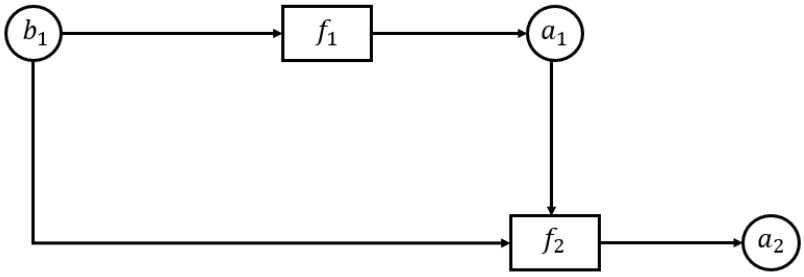


Figure 3.1: A Simple Bipartite Graph Network.

For the basic graph in figure 3.1, the algorithm can be initialized as:

Input for the network can be defined as:

$$b = b_1$$

Output for the network can be defined as:

$$a_l = \begin{bmatrix} a_1 & a_2 \end{bmatrix}$$

Therefore $n = 2$ and $m = 1$ based upon the number of input and output respectively, and $n_f = (m + n)(m + 1) = 6$.

On comparing with the standard class of Bilinear Equation in 3.1 the system dynamics for the network can be expressed as:

$$a_1 = \theta_{1,(1)}a_1 + \theta_{1,(2)}a_2 + \kappa_{1,(1)}b_1 + \psi_{1,(1,1)}a_1b_1 + \psi_{1,(2,1)}a_2b_1 + \omega_{1,(1,1)}b_1b_1$$
$$a_2 = \theta_{2,(1)}a_1 + \theta_{2,(2)}a_2 + \kappa_{2,(1)}b_1 + \psi_{2,(1,1)}a_1b_1 + \psi_{2,(2,1)}a_2b_1 + \omega_{2,(1,1)}b_1b_1$$

Therefore based upon the network in Figure 3.1, output nodes (a_1 and a_2) can be defined as:

$$a_1 = \kappa_{1,(1)}b_1$$
$$a_2 = \psi_{2,(1,1)}a_1b_1$$

Above equation can be expressed in SOP, where the coefficients (κ, ψ, ω and θ) are arranged in lexicographic order. It can be expressed as:

$$a_1 = \varphi_{(1,1)}f_1(a, b)$$
$$a_2 = \varphi_{(2,2)}f_2(a, b)$$

Coefficient Matrix/ Function Term	Variable
$\varphi_{(1,1)}$	$\kappa_{1,(1)}$
$\varphi_{(2,2)}$	$\psi_{2,(1,1)}$
f_1	a_1
f_2	$a_1 b_1$

Table 3-2.1: Relation between Coefficient Matrix/Function Term and Variables.

Table 3-2.1 shows the relationship between the coefficient matrix terms, function terms and various variable. Where $\varphi_{i,(i,j)}$ are the elements of the coefficient matrix and f_h represents the relation between the input and output.

Coefficient matrix will be of size $\varphi \in \mathbb{R}^{2\times 6}$. Finally, mean and covariance of vector d, visited node and transversed edge set at the start of the algorithm can be expressed as:

$$\begin{aligned}\bar{d}(0) &= \begin{bmatrix}0 & 0 & b_1 & 0 & 0\end{bmatrix}\\ \Sigma_d(0) &= \operatorname{diag}\begin{pmatrix}0 & 0 & \sigma^2[b_1] & 0 & 0\end{pmatrix}\\ \mathcal{V}(0) &= b_1\\ \mathcal{T}(0) &= \emptyset\end{aligned}$$

$\mathcal{V}(0)$ and $\mathcal{T}(0)$ indicate at the start of the algorithm that only mean and covariance of the input b is only known. While describing the detailed working of the algorithm in the subsequent section, green color is used to indicate that message is passed through this path, whereas red color is used to indicate no message is passed through this path.

Step 1

Variable $\longrightarrow$ Factor $(q = \frac{1}{2})$

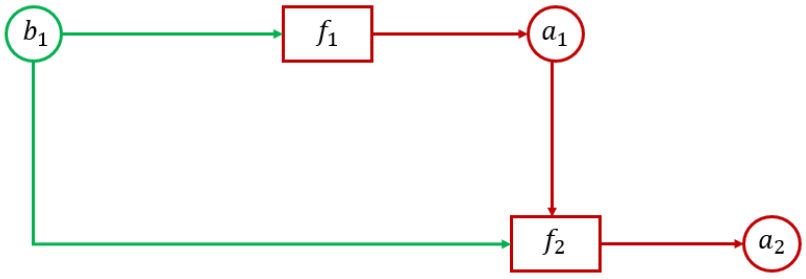


Figure 3.2: Bipartite Graph at Step = 1.

Variable	Status	Factor	Status
a_1	Waiting	f_1	Waiting
a_2	Waiting	f_2	Waiting
b_1	**Sent**		

Table 3-2.2: Node status of Bipartite Graph Network at Step = 1.

$$\mathcal{V}(\frac{1}{2}) = [b_1, f_1]$$
$$\mathcal{T}(\frac{1}{2}) = [(b_1, f_1), (b_1, f_2)]$$
$$\overline{d}(\frac{1}{2}) = [0, 0, \overline{b_1}, \overline{b_1}, 0]$$
$$\Sigma_d(\frac{1}{2}) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2[b_1] & \sigma^2[b_1] & 0 \\ 0 & 0 & \sigma^2[b_1] & \sigma^2[b_1] & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Step 2

Factor $\longrightarrow$ Variable $(q = 1)$

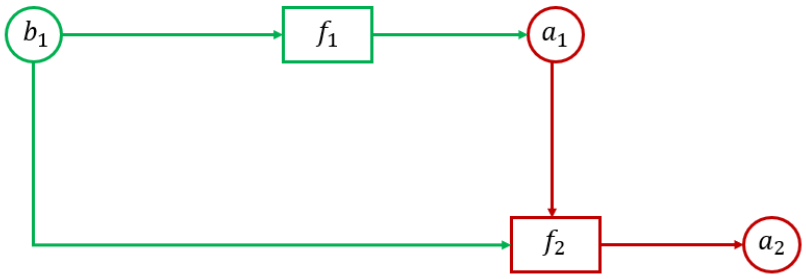


Figure 3.3: Bipartite Graph at Step $= 2$.

Variable	Status	Factor	Status
a_1	Waiting	f_1	**Sent**
a_2	Waiting	f_2	Waiting
b_1	Sent		

Table 3-2.3: Node status of Bipartite Graph Network at Step $= 2$.

$$\mathcal{V}(1) = [b_1, f_1, a_1]$$
$$\mathcal{T}(1) = [(b_1, f_1), (b_1, f_2), (f_1, a_1)]$$

In this step only f_1 can transmit the message as f_2 is waiting to receive incoming message from its incoming edge (a_1, f_2). As discussed in previous a_l is linear combination of function

f_h. Therefore, ϕ matrix is used. ϕ matrix can be defined as:

$$\Phi = \begin{pmatrix} 0 & 0 & 0 & \varphi_{(1,1)} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The mean and covariance of d vector can be updated according to $\bar{d}(q) = \Phi\bar{d}(q-1)$ and $\Sigma_d = \Phi\Sigma_d(q-1)\Phi^T$ respectively.

$$\bar{d}(1) = \Phi\bar{d}q(\frac{1}{2})$$

$$\bar{d}(1) = \begin{pmatrix} 0 & 0 & 0 & \varphi_{(1,1)} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ \bar{b}_1 \\ \bar{b}_1 \\ 0 \end{pmatrix} = \begin{pmatrix} \varphi_{(1,1)]}\bar{b}_1 \\ 0 \\ \bar{b}_1 \\ \bar{b}_1 \\ 0 \end{pmatrix}$$

$$\Sigma_d(1) = \Phi\bar{d}(\frac{1}{2})\Phi^T$$

$$= \begin{pmatrix} 0 & 0 & 0 & \varphi_{(1,1)} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2[b_1] & \sigma^2[b_1] & 0 \\ 0 & 0 & \sigma^2[b_1] & \sigma^2[b_1] & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ \varphi_{(1,1)} & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} \varphi^2_{(1,1)}\sigma^2[b_1] & 0 & \varphi_{(1,1)}\sigma^2[b_1] & \varphi_{(1,1)}\sigma^2[b_1] & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \varphi_{(1,1)}\sigma^2[b_1] & 0 & \sigma^2[b_1] & \sigma^2[b_1] & 0 \\ \varphi_{(1,1)}\sigma^2[b_1] & 0 & \sigma^2[b_1] & \sigma^2[b_1] & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$\Sigma_{d(1,1)}$, $\Sigma_{d(1,3)}$, $\Sigma_{d(1,4)}$, $\Sigma_{d(2,1)}$, $\Sigma_{d(3,1)}$, $\Sigma_{d(4,1)}$ represents the nonzero terms the covariance matrix (i.e., Σ_d)

Step 3

Variable $\longrightarrow$ Factor (q = $1\frac{1}{2}$)

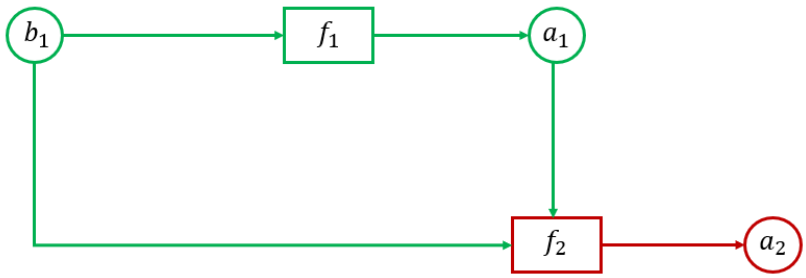


Figure 3.4: Bipartite Graph at Step = 3.

Variable	Status	Factor	Status
a_1	**Sent**	f_1	Sent
a_2	Waiting	f_2	Waiting
b_1	Sent		

Table 3-2.4: Node status of Bipartite Graph Network at Step $= 3$.

$$\mathcal{V}(1\frac{1}{2}) = [b_1, f_1, a_1, f_2]$$
$$\mathcal{T}(1\frac{1}{2}) = [(b_1, f_1), (b_1, f_2), (f_1, a_1), (f_2, a_1)]$$

$f_2 = b_1 a_1$. Therefore the mean (i.e., $\mathbb{E}[f_2]$) and covariance (i.e., $\sigma^2[f_2, a_1]$, $\sigma^2[f_2, a_2]$, $\sigma^2[f_2, f_1]$) of all the nodes involving f_2 are computed using the Bohrnstedt and Goldberger formula [3].

$$\bar{d} = \begin{bmatrix} \mathbb{E}[a_1] & 0 & \bar{b}_1 & \bar{b}_1 & \mathbb{E}[f_2] \end{bmatrix}$$
$$\Sigma_d(1\frac{1}{2}) = \begin{pmatrix} \varphi^2_{(1,1)}\sigma^2[b_1] & 0 & \varphi_{(1,1)}\sigma^2[b_1] & \varphi_{(1,1)}\sigma^2[b_1] & \sigma^2[f_2, a_1] \\ 0 & 0 & 0 & 0 & 0 \\ \varphi_{(1,1)}\sigma^2[b_1] & 0 & \sigma^2[b_1] & \sigma^2[b_1] & \sigma^2[f_2, b_1] \\ \varphi_{(1,1)}\sigma^2[b_1] & 0 & \sigma^2[b_1] & \sigma^2[b_1] & \sigma^2[f_2, f_1] \\ \sigma^2[f_2, a_1] & 0 & \sigma^2[f_2, b_1] & \sigma^2[b_1, f_1] & E[f_2] \end{pmatrix}$$

Step 4

Factor $\longrightarrow$ Variable $(q = 2)$

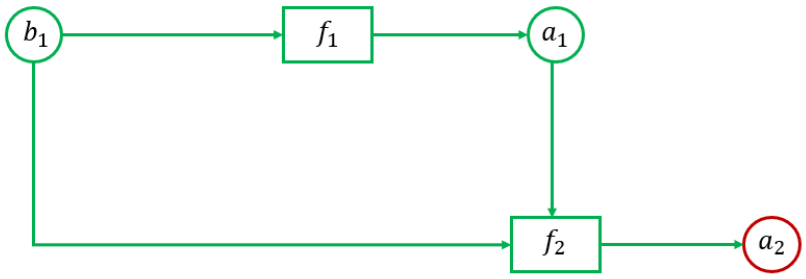


Figure 3.5: Bipartite Graph at Step $= 4$.

Variable	Status	Factor	Status
a_1	Sent	f_1	Sent
a_2	Waiting	f_2	**Sent**
b_1	Sent		

Table 3-2.5: Node status of Bipartite Graph Network at Step $= 4$.

$$\mathcal{V}(2) = [b_1, f_1, a_1, f_2, a_2]$$
$$\mathcal{T}(2) = [(b_1, f_1), (b_1, f_2), (f_1, a_1), (f_2, a_1), (f_2, a_2)]$$

In last step mean of node a_2 can be updated using $\varphi_{(2,2)}$ times the mean of f_2, while covariance of node a_2 can be updated as $\varphi^2_{(2,2)}$ times those of f_2.

$$\bar{d} = \begin{bmatrix} \mathbb{E}[a_1] & \mathbb{E}[f_2] & \bar{b}_1 & \bar{b}_1 & \mathbb{E}[f_2] \end{bmatrix}$$

$$\Sigma_d(2) = \begin{pmatrix} \varphi^2_{(1,1)}\sigma^2[b_1] & \varphi^2_{(2,2)}\sigma^2[f_2, a_1] & \varphi_{(1,1)}\sigma^2[b_1] & \varphi_{(1,1)}\sigma^2[b_1] & \sigma^2[f_2, a_1] \\ \varphi^2_{(2,2)}\sigma^2[f_2, a_1] & \varphi^2_{(2,2)}E[f_2] & \varphi^2_{(2,2)}\sigma^2[f_2, b_1] & \varphi^2_{(2,2)}\sigma^2[f_2, a_1] & \varphi^2_{(2,2)}E[f_2] \\ \varphi_{(1,1)}\sigma^2[b_1] & \varphi^2_{(2,2)}\sigma^2[f_2, b_1] & \sigma^2[b_1] & \sigma^2[b_1] & \sigma^2[f_2, b_1] \\ \varphi_{(1,1)}\sigma^2[b_1] & \varphi^2_{(2,2)}\sigma^2[f_2, a_1] & \sigma^2[b_1] & \sigma^2[b_1] & \sigma^2[f_2, f_1] \\ \sigma^2[f_2, a_1] & \varphi^2_{(2,2)}E[f_2] & \sigma^2[f_2, b_1] & \sigma^2[b_1, f_1] & E[f_2] \end{pmatrix}$$

Final Outcome

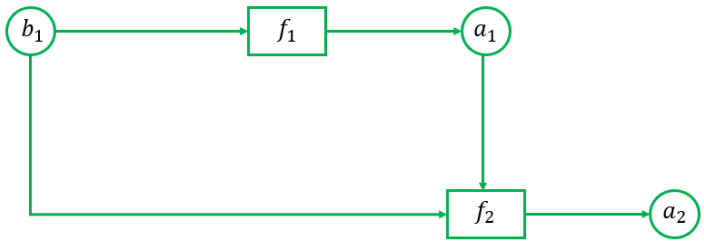


Figure 3.6: Bipartite Graph at Final Outcome.

Variable	Status	Factor	Status
a_1	Sent	f_1	Sent
a_2	Waiting	f_2	Sent
b_1	**Received**		

Table 3-2.6: Node status of Bipartite Graph Network at Final Stage.

In this section, a detailed explanation of the MPBUP algorithm was given through a bipartite graph example. It can see that initially, at the start, we only know about the mean and covariance of the input node. When the algorithm is propagated further into the next iteration, the moments of the input nodes are used to calculate the mean and covariance of the other node. Finally, moments of the output nodes are generated. This process continues until all the nodes are traveled, and we know the output nodes.

MPBUP algorithm is a novel algorithm to find the moments of the random variables for which we have limited knowledge.

3-3 MPBUP Implementation

MPBUP Algorithm was designed and successfully implemented on MATLAB (9.7, R2020b). The main aim while designing the algorithm was to keep it as automated as possible so that it could be applied on any graphical network. The working of the algorithm is divided into two parts.

An adjacent matrix (i.e., the matrix element indicates that the pairs of vertices are adjacent or not) is input into the system. Next, a corresponding bipartite directed graph is generated using the command digraph in MATLAB. In the graph, both variable and factor node-set are segregated. Then a function is created which operates the standard MPBUP Algorithm. The function generates the mean and covariance matrix for the output nodes when the corresponding graph, coefficient matrix, number of input and output nodes, and the moments of the input node are given as input into the function. Thus, the function replicates the process of the MPBUP algorithm, where the central moments of the input node are propagated to find the central moments of other nodes present in the graph. The function also takes Bohrnstedt and Goldberger's formula into account for calculating the moments of the bilinear nodes. The implementation of the algorithm and its validation are discussed in detail in the subsequent chapter.

3-4 MPBUP Algorithm for Polynomial System

MPBUP algorithm is a novel algorithm that has been designed to propagate uncertainty into the system when uncertainty acting on the system is a stochastic quantity. It uses the mean and covariance of the input node to propagate uncertainty into the system. The main aim of the algorithm is to propagate uncertainty through the system in terms of the moments (i.e., mean and covariance) to find moments of the various node connected in the network at each iteration. To calculate the mean and covariance for the combination of a random variable, Bohrnstedt and Goldberger's formula in equation 3.3 is taken into account.

The main problem with using these formulas is that they are restricted to bilinear terms (i.e., a product of two-term). Therefore, they cannot be applied directly to the polynomial terms (i.e., a product of two or more terms). This problem can cause a significant setback in the practical use of the algorithm for uncertainty propagation. Furthermore, as many industrial processes will not be bilinear, it will not be easy to model polynomial systems to linear or bilinear systems. Therefore to compensate for this bottleneck in the algorithm, research was conducted as a part of my thesis, and an optimal solution was found. To understand the problem and the optimum solution derived, a polynomial example is taken into consideration. Figure 3.7 is an example of a bipartite graph for a polynomial system.

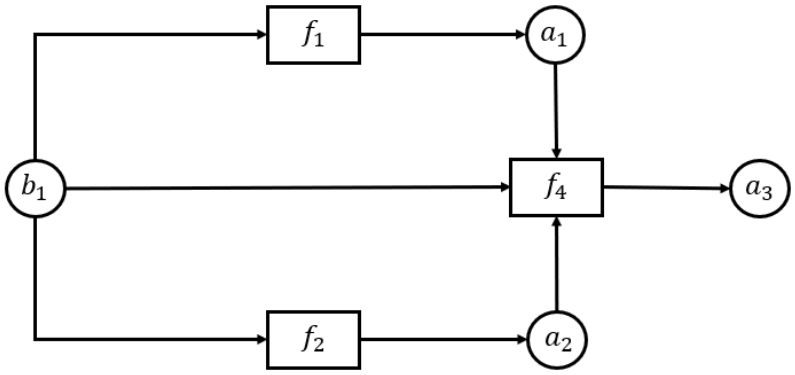


Figure 3.7: Bipartite Graph for Polynomial System.

The dynamics of the various node in the graph can be described as:

$$\begin{aligned} f_1 &= b_1 \\ f_2 &= b_1 \\ a_1 &= \varphi_{(1,1)} f_1 \\ a_2 &= \varphi_{(2,2)} f_2 \\ f_4 &= a_1 a_2 b_1 \\ a_3 &= \varphi_{(3,3)} f_4 \end{aligned} \tag{3.4}$$

It can be seen that f_4 is a product of three random variable (i.e., a_1, a_2 and b_1). Mean and covariance of the node f_4 will depend upon mean and covariance of the node a_1, a_2 and b_1. To compute the mean and covariance of node f_4, which is the product of random variable, Bohrnstedt and Goldberger[3] formula is used. As discussed these formula are limited to bilinear system. To apply these formula on the polynomial system, *Multiplicative Property* is used. In other words polynomial system are converted to bilinear system using Multiplicative Property. Where, two or more random variable are combined to form a single pseudo node.

$$f_4 = a_1 a_2 b_1$$

Let us assume $\beta = a_1 a_2$. Node a_1 and a_2 are multiplied to form a psuedo node α. As a result, the polynomial term f_4 can be converted to bilinear term ($f_4 = \beta b_1$). The Bohrnstedt and Golderger formula can be modified as:

$$\begin{aligned} \mathbb{E}(a_1 a_2 b_1) &= \mathbb{E}(\beta b_1) = \mathbb{E}(\beta)\mathbb{E}(b_1) + \mathrm{Cov}(\beta b_1) \\ \mathrm{Var}[a_1 a_2 b_1] &= \mathrm{Var}[\beta b_1] = \mathrm{Cov}[\beta_1^2, b_1^2] + (\mathrm{Var}[\beta] + \mathbb{E}[\tilde{\beta}]^2) + (\mathrm{Var}[b_1] + \mathbb{E}[b_1]^2) - (\mathrm{Cov}[\beta, b_1] + \mathbb{E}[\beta]\mathbb{E}[b_1])^2 \\ \mathrm{Cov}[a_1 a_2 b_1, \tilde{c}] &= \mathrm{Cov}[\beta b_1, \tilde{c}] = \mathbb{E}[\beta]\mathrm{Cov}[b_1, \tilde{c}] + \mathbb{E}[b_1]\mathrm{Cov}[\beta, \tilde{c}] + \mathbb{E}[\triangle\beta \triangle \tilde{b}_1 \triangle c] \end{aligned} \tag{3.5}$$

Therefore, while using the Multiplicative Property, firstly, mean and covariance of the $\beta = a_1 a_2$ is found using the Bohrnstedt and Goldberger's formula. Then, the central moment of β found is used in the Equation 3.5; as a result, the polynomial term is converted to the bilinear terms, and its central moments can now easily be calculated. In this way, the

central moment of the polynomial term of n degree can be evaluated using Bohrnstedt and Goldberger's formula.

Taking the multiplicative property into account, the input (b_i) and output (a_i) node sets can be defined as:-

$$b = b_1$$
$$a = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}$$

On comparing with the standard class of Bilinear Equation in 3.1. The system dynamics for the network in Equation 3.4 can be defined as:

$$a_1 = \theta_{1,(1)}a_1 + \theta_{1,(2)}a_2 + \theta_{1,(3)}a_3 + \kappa_{1,(1)}b_1 + \psi_{1,(1,1)}a_1b_1 + \psi_{1,(2,1)}a_2b_1 + \psi_{1,(3,1)}a_3b_1 + \omega_{1,(1,1)}b_1b_1$$
$$a_2 = \theta_{2,(1)}a_1 + \theta_{2,(2)}a_2 + \theta_{2,(3)}a_3 + \kappa_{2,(1)}b_1 + \psi_{2,(1,1)}a_1b_1 + \psi_{2,(2,1)}a_2b_1 + \psi_{2,(3,1)}a_3b_1 + \omega_{2,(1,1)}b_1b_1$$
$$a_3 = \theta_{3,(1)}a_1 + \theta_{3,(2)}a_2 + \theta_{3,(3)}a_3 + \kappa_{3,(1)}b_1 + \psi_{3,(1,1)}a_1b_1 + \psi_{3,(2,1)}a_2b_1 + \psi_{3,(3,1)}a_3b_1 + \omega_{3,(1,1)}b_1b_1$$

Therefore based on the dynamics in Equation 3.4:

$$\begin{aligned} a_1 &= \kappa_{1,(1)}b_1 \\ a_2 &= \kappa_{2,(1)}b_1 \\ a_3 &= \psi_{3,(1,1)}a_1a_2b_1 = \beta b_1 \end{aligned} \tag{3.6}$$

Above equation can be expressed in terms of standard SOP as:

$$\begin{aligned} a_1 &= \varphi_{(1,1)}f_1(a,b) \\ a_2 &= \varphi_{(2,2)}f_2(a,b) \\ a_3 &= \varphi_{(3,4)}f_4(a,b) \end{aligned} \tag{3.7}$$

Coefficient Matrix/ Function Term	Variable
$\varphi_{(1,1)}$	$\kappa_{1,(1)}$
$\varphi_{(2,2)}$	$\kappa_{2,(2)}$
$\varphi_{(3,4)}$	$\psi_{3,(1,1)}$
f_1	b_1
f_2	b_1
f_4	$a_1a_2b_1 = \beta b_1$

Table 3-4.1: Relation between Coefficient Matrix/Function Term and Variables.

Table 3-4.1 shows the relationship between the coefficient matrix terms, function terms, and various variables. For example, where $\varphi_{(i,j)}$ are the elements of the coefficient matrix and f_h represents the relationship between input and output.

Finally, a modified Bohrnstedt and Goldberger formula is used to implement the MPBUP algorithm on the polynomial bipartite graph network for uncertainty propagation. Validation is performed for the output result in the subsequent chapter.

Chapter 4

Algorithm Implementation and Validation

4-1 Monte Carlo Validation

Monte Carlo (MC) Simulation is used to validate the algorithm and output generated by the algorithm. MC simulation works on the principle of random sampling. It predicts and compares output values based on the estimated range versus a fixed input value. For example, to validate the algorithm, a set of randomly generated values following a particular type of distribution (i.e., Normal, Uniform, Poisson, etc.) are given as an input in the algorithm, and corresponding output values are generated for the output nodes.

The set of output values generated for the output node is used to compute Empirical Mean. The empirical mean value is compared against the output generated for the fixed input value (i.e., Analytical mean) to check whether there is a convergence between analytical mean and empirical mean to validate the algorithm. According to the Law of Large Numbers (LLN), the average of the result obtained from a large number of trials should be close to the predicted value of trials and will tend to become closer to the predicted value as more trials are performed. LLN guarantee stability for the average of some random events. MC simulation is conducted for our algorithm to validate the mean and covariance matrix generated for the output nodes. A detailed explanation of how to perform MC simulation is described in Appendix A. Empirical Mean for x random samples can be defined as:

$$\mathbb{E}(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i) \tag{4.1}$$

Through MC simulation, various analyses can be drawn for the output generated under the corresponding random input variable for various sample size such as mean error propagation, standard deviation and input/output data distribution. These analyses are used to validate the output values generated in terms of the spread of the distribution, error generated at each output sample, etc.

Output generated under the MC simulation is used to check the deviation of the output generated at each sample of the data set. It is done by computing the error between the Empirical value (i.e., the value obtained from each testing sample) and Estimated Value (i.e., analytical value generated from the algorithm). Error obtained for each sample in a data set can compute the overall mean error propagated ($\triangle$). The mean error propagated can be calculated using the formula:

$$\bar{\triangle} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}(y_i) - \mu_y \tag{4.2}$$

Where, μ_y is estimated mean value and $\mathbb{E}(\mu_y)$ is the empirical value generated for each sample value.

4-2 Algorithm Implementation for Bilinear Systems

The algorithm is implemented on MATLAB. The figure 4.1 shows the corresponding graph generated in the MATLAB for the network in the figure 3.1.

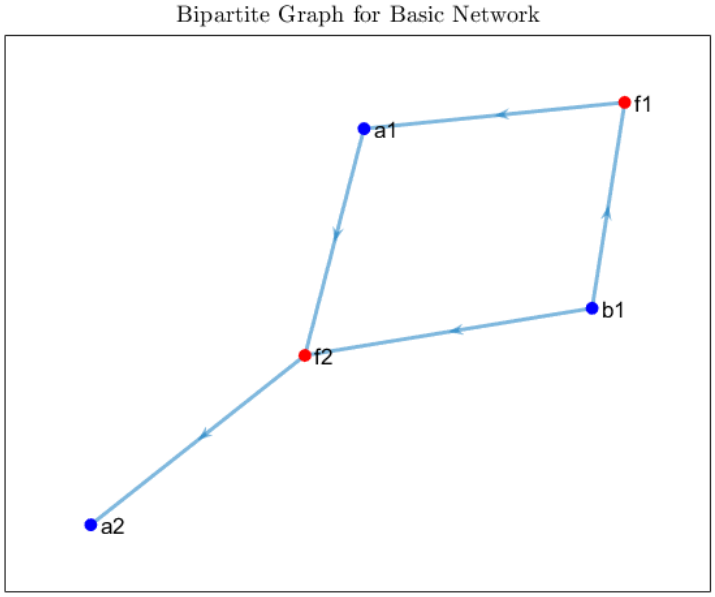


Figure 4.1: Bipartite Graph Generated in MATLAB for the Basic Network.

In the figure 4.1 variable and factor nodes are represented in red and blue colour respectively. Finally now MPBUP algorithm is implemented to propagate moments of the input node (i.e. b_1) to derive the moments of the output node (i.e., a_1 and a_2) respectively. The coefficient matrix (i.e., φ) is also given as input into the algorithm. The dynamics of various node in

the Basic Network based on input node (b_1) in Figure 4.1 can be described as:

$$\begin{aligned} f_1 &= b_1 \\ f_2 &= a_1 b_1 \\ a_1 &= \varphi_{(1,1)} f_1 \\ a_2 &= \varphi_{(2,2)} f_2 \end{aligned}$$

Therefore the algorithm can be initialized as:

For the input node b_1, mean and variance can be defined as $\mu(b_1) = 5$ and $\sigma(b_1) = 0.25$ respectively.

The non zero coefficient matrix elements (ie., $\varphi_{(1,1)}$, $\varphi_{(2,2)}$) can be defined as:

$$\varphi = \begin{pmatrix} 0.85 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.65 & 0 & 0 & 0 & 0 \end{pmatrix}$$

At the end of the algorithm , mean ($\bar{d}$) and covariance matrix (Σ_d) of the vector (d) can be defined as:

$$\begin{aligned} \bar{d} &= \begin{bmatrix} 3.2500 & 13.9506 & 5.0000 & 5.0000 & 16..4125 \end{bmatrix} \\ \Sigma_d &= \begin{pmatrix} 0.1056 & 0.7631 & 0.1625 & 0.1625 & 1.0563 \\ 0.7631 & 1.0567 & 1.1741 & 1.1741 & 1.0567 \\ 0.1625 & 1.1741 & 0.2500 & 0.2500 & 1.6250 \\ 0.1625 & 1.1741 & 0.2500 & 0.2500 & 1.6250 \\ 1.0563 & 1.0567 & 1.6250 & 1.6250 & 1.4625 \end{pmatrix} \end{aligned} \tag{4.3}$$

Mean of the output node a_1 and a_2 generated through the algorithm can be defined as 3.2500 and 14.0887 respectively, when the mean of 5 is given as an input to the input node b_1.

4-3 Validation of the Bilinear System

MC simulation in this section is conducted to validate the output generated for the basic network in the Figure 3.1.

4-3-1 MC Mean Validation

MC simulation is done to validate the output mean generated for the output node, such as a_1 and a_2 for the network in the Figure 3.1, when an input is given to the input node b_1. To validate the algorithm analytical mean value result generated in Equation 4.3 is compared against the empirical mean value for various sample sizes.

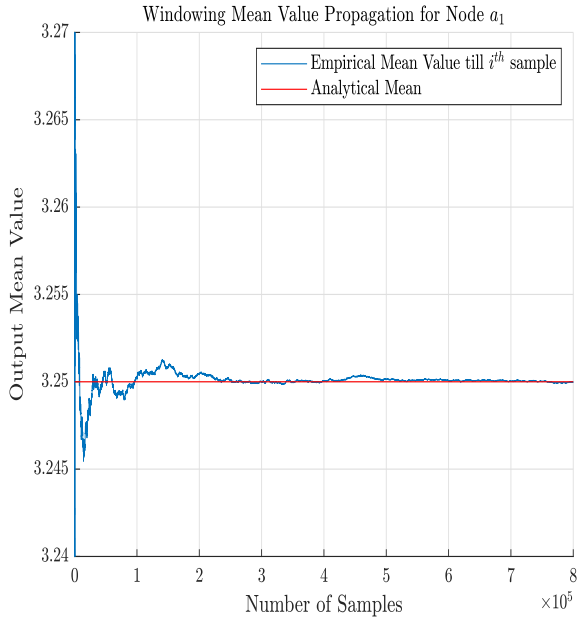
MC simulation is conducted for the various sample size for the Network in figure 3.1. Input given to the input node b_1 is random numbers. These random numbers are normally distributed along with the mean $\mu(b_1)$ of 5 and standard deviation $\sigma(b_1)$ of 0.25 for various sample sizes. During MC simulation mean error propagated between analytical and empirical mean value is also calculated using Equation 4.2 .

Table 4-3.1 shows the comparison between the analytical and empirical mean for various sample sizes, and it also shows the mean error propagated between the analytical and empirical value for the output node (a_1 and a_2) for various sample size. We can also see a convergence between the analytical and empirical mean as the sample size increases. Around 1000000 samples, both analytical and empirical mean becomes identical, and the mean error becomes zero. The validation experiment shows the results of prediction based on the fixed input given into the induced network is comparable to those generated by the MPBUP algorithm under a variety of uncertain knowledge domains.

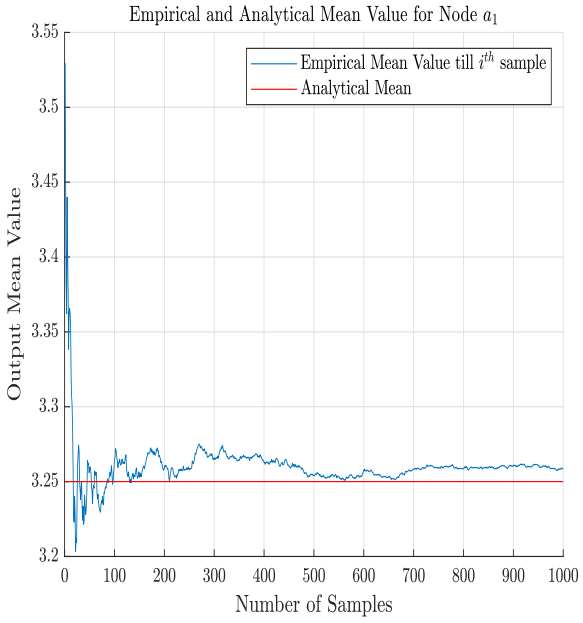
	Sample Size	a_1	a_2	Error (a_1)	Error (a_2)
Analytical Mean		3.2500	13.9506		
Empirical Mean	10	3.4529	14.2557	0.2029	0.3051
Empirical Mean	100	3.2761	13.9633	0.0261	0.0127
Empirical Mean	1000	3.2364	13.9481	-0.0136	-0.0025
Empirical Mean	10000	3.2517	13.9469	0.0017	-0.0037
Empirical Mean	100000	3.2500	13.9503	0.0003	0.0003
Empirical Mean	500000	3.2502	13.9505	0.0002	0.0001
Empirical Mean	1000000	3.2500	13.9506	0.0000	0.0000

Table 4-3.1: The Analytical and Empirical Mean Value of the Output Node a_1 and a_2.

Figure 4.2 and 4.3 shows the propagation of the mean output value for output nodes a_1 and a_2 respectively. Empirical mean value is calculated for each sample iteration of the output node a_1 and a_2 during MC simulation.

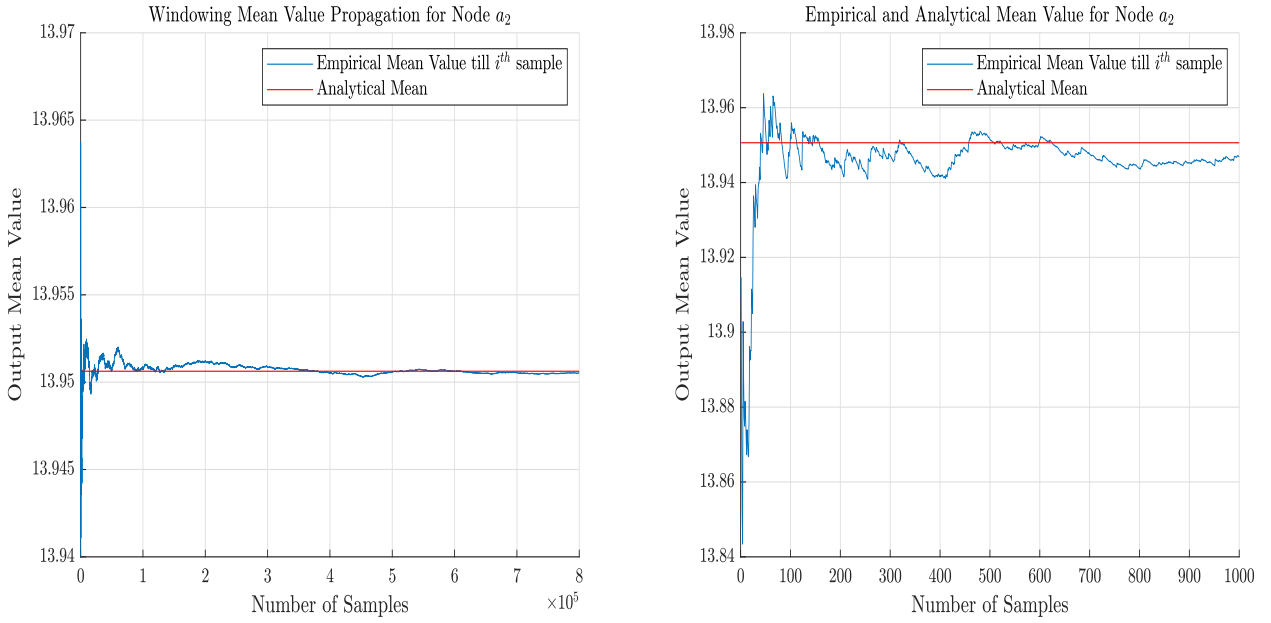


(a) Empirical and Analytical Mean Value.



(b) Empirical and Analytical Mean Value for 1000 Samples.

Figure 4.2: Comparison between Empirical and Analytical Mean Value Propagation for Node a_1 in Basic Network in MC Simulation.



(a) Comparison of Empirical and Analytical Mean Value. **(b)** Empirical and Analytical Mean Value for 1000 Samples.

Figure 4.3: Comparison between Empirical and Analytical Mean Value Propagation for Node a_2 in Basic Network in MC Simulation.

Figures 4.2 and 4.3 depicts the comparison between the behavior of the empirical mean value generated for the output node a_1 and a_2 respectively under a random input sample size of 800000 (shown as blue line), and the predicted analytical mean value (shown as red line). For empirical mean value propagation, it is generated by evaluating the mean value till i^{th} samples at every i^{th} instant.

The Figures in the left 4.2a and 4.3a shows the magnified view along the y axis of empirical mean value propagation compared to the analytical mean value . We can see that around 800000 samples, both analytical and empirical mean values start converging, and error becomes negligible by LLN. Figures in the right 4.2b and 4.3b shows the initial distortion between empirical mean and analytical mean value for a small sample size of 1000 samples. We can see that there is always a huge error present between the analytical and empirical mean value for a small size. Figure 4.2 and 4.3 is used for supporting the results obtained in Table 4-3.1 for the validation of the output means and to see the convergence between the analytical and empirical mean value.

Figure 4.4 shows the distribution of output sample values generated for node a_1 for corresponding 100000 random input samples in MC simulation. Figure 4.4 also shows the mean value and 99% confidence limits (using red dotted line). We can see output sample values for node a_1 also follow a normal distribution similar to the input. Regarding the optimality of the algorithm, we can see that output values lies within a confidence interval of 95% (i.e., 2.6002 and 3.8996). Confidence intervals represent the interval that would contain the actual mean value when the random sample is drawn many times. Therefore, we can conclude that under the uncertain or random conditions, there are 95% chances that output generated by the algorithm will contain the actual value of node a_1 with the confidence limit, and the

algorithm will not generate any extreme values under uncertain conditions.

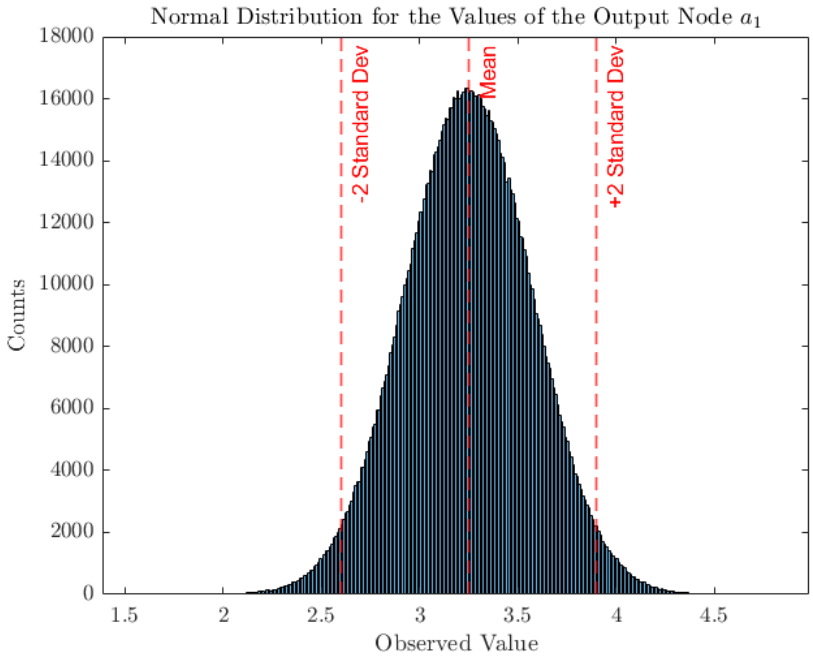


Figure 4.4: Distribution of Output Sample Values Generated for the Output Node a_1 using MC Simulation in Basic Network.

4-3-2 MC Covariance Validation

Similar to mean, MC simulation is also used to validate the covariance matrix generated for the network in the figure 3.1. To validate the covariance matrix, the analytical result generated in 4.3 is compared against the empirical values for various sample sizes.

MC Simulation for the validation of the covariance matrix is done under the similar condition used to validate the output mean value. It can be seen that around 1000000 samples, there is a convergence between the analytical covariance matrix and empirical covariance matrix. Therefore, the validation experiment confirms the comparable performance between the output covariance matrix generated based on the prediction of the induced network and under the variety of uncertain knowledge. The covariance matrix in 4.4 shows empirical covariance matrix generated at 1000000 samples. On comparing the empirical covariance matrix generated in Equation 4.4 with analytical covariance matrix generated in Equation 4.3. We can see a convergence between both the results, and hence, it validates the output covariance matrix generated under certain fixed input conditions versus under some uncertain conditions.

$$\Sigma_d = \begin{pmatrix} 0.1056 & 0.7631 & 0.1625 & 0.1625 & 1.0563 \\ 0.7631 & 1.0567 & 1.1741 & 1.1741 & 1.0567 \\ 0.1625 & 1.1741 & 0.2500 & 0.2500 & 1.6250 \\ 0.1625 & 1.1741 & 0.2500 & 0.2500 & 1.6250 \\ 1.0563 & 1.0567 & 1.6250 & 1.6250 & 1.4625 \end{pmatrix} \tag{4.4}$$

Therefore, through the MC Simulation Output generated by the MPBUP algorithm is successfully validated. The covariance matrix generated under various sample size in MC simulation can be found in the Appendix C.

4-4 Algorithm Implementation for Polynomial System

In the previous chapter, the bottleneck of the MPBUP algorithm is discussed related to evaluating the mean and covariance of a polynomial node. To overcome that problem, Multiplicative Property between two terms is used. It converts the desired higher degree of the polynomial terms into a lower degree of terms (i.e., Bilinear/Linear). The main reason to use multiplicative property is that Goldberger and Bohrnstedt's formula for mean and covariance for the product of random variable can be used. To validate the research, the MPBUP algorithm is applied on the polynomial system in figure 3.7 using the multiplicative property, and to successfully validate the result, and MC simulation is performed. Figure 4.5 shows the corresponding bipartite graph generated in MATLAB.

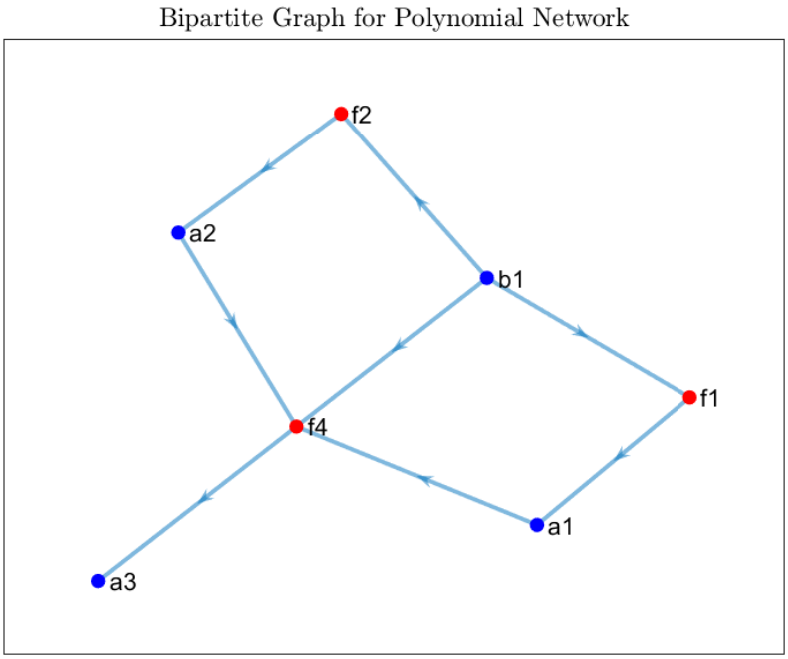


Figure 4.5: Bipartite Graph Generated in MATLAB for Polynomial System.

Dynamics of the graph is defined in the previous chapter. $\beta = a_1 a_2$ is assumed. β act as pseudo node which is a combination of two random variable node (i.e., in this particular example). As a result polynomial node (f_3) is converted to a bilinear node. Therefore the number of inputs (b_i) and outputs (a_i) can be defined as $m = 1$ and $n = 3$ respectively.

β is an extra pseudo node used for our reference (i.e., not present). The reason to model an extra node is to model the system in terms of bilinear system representation by converting

the polynomial term and to define the coefficient matrix for the algorithm implementation. In the previous chapter, system dynamics were converted to the standard bilinear equation representation as in Equation 3.6 and further to standard SOP as in Equation 3.7. Algorithm is implemented to propagate uncertainty through the network from the input node (i.e., b_1) to the output nodes (i.e. a_1, a_2, a_3). In other words, the moments of the input nodes are propagated into the network to find the moments of the output node. The size of the coefficient matrix can be defined as $\varphi \in \mathbb{R}^{n \times n_f}$. Where $n = 3$ and n_f correspond to $(n+m)(m+1)$. In our case ($n_f = (3+1)(1+1) = 8$) and $\varphi \in \mathbb{R}^{3\times 8}$.

The mean and variance of the node b_1 can be defined as $\mu(b_1) = 5$ and $\sigma(b_1) = 0.25$ respectively and algorithm is initialized.

For Implementation, the coefficient matrix (φ) can be defined as, where the nonzero elements of the coefficient matrix (i.e., $\varphi_{(1,1)}$, $\varphi_{(2,2)}$, $\varphi_{(4,4)}$) are represented in the table 3-4.1.

$$\varphi = \begin{pmatrix} 0.65 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.85 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.95 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Inputs given to the algorithm designed in MATLAB are coefficient matrix (φ), mean $\mu(b_1)$ and variance $\sigma(b_1)$. At the end of the algorithm, mean ($\bar{d}$) and covariance matrix (Σ_d) for the vector (d) can be defined as:

$$\bar{d} = \begin{bmatrix} 3.2500 & 4.2500 & 67.5777 & 5.0000 & 5.0000 & 5.0000 & 71.3444 \end{bmatrix}$$

$$\Sigma_d = \begin{pmatrix} 0.1056 & 0 & 4.5052 & 0.1625 & 0.1625 & 0 & 4.9919 \\ 0 & 0.1806 & 5.8914 & 0.2125 & 0 & 0.2125 & 6.5278 \\ 4.5052 & 5.8914 & 0 & 8.8283 & 8.8283 & 8.8283 & 64.1988 \\ 0.1625 & 0.2125 & 8.8283 & 0.2500 & 0.2500 & 0.2500 & 9.7821 \\ 0.1625 & 0 & 8.8283 & 0.2500 & 0.2500 & 0 & 9.7821 \\ 0 & 0.2125 & 8.8283 & 0.5000 & 0 & 0.2500 & 9.7821 \\ 4.9919 & 6.5278 & 64.1988 & 9.7821 & 9.7821 & 9.7821 & 71.1344 \end{pmatrix} \tag{4.5}$$

Therefore, the mean generated for the output node a_1, a_2 and a_3 through the algorithm can be defined as 3.2500, 4.2500 and 67.5777 respectively.

4-5 Validation of a Polynomial System

Similar to Bilinear System, a validation test is also performed on the polynomial system in Figure 4.5 using the MC simulation. MC simulation is performed to validate the results obtained through the algorithm in equation 4.5. A similar setup compared to a bilinear system is used. Where Input given to the input node are random numbers of a particular distribution and the corresponding outputs are generated. Outputs are used to compute the empirical mean, which is compared with the analytical mean.

4-5-1 MC Mean Validation

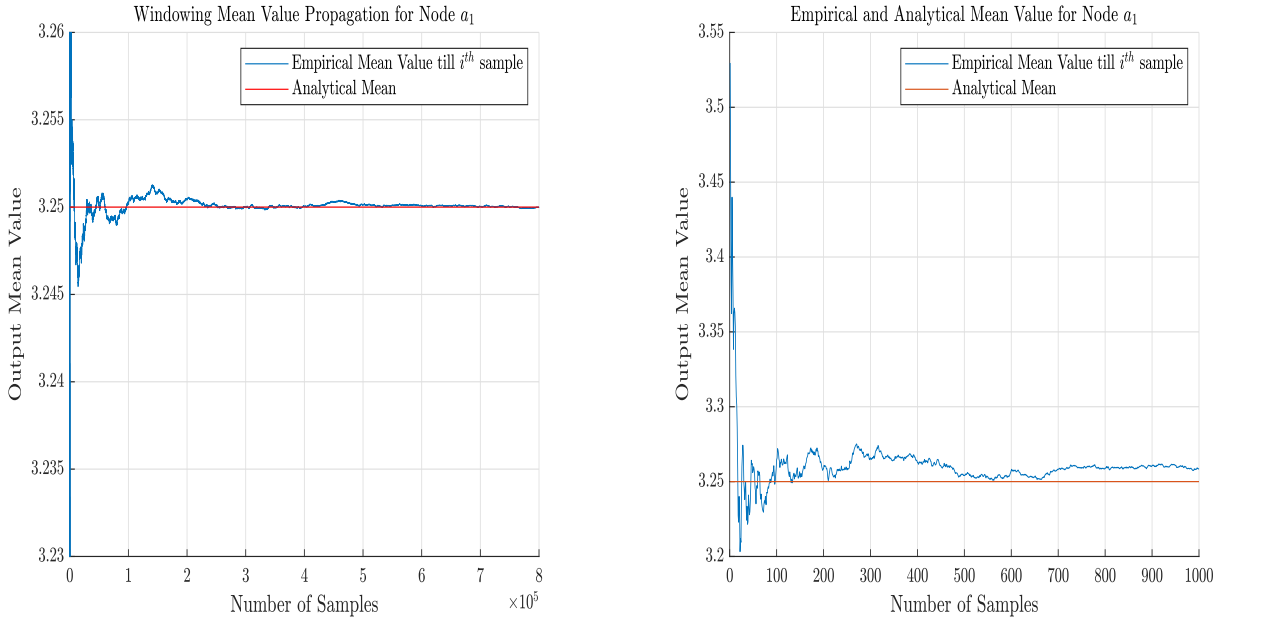
MC simulation is conducted to validate the output mean obtained for the output node (a_1, a_2 and a_3) for the network in the figure 4.5. The analytical mean value is obtained through

the algorithm in Equation 4.5. The empirical mean is computed by giving input to the input node b_1. Input are random numbers, which are normally distributed with the mean $\mu(b_1)$ of 5 and standard deviation $\sigma(b_1)$ of 0.25 for various sample sizes.

	Sample Size	a_1	a_2	a_3	Error (a_1)	Error (a_2)	Error (a_3)
Analytical Mean		3.2500	4.2500	67.5777			
Empirical Mean	10	3.4529	4.5153	71.9255	-0.2029	-0.2653	-4.3478
Empirical Mean	100	3.2761	4.2841	67.7587	0.2239	-0.0350	0.0010
Empirical Mean	1000	3.2364	4.2322	67.5413	0.0136	0.0178	0.0364
Empirical Mean	10000	3.2517	4.2523	67.5248	0.0017	0.0023	0.0529
Empirical Mean	100000	3.2500	4.2500	67.5737	0.0000	0.0000	0.0400
Empirical Mean	500000	3.2502	4.2503	67.5763	-0.0002	-0.0003	0.0014
Empirical Mean	1000000	3.2502	4.2503	67.5809	-0.0002	-0.0003	-0.0032
Empirical Mean	5000000	3.2500	4.2500	67.5777	0.0000	0.0000	0.0000

Table 4-5.1: Analytical and Empirical Mean Value for the Output Nodes a_1, a_2 and a_3.

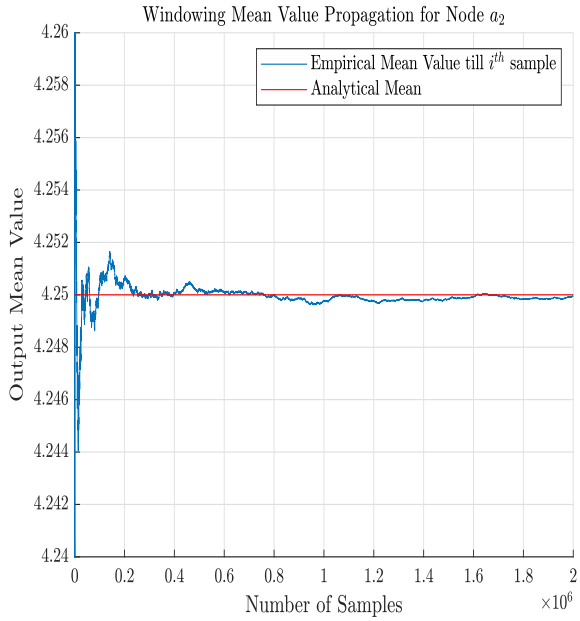
Table 4-5.1 shows the comparison between the analytical and empirical mean for various sample sizes. It also shows the mean error propagated between the analytical and empirical mean value for the output node (a_1, a_2 and a_3) for various sample sizes. We can see that there is a convergence between the analytical mean and empirical mean around 5000000 random samples, and the mean error becomes zero for all the output nodes. The validation experiment also validates the comparable performance of the output means value generated through the algorithm for the polynomial system.



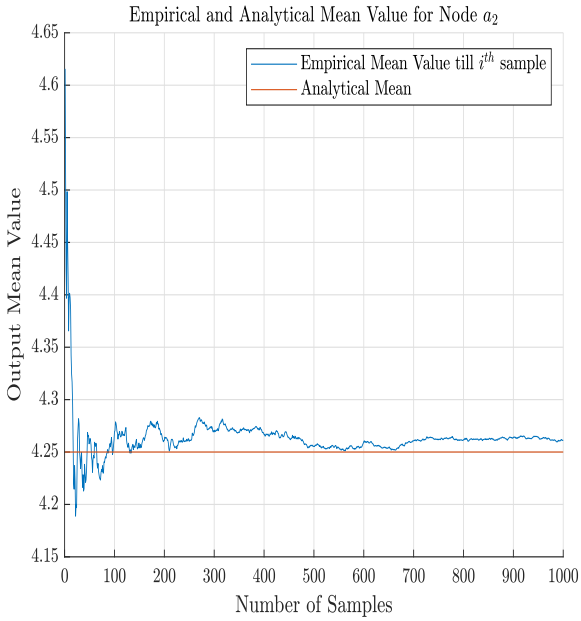
(a) Empirical and Analytical Mean Value.

(b) Empirical and Analytical Mean Value for 1000 Samples.

Figure 4.6: Comparison between Empirical and Analytical Mean Value Propagation for Node a_1 in Polynomial System for MC Simulation.

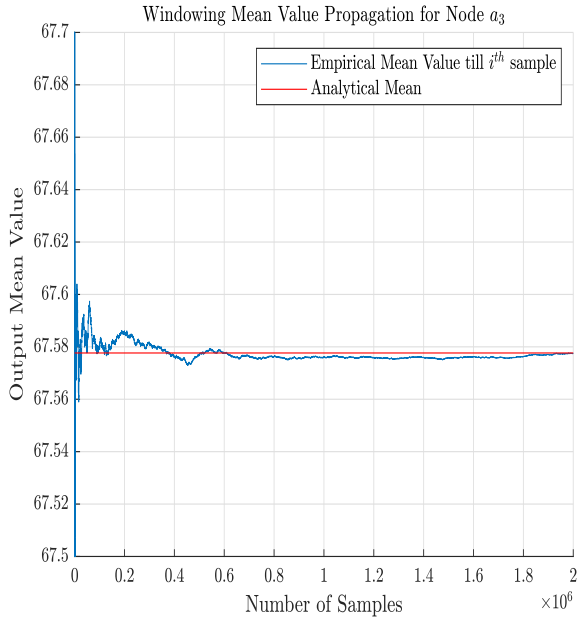


(a) Empirical and Analytical Mean Value.

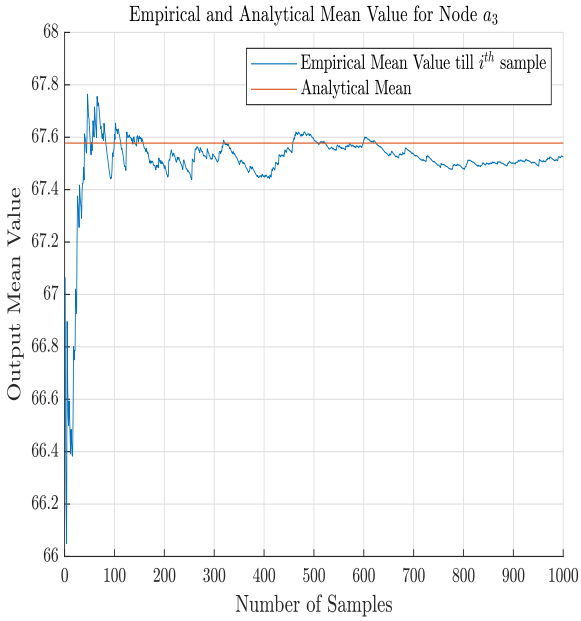


(b) Empirical and Analytical Mean Value for 1000 Samples.

Figure 4.7: Comparison between Empirical and Analytical Mean Value Propagation for Node a_2 in Polynomial System for MC Simulation.



(a) Empirical and Analytical Mean Value.



(b) Empirical and Analytical Mean Value for 1000 Samples.

Figure 4.8: Comparison between Empirical and Analytical Mean Value Propagation for Node a_3 in Polynomial System for MC Simulation.

Figures 4.6, 4.7 and 4.8 depicts the comparison between the behavior of the empirical mean value generated for the output node a_1, a_2 and a_3 respectively under a large random input samples (shown as blue line) which are normally distributed, and the predicted analytical mean value (shown as red line). For empirical mean value propagation, it is generated by evaluating the mean value till i^{th} samples at every i^{th} instant.

The Figures in the left 4.6a 4.7a and 4.8a shows the magnified view along y axis of empirical mean value propagation compared to the analytical mean value. We can see that around 2000000 samples, both analytical and empirical mean values start converging, and error becomes negligible for output node a_1, a_2 and a_3 by LLN. Figures in the right 4.6b, 4.7b and 4.8b shows the initial distortion in the empirical mean compared to the analytical mean value for a small sample size of 1000 samples. We can see that there is always a huge error present between the analytical and empirical mean value for small sample size. Figure 4.6, 4.7 and 4.8 are used to support the results obtained under the MC for the output mean value in Table 4-5.1 and to validate the output mean generated through the MPBUP algorithm by seeing the convergence between the analytical and empirical mean value.

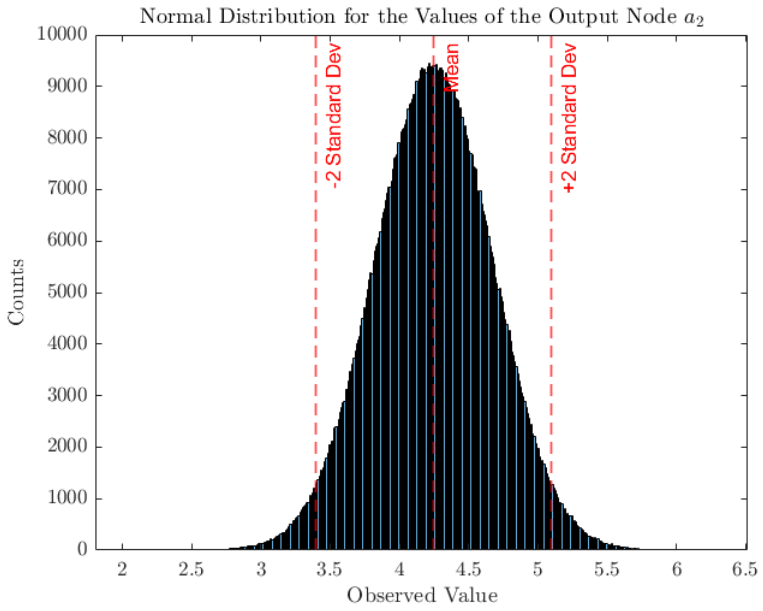


Figure 4.9: Distribution of Output Sample Values Generated for the Output Node a_2 using MC Simulation for Polynomial System.

Similar to the analysis done in basic network implementation. Figure 4.9 shows the distribution of output sample values generated for node a_2 under MC simulation for 1000000 random input samples. Figure 4.9 also shows the mean value (i.e., 4.2500) for the output samples and 95% confidence limit (using red dotted line). We can see that the output sample value generated for node a_2 also follows a normal distribution similar to the input. Regarding the optimality of the algorithm, we can see that output values lie within a confidence interval of 95% (i.e., between 3.4001 and 5.0994). Therefore it can be concluded that under the uncertain or random conditions, there are 95% chances that output generated by the algorithm will contain the actual value of node a_2. Additionally, we can also say that algorithm will

not generate any extreme values under uncertain conditions for all other output nodes (i.e., a_1 and a_3), similar behavior is observed.

4-5-2 MC Covariance Validation

Similar to validating the output mean, MC simulation is also performed to validate the output covariance matrix. To validate the covariance matrix, the analytical covariance matrix generated in Equation 4.5 is compared against the empirical values for various sample sizes. MC simulation is also conducted in a similar condition used for the validation of output mean value. MC simulation shows the convergence between the analytical covariance matrix and empirical covariance matrix around 1000000 samples. Therefore, MC simulation confirms the comparable performance of the output covariance matrix generated through the algorithm. 4.6 shows empirical covariance matrix generated for 1000000 samples.

$$\Sigma_d = \begin{pmatrix} 0.1056 & 0 & 4.5052 & 0.1625 & 0.1625 & 0 & 4.9919 \\ 0 & 0.1806 & 5.8914 & 0.2125 & 0 & 0.2125 & 6.5278 \\ 4.5052 & 5.8914 & 0 & 8.8283 & 8.8283 & 8.8283 & 64.1988 \\ 0.1625 & 0.2125 & 8.8283 & 0.2500 & 0.2500 & 0.2500 & 9.7821 \\ 0.1625 & 0 & 8.8283 & 0.2500 & 0.2500 & 0 & 9.7821 \\ 0 & 0.2125 & 8.8283 & 0.5000 & 0 & 0.2500 & 9.7821 \\ 4.9919 & 6.5278 & 64.1988 & 9.7821 & 9.7821 & 9.7821 & 71.1344 \end{pmatrix} \tag{4.6}$$

The validation test conducted for output mean and covariance generated through the algorithm and various other analyses shows that the MPBUP algorithm can propagate uncertainty into a polynomial system. However, it also works well, considering different uncertain conditions, and using Goldberger and Bohrnstedt's formula to calculate mean and covariance for polynomial nodes. Therefore it can be concluded that with a little modification in the dynamics of the system (i.e., converting all the polynomial terms present in the dynamics to bilinear terms), the MPBUP algorithm can be applied to the system. The only thing that needs to be considered is converting all the polynomial node terms present in the graph to bilinear terms.

Chapter 5

MPBUP Algorithm for Threshold Bounding

In the previous chapter, the MPBUP algorithm is discussed. The main aim of the MPBUP algorithm is to propagate the random or stochastic quantities (i.e., uncertainty) affecting the system through time steps. As minimal knowledge about the random variable is available, it is not easy to propagate it in the next step and find its overall effect. One of the main applications of the MPBUP algorithm can be formulated to propagate uncertainty into the system to determine a robust threshold to minimize FAR. Therefore, this chapter gives a detailed discussion on determining a robust threshold using the MPBUP algorithm.

5-1 MPBUP algorithm on the State Space Model

State Space Model (SSM) is a mathematical model of any physical system, where the set of input (u), output (y), and state (x) variables are related by a first-order differential equation. Thus, state variables represent the state of the entire system at any given instant of time. Input Variables are defined as the input provided in any physical system whereas, and output variables are defined as the output generated from the physical system based upon the corresponding input.

An State Space Model (SSM) is taken into account to apply the Message Passing Bilinear Uncertainty Propagation (MPBUP) algorithm to propagate uncertainty into any physical system for robust threshold detection. State Space Model (SSM) defines the dynamics of the system, taking uncertainty into account at every time instant. Therefore, to understand the modeling of the state-space system for the Message Passing Bilinear Uncertainty Propagation (MPBUP) algorithm, a $2nd$ order system is considered. The algorithm is implemented on the $2nd$ order system, and the output generated through the algorithm is validated.

Equation 5.1 shows the $2nd$ order system. where x_1 and x_2 represents the states acting on the system at particular time instant. $x_{1_{+1}}$ and $x_{2_{+1}}$ represents the state of the system at

next time step. u_1 and u_2 are input acting in the system. Block matrix A and B represent the state and input matrix, respectively.

$$\begin{pmatrix} x_{1_{+1}} \\ x_{2_{+1}} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} B_{11} \\ B_{21} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \tag{5.1}$$

Where, $x \in \mathbb{R}^2$ and $u \in \mathbb{R}^2$.

5-1-1 Modelling of State Space Model

On expanding equation in 5.1 can be rewritten as:

$$\begin{aligned} x_{1_{+1}} &= A_{11}x_1 + A_{12}x_2 + B_{11}u_1 \\ x_{2_{+1}} &= A_{21}x_1 + A_{22}x_2 + B_{21}u_2 \end{aligned} \tag{5.2}$$

Equation 5.2 represents the $2nd$ order SSM, where the input u_1 and u_2 acting in the model are known and deterministic (i.e., they remain constant through the whole time step). x_1 and x_2 are the state of the system at that particular time instant, whose mean and covariance are known from the previous time step. As a result, they act as an input to find the system's state for the next time step. x_{+1} and x_{+2} act as the output in the SSM, as x_{+1} and x_{+2} values are solved at that particular time step. As x_{+1} and x_{+2} values are quantities to be solved at a particular time instant. To apply MPBUP algorithm, input (b) and output (a) for the algorithm can be defined as:

$$\begin{aligned} a &= \begin{bmatrix} x_{1_{+1}} & x_{2_{+1}} \end{bmatrix} \\ b &= \begin{bmatrix} u_1 & u_2 & x_1 & x_2 \end{bmatrix} \end{aligned} \tag{5.3}$$

Therefore, $m = 4$ (i.e. number of input) and $n = 2$ (i.e., number of output).

To apply MPBUP algorithm on the SSM, Firstly, the SSM Equation in 5.2 is converted to a standard bilinear equation model with reference to 3.1. Therefore, the bilinear model for SSM can be defined as:

$$\begin{aligned} a_1 &= \kappa_{1,(1)}b_1 + \kappa_{1,(2)}b_2 + \kappa_{1,(3)}b_3 + \kappa_{1,(4)}b_4 \\ a_2 &= \kappa_{2,(1)}b_1 + \kappa_{2,(2)}b_2 + \kappa_{2,(3)}b_3 + \kappa_{2,(4)}b_4 \end{aligned} \tag{5.4}$$

On comparing 5.4 with 5.2, the relationship between the Variable in the SSM and Input/ Output Nodes described using Table 5-1.1

Input Node	Variable	Output Node	Variable
b_1	u_1	a_1	$x_{1_{+1}}$
b_2	u_2	a_2	$x_{2_{+!}}$
b_3	x_1		
b_4	x_2		

Table 5-1.1: Relation between Variable and Input-Output Nodes in SSM.

Similarly the relationship between coefficient elements in Equation 5.4 and SSM block elements in Equation 5.2 can be defined as:

Coefficient Terms	State Space Block Elements
$\kappa_{1,(1)}$	B_{11}
$\kappa_{1,(2)}$	0
$\kappa_{1,(3)}$	A_{11}
$\kappa_{1,(4)}$	A_{12}
$\kappa_{2,(1)}$	0
$\kappa_{2,(2)}$	B_{21}
$\kappa_{2,(3)}$	A_{21}
$\kappa_{2,(4)}$	A_{22}

Table 5-1.2: Relation between Coefficient and State Space Block Elements in SSM.

Finally the bilinear model is converted to standard SOP with reference to Equation 3.2, to define factor nodes, the coefficient matrix (φ) and consequently to draw bipartite graph. Size of the coefficient matrix can be defined as $\varphi \in \mathbb{R}^{n \times n_f}$, where $n = 2$ and n_f corresponds to $(n+m)(m+1)$. In our case ($n_f = (4+2)(4+1) = 30$) and $\varphi \in \mathbb{R}^{2 \times 30}$. The dynamics of the network in terms of SOP can be described as:

$$\begin{aligned} a_1 &= \varphi_{(1,1)} f_1 + \varphi_{(1,3)} f_3 + \varphi_{(1,4)} f_4 \\ a_2 &= \varphi_{(2,2)} f_2 + \varphi_{(2,3)} f_3 + \varphi_{(2,4)} f_4 \end{aligned} \tag{5.5}$$

f_h are the factor terms which defines the relation between input and output node and $\varphi_{(i,j)}$ corresponds to the elements of the coefficient matrix. The relationship between the factors (f_h) and variables ($a_1 \cdots a_n, b_1 \cdots b_m$) in Equation 5.5 can be expressed in terms of a bipartite graph in Figure 5.1.

Coefficient Matrix Elements/ Function Term	Coefficient Terms	State Space Block Elements
$\varphi_{(1,1)}$	$\kappa_{1,(1)}$	B_{11}
$\varphi_{(1,3)}$	$\kappa_{1,(3)}$	A_{11}
$\varphi_{(1,4)}$	$\kappa_{1,(4)}$	A_{12}
$\varphi_{(2,2)}$	$\kappa_{2,(2)}$	B_{22}
$\varphi_{(2,3)}$	$\kappa_{2,(3)}$	A_{21}
$\varphi_{(2,4)}$	$\kappa_{2,(4)}$	A_{22}
f_1	b_1	u_1
f_2	b_2	u_2
f_3	b_3	x_1
f_4	b_4	x_2

Table 5-1.3: Relation between Coefficient Matrix Elements/Function Terms, Coefficient Terms and State Space Block Elements in SSM.

Table 5-1.3 shows the relation between various variables involved in bilinear Equation 5.4 and the standard SOP Equation 5.5 with the SSM in Equation 5.1.

5-1-2 Algorithm Implementation

Based upon the dynamics in equation 5.5, the nodes can be divided into two sets such as:

$$\text{Variable Node}(\mathbb{V}) = \{a_1, a_2, b_1, b_2, b_3, b_4\}$$
$$\text{Factor Node}(\mathbb{F}) = \{f_1, f_2, f_3, f_4\}$$

Figure 5.1 shows the bipartite graph generated for the $2nd$ order system in MATLAB. The factor node set ($\mathbb{F}$) are represented in red in colour where as variable node set ($\mathbb{V}$) are represented in blue in colour.

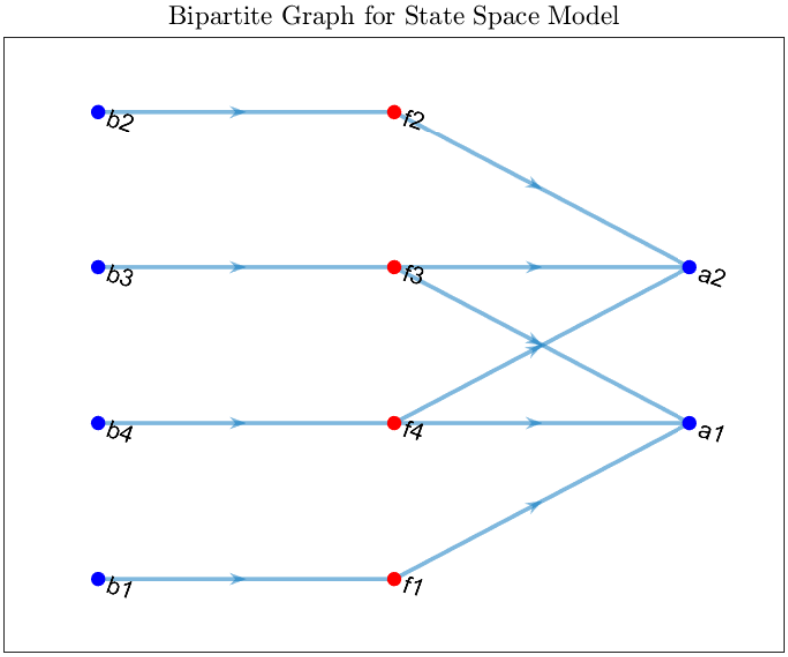


Figure 5.1: Bipartite Graph for the $2nd$ order system.

Therefore, the dynamics of the bipartite graph for the state space model can be defined as:

$$\begin{aligned} f_1 &= b_1 \\ f_2 &= b_2 \\ f_3 &= b_3 \\ f_4 &= b_4 \\ a_1 &= \varphi_{(1,1)} f_1 + \varphi_{(1,3)} f_3 + \varphi_{(1,4)} f_4 \\ a_2 &= \varphi_{(2,2)} f_2 + \varphi_{(2,3)} f_3 + \varphi_{(2,4)} f_4 \end{aligned}$$

MPBUP algorithm is implemented to propagate moment of the input nodes (i.e., b_1, b_2, b_3 and b_4) to derive the corresponding moment of the output nodes (i.e., a_1 and a_2) respectively.

For implementation purpose the non zero elements of the coefficient matrix (i.e., $\varphi_{(1,1)}$, $\varphi_{(1,3)}$, $\varphi_{(1,4)}$, $\varphi_{(2,2)}$, $\varphi_{(2,3)}$, $\varphi_{(2,4)}$) can be expressed as in the Table 5-1.4

Coefficient Matrix Element	Assumed Values
$\varphi_{(1,1)}$	1
$\varphi_{(1,3)}$	2
$\varphi_{(1,4)}$	1
$\varphi_{(2,2)}$	3
$\varphi_{(2,3)}$	1
$\varphi_{(2,4)}$	4

Table 5-1.4: Assumed Values for the Coefficient Matrix Elements in SSM .

Therefore:

$$\varphi = \begin{pmatrix} 1 & 0 & 1 & 3 & 0 & 0 \cdots 0 \\ 0 & 2 & 1 & 4 & 0 & 0 \cdots 0 \end{pmatrix}$$

The nonzero elements of the coefficient matrix in Table 5-1.4 represents the elements of the block matrix A and B in SSM. To get a better understanding on the relationship between the elements of the block matrix (i.e., A and B), coefficient matrix (i.e., φ) and the assumed values it can be referred to the Table 5-1.3 and 5-1.4 respectively.

Mean (μ) and variance (σ) for the inputs nodes (i.e., $b_1 \cdots b_4$) in the Figure 5.1 can be assumed as:

Node	Mean (μ)	Variance (σ)
b_1	5	0.25
b_2	5	0.25
b_3	3	0.5
b_4	3	0.5

Table 5-1.5: Mean and Variance for the Input Node in SSM.

At the end of the algorithm, mean ($\bar{d}$) and Variance (Σ_d) of vector (d) can be defined as:

$$\bar{d} = \begin{bmatrix} 17 & 25 & 5 & 5 & 3 & 3 & 5 & 5 & 3 & 3 \end{bmatrix}$$

$$\Sigma_d = \begin{pmatrix} 10.75 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10.75 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \end{pmatrix} \tag{5.6}$$

As we are only interested in the mean and covariance of the output node (a_1 and a_2), therefore we will extract the mean and covariance of the output node from $\bar{d}$ and Σ_d respectively. The mean and covariance of the output node generated through the algorithm can be defined as:

Node	Mean	Variance
a_1	17	10.75
a_2	25	10.75

Table 5-1.6: Mean and Variance for the Output Node in SSM.

5-2 Validation of a State Space Model

Similar to Bilinear and Polynomial Systems, validation test is also performed for the MPBUP algorithm implemented on the SSM, to validate the mean and variance generated for the output node (i.e., a_1,and a_2) found in table 5-1.6. Compared to the previous experiments for the validation, a similar setup is used. Random input numbers of particular distribution are given as input to the network in Figure 5.1 and the corresponding output is generated. Outputs of various sample sizes are used to obtain the empirical mean.

5-2-1 MC Mean Validation

MC simulation is conducted to validate the output mean obtained for the output node (a_1 and a_2) for the network in Figure 5.1. Analytical mean is obtained through the algorithm is shown in Table 5-1.6, which is compared against the empirical mean value obtained for various sample size in Table 5-2.2

As discussed in previous section input in the state space model are either constant and deterministic (i.e., u_1 and u_2) or are known from previous time step (i.e., x_1 and x_2). Therefore to conduct the MC simulation the elements of the coefficient matrix (φ) are assumed to be random variable. Random inputs are given into ($\varphi_{(1,1)}$, $\varphi_{(1,3)}$, $\varphi_{(1,4)}$, $\varphi_{(2,2)}$, $\varphi_{(2,3)}$, $\varphi_{(2,4)}$). These random inputs are normally distribution with their mean and variance defined as:

Node	Mean	Variance
$\varphi_{(1,1)}$	1	0.25
$\varphi_{(1,3)}$	2	0.25
$\varphi_{(1,4)}$	1	0.25
$\varphi_{(2,2)}$	3	0.25
$\varphi_{(2,3)}$	1	0.25
$\varphi_{(2,4)}$	4	0.25

Table 5-2.1: Mean and Variance of the Coefficient Matrix for the Generating Random Normal Distribution in SSM

Therefore, we can say that for this particular experiment, uncertainty is propagated through the non-zero elements of the coefficient matrix (φ), which signifies the uncertainty in the block matrix (A and B).

	Sample Size	a_1	a_2	Error (a_1)	Error (a_2)
Analytical Mean		17	25		
Empirical Mean	10	17.3956	24.9176	-0.3956	0.0824
Empirical Mean	100	17.3196	25.0931	-0.3196	-0.0931
Empirical Mean	1000	17.0210	24.9165	-0.0210	0.0835
Empirical Mean	10000	16.9273	25.0388	0.0727	0.0388
Empirical Mean	100000	16.9944	24.9954	0.0056	0.0446
Empirical Mean	500000	16.9953	24.9938	0.0047	0.0062
Empirical Mean	1000000	17.0033	25.0012	-0.0033	0.0012
Empirical Mean	2500000	17.0000	25.0000	0.0000	0.0000

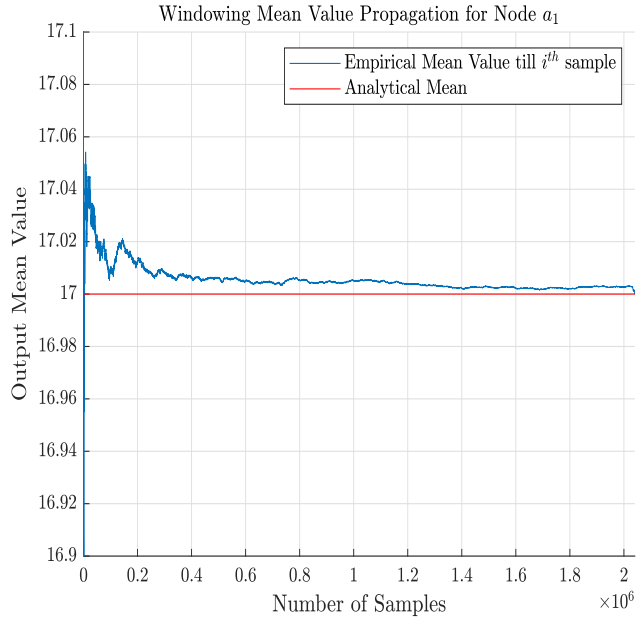
Table 5-2.2: Analytical and Empirical Mean Value for the Output Node a_1 and a_2 in SSM

Table 5-2.2 shows the comparison between the analytical and empirical mean values for various sample sizes and errors obtained between the analytical and empirical mean. It can be seen that at around 250000 samples, there is a convergence between the analytical and empirical mean value and, the error becomes zero. The validation experiment shows that the prediction result based upon the MPBUP algorithm is comparable to the result generated under a variety of uncertain knowledge.

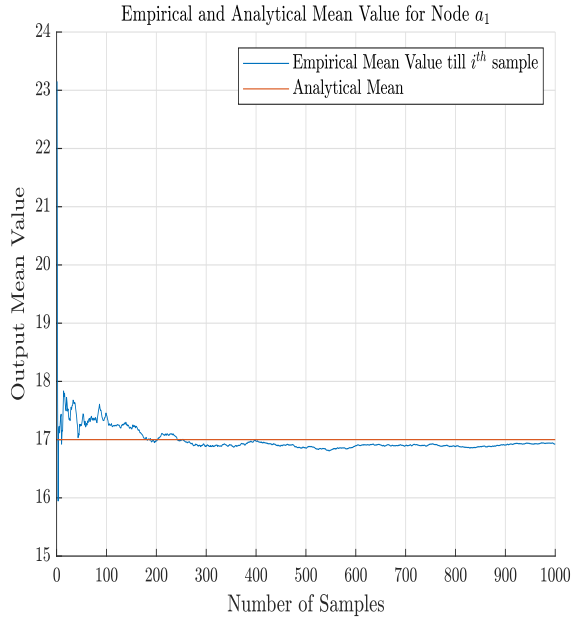
Figures 5.2 and 5.3 depicts the comparison between the behavior of the empirical mean value generated for the output node a_1 and a_2 respectively under a large random input samples (shown as blue line) which are randomly distributed, and the predicted analytical mean value (shown as red line). Empirical mean value at each i^{th} sample is generated by evaluating the mean value till i^{th} samples at every i^{th} instant.

The Figures in the left 5.2a and 5.3a shows magnified view along y axis of the empirical mean value propagation compared to the analytical mean value We can see that around 2000000 samples, both analytical and empirical mean values almost converges, and error becomes negligible for output node a_1 and a_2 by LLN. Figures in the right 4.6b, and 4.7b shows the initial distortion in the empirical mean compared to the analytical mean value for a small sample size of 1000 samples. There is always an error between the analytical and empirical

mean value for a small sample size.

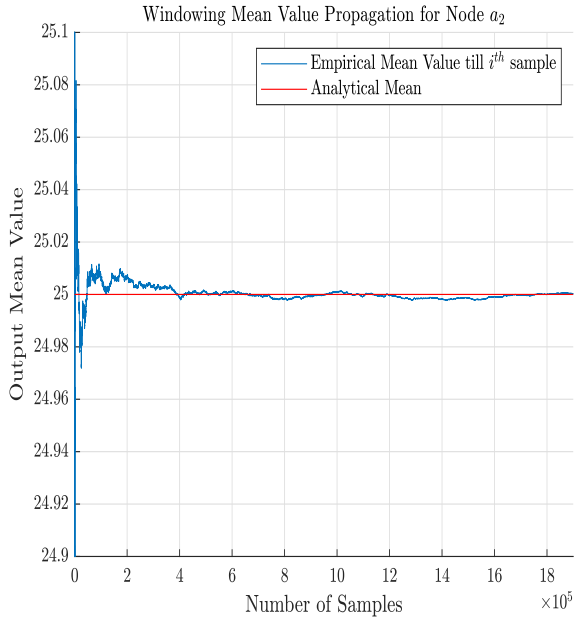


(a) Empirical and Analytical Mean Value.

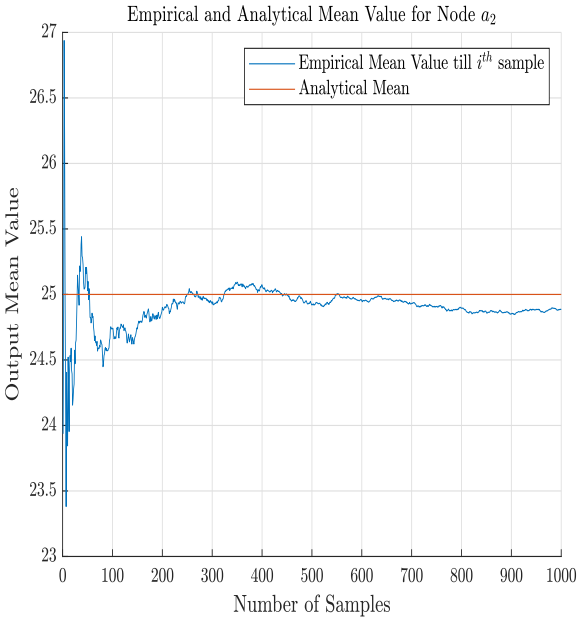


(b) Empirical and Analytical Mean Value for 1000 Samples.

Figure 5.2: Comparison between Empirical and Analytical Mean Value Propagation for Node a_1 in SSM for MC Simulation.



(a) Empirical and Analytical Mean Value.



(b) Empirical and Analytical Mean Value for 1000 Samples

Figure 5.3: Comparison between Empirical and Analytical Mean Value Propagation for Node a_2 in SSM for MC Simulation.

Figures 5.2 and 5.3 are used to support the result obtained in the Table 5-2.2 and to validate the output mean value generated through the algorithm MPBUP by seeing convergence between analytical and empirical mean value.

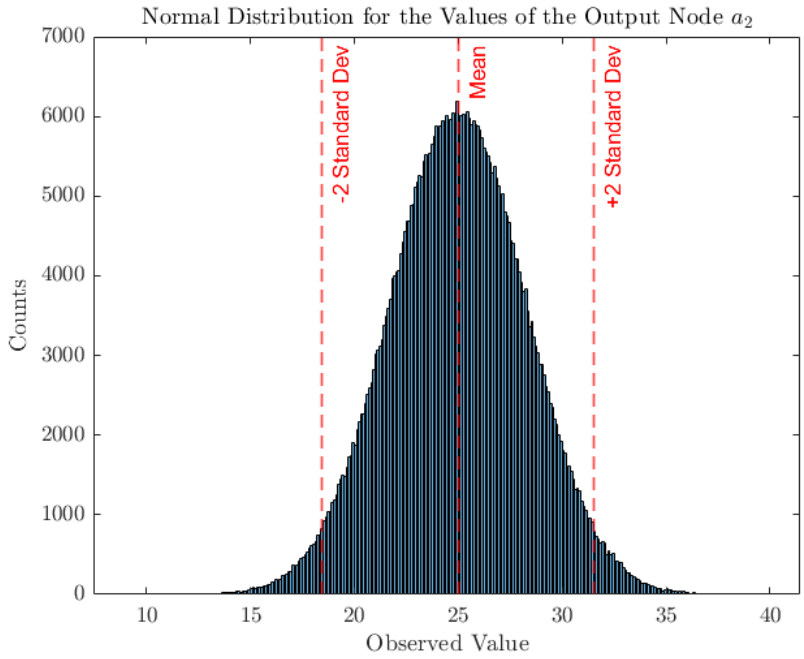


Figure 5.4: Distribution of Output Sample Values Generated for the Output Node a_2 using MC Simulation in SSM.

Figure 5.4 shows the distribution of output sample values generated for node a_2 underMC simulation for 500000 random input samples. Figure 5.4 also shows the mean value (i.e., 24.9998) generated for the output samples and 95% confidence limit (shown as red dotted line). We can see that the output sample value generated for node a_2 also follows a normal distribution similar to the input. Regarding the optimality of the algorithm, it is seen that output values lie within a confidence interval of 95% (i.e., between 18.4342 and 31.5533). Therefore it can be concluded that under the uncertain or random conditions, there are 95% chances that output generated by the algorithm will contain the actual value of node a_2. Additionally, we can also say that the algorithm will not generate any extreme values under uncertain conditions. For other output nodes (i.e., a_1), similar behavior is observed.

5-2-2 MC Coariance Validation

Similar to mean, MC simulation is also performed to validate the covariance matrix generated for the output node in network 5.1. To validate the result analytical mean obtained in Equation 5.6 through the algorithm is compared against the empirical mean for various sample sizes.

MC validation for the covariance is conducted under a similar setup compared to polynomial and bilinear validation. It is seen that around 2000000 samples, there is a convergence between the analytical and empirical covariance matrix generated. As a result validation experiment

confirm the comparable performance between covariance generated through the prediction and under a variety of uncertain knowledge. Covariance matrix in equation 5.7 shows the empirical matrix generated under 2000000 samples. The covariance matrix obtained under various sample size during MC simulation can be found in the Appendix C.

$$\Sigma_d = \begin{pmatrix} 10.75 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10.75 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \end{pmatrix} \tag{5.7}$$

The validation test conducted for output mean and covariance generated through the algorithm and various other analyses shows that the MPBUP algorithm can be used to propagate uncertainty into the system. It also works well, considering different uncertain conditions taken into account. The only thing that is needed to keep in mind is to carefully convert the SSM (i.e., dynamical model) to the standard SOP model to implement the algorithm.

5-3 Modeling of MPBUP algorithm on Four Tank System

In the previous sections and chapters, a detailed description of the MPBUP algorithm is explained in terms of modeling a system into a bilinear model, designing a bipartite graph, algorithm iteration, uncertainty propagation, and calculation of output node moments, and validation. In this section similar procedure compared to SSM is followed, but only the MPBUP algorithm is applied on the dynamical system, which is four tank system. It is an extended version of the SSM.

MPBUP algorithm finds its application in propagating uncertainty into the system to calculate the threshold bound in FDD. The algorithm is finally implemented on an actual real-time application (i.e., Four Tank System) to propagate uncertainty into the system at each time step to calculate robust probabilistic threshold bound. A similar procedure is followed, where uncertainty affecting the system is modeled into a standard bilinear model and converted into SOP format. Finally, the bipartite graph is designed, and input uncertainty (i.e., mean and variance) is propagated through the graph to calculate the mean and variance of the output node.

5-3-1 Four Tank System.

Four Tank System or quadruple tank is a standard nonlinear process. It is a bench process in control engineering to study the nonlinear effect as it is difficult to optimize the output. It consists of four tanks which are usually interconnected with each other with their input and output. It is a standard Multiple-Input and Multiple-Output (MIMO) process. The system is controlled by two inputs (i.e., pump speed). These inputs are also used to manipulate to

control the level of two tanks. Thus, the system shows interacting, multivariate dynamics as each of the pumps affects the output. The system also consists of adjustable valves to adjust the level of water in the tank [14]. The systematic diagram of the four tank system is shown in Figure 5.5.

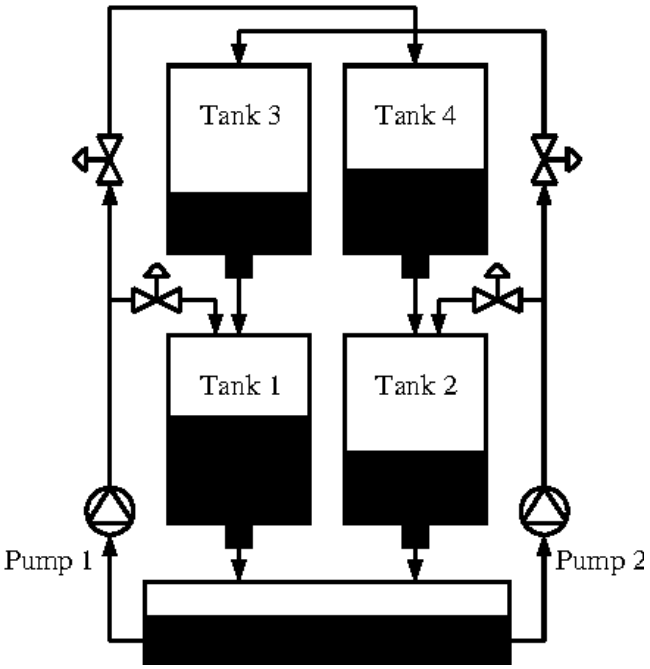


Figure 5.5: Systematic Diagram of Four Tank System [15]

Bernoulli law is used for modeling the mathematical model of the four-tank system. Therefore the dynamical model of four tanks can be expressed in Equation 5.8.

$$\begin{aligned}
\frac{dx_1}{dt} &= -\frac{a_{p_1}}{A_1}\sqrt{2gx_1} + \frac{a_{p_3}}{A_1}\sqrt{2gx_3} + \frac{\eta_1 \varsigma_1}{A_1}\nu_1 \\
\frac{dx_2}{dt} &= -\frac{a_{p_2}}{A_2}\sqrt{2gx_2} + \frac{a_{p_4}}{A_2}\sqrt{2gx_4} + \frac{\eta_2 \varsigma_2}{A_2}\nu_2 \\
\frac{dx_3}{dt} &= -\frac{a_{p_3}}{A_3}\sqrt{2gx_3} + \frac{(1-\eta_2)\varsigma_2}{A_3}\nu_2 \\
\frac{dx_4}{dt} &= -\frac{a_{p_4}}{A_4}\sqrt{2gx_4} + \frac{(1-\eta_2)\varsigma_1}{A_4}\nu_1
\end{aligned} \tag{5.8}$$

Here, Bernoulli law is used to model the flow out of the tank. x_i is the level of water in the tank i, ν_i is the manipulated input into the system. A_i is the area of tank i, and a_{p_i} is the area of the pipe flowing out of the tank i. The ratio of water diverted from one tank to another tank η_i. ς_i is the gain of the respective pump.

Equation 5.8 shows nonlinear mathematical model of four tank system. To apply the MPBUP algorithm on the system, the model is linearized around the operating point, which is the height of the tanks (i.e., x_1, x_2, x_3, x_4) [19]. Therefore the linearized model of four tank

system can be defined in the Equation 5.9 as:

$$\frac{dx}{dt} = \begin{pmatrix} -\frac{1}{T_1} & 0 & \frac{A_3}{A_1 T_3} & 0 \\ 0 & -\frac{1}{T_2} & 0 & \frac{A_4}{A_2 T_4} \\ 0 & 0 & -\frac{1}{T_3} & 0 \\ 0 & 0 & 0 & -\frac{1}{T_4} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} \frac{\eta_1 \varsigma_1}{A_1} & 0 \\ 0 & \frac{\eta_2 \varsigma_2}{A_3} \\ 0 & \frac{(1-\eta_2)\varsigma_2}{A_3} \\ \frac{(1-\eta_1)\varsigma_1}{A_4} & 0 \end{pmatrix} \begin{pmatrix} \nu_1 \\ \nu_2 \end{pmatrix} \tag{5.9}$$

$$T_i = \frac{A_i}{a_{p_i}} \sqrt{\frac{2h_i(0)}{g}}$$

System Parameter	Value	System Parameter	Value
a_{p_1}, a_{p_2}	2.3 cm^2	ς_1	5.51 $cm^3/s(\%)$
a_{p_3}, a_{p_4}	2.3 cm^2	ς_2	6.58 $cm^3/s(\%)$
A_1, A_2, A_3, A_4	730 cm^2	g	981 cm/s^2
$\nu_1(0)$, $\nu_1(0)$	60 %	η_1	0.3333
T_1	53.8 sec	η_2	0.307
T_2	48.0 sec	$x_1(0)$	14.1 cm
T_3	38.5 sec	$x_2(0)$	11.2 cm
T_4	31.1 sec	$x_3(0)$	7.2 cm
		$x_4(0)$	4.7 cm

Table 5-3.1: Model Parameters of Experimental Four Tank System.

Table 5-3.1 shows the values of various model parameters and constants, involved in four tank system.

Equation 5.9 shows the linearized model of the system. To determine uncertainty propagation into the system at each time step, the discrete model is considered. Therefore to discretize, the model Euler Approximation is taken into account. Therefore, Euler Approximation using forward difference can be described as:

$$x(k+1) = x(k) + h\dot{x}$$

Equation 5.10 shows the discretized model for four tank system. where, h is the sampling time.

$$\begin{pmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \end{pmatrix} = \begin{pmatrix} 1-\frac{h}{T_1} & 0 & \frac{A_3 h}{A_1 T_3} & 0 \\ 0 & 1-\frac{h}{T_2} & 0 & \frac{A_4 h}{A_2 T_4} \\ 0 & 0 & 1-\frac{h}{T_3} & 0 \\ 0 & 0 & 0 & 1-\frac{h}{T_4} \end{pmatrix} \begin{pmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{pmatrix} + \begin{pmatrix} \frac{\eta_1 \varsigma_1 h}{A_1} & 0 \\ 0 & \frac{\eta_2 \varsigma_2}{A_3 h} \\ 0 & \frac{(1-\eta_2)\varsigma_2 h}{A_3} \\ \frac{(1-\eta_1)\varsigma_1 h}{A_4} & 0 \end{pmatrix} \begin{pmatrix} \nu_1(k) \\ \nu_2(k) \end{pmatrix} \tag{5.10}$$

There are various ways to select the sampling time (h) to discretize the model. One of the commonly used approaches to choose a proper sampling time is by giving a step input to a continuous system, and the number of samples per rise time should be in the range of $4-10$ (i.e., $N_r = \frac{T_r}{h} \approx 4-10$).

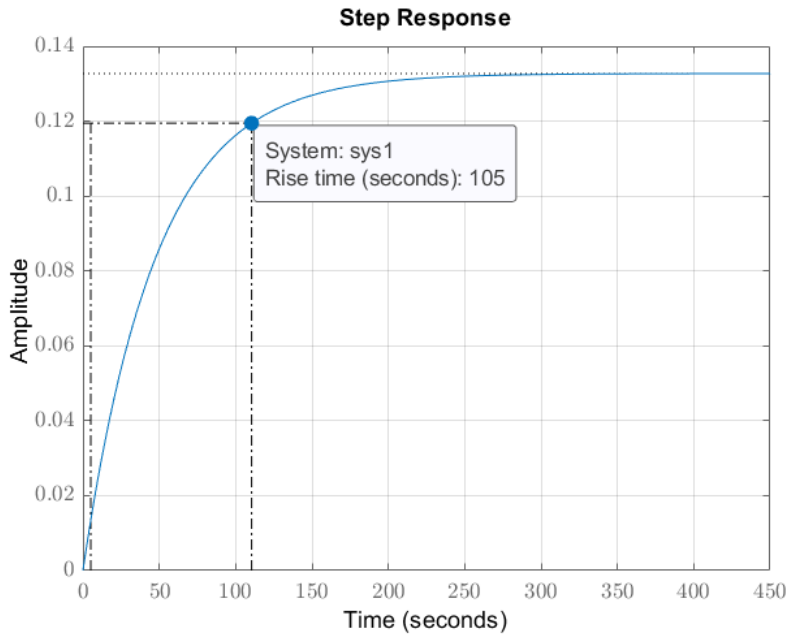


Figure 5.6: Step Response for Four Tank System.

Figure 5.6 shows the step response of the four tank system for the tank 2, when input (ν_2) is given. It is seen that the rise time for the system is around 105 seconds. As a result sampling time of 10 is used to discretize the system. The equation shows the discretized model for the four tank system at a sampling time of 10 seconds.

$$\begin{pmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \end{pmatrix} = \begin{pmatrix} 0.8034 & 0 & 0.2081 & 0 \\ 0 & 0.8118 & 0 & 0.2470 \\ 0 & 0 & 0.7710 & 0 \\ 0 & 0 & 0 & 0.7248 \end{pmatrix} \begin{pmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{pmatrix} + \begin{pmatrix} 0.0229 & 0.0070 \\ 0.0007 & 0.0250 \\ 0 & 0.0550 \\ 0.0430 & 0 \end{pmatrix} \begin{pmatrix} \nu_1(k) \\ \nu_2(k) \end{pmatrix}$$
$$\begin{pmatrix} y_1(k) \\ y_2(k) \\ y_3(k) \\ y_4(k) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{pmatrix} \tag{5.11}$$

Finally, all the prerequisites needed to implement the MPBUP algorithm are being implemented. As discussed in the beginning MPBUP algorithm is used to propagate uncertainty, and it is a part of the probabilistic-based approach for FDD. Therefore observer-based approach discussed in section 2-4-1 is used to model the system. Equation 2.4 shows the total uncertainty propagated through the system, where L is the observer gain such that $A - LC$ is Hurwitz and the system is stable. Therefore the poles of the observer gain can be defined as:

$$p = \begin{bmatrix} 0.55 & 0.65 & 0.7 & 0.8 \end{bmatrix}$$

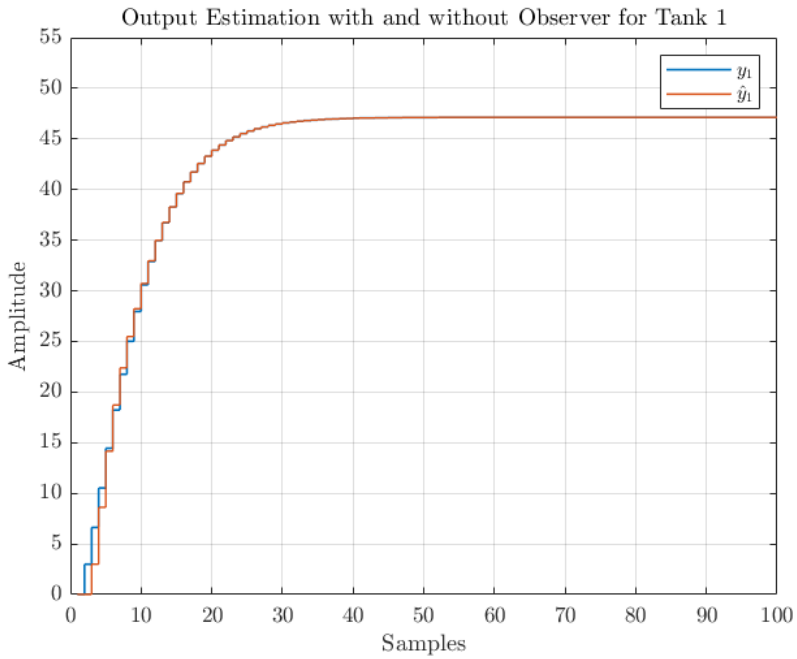


Figure 5.7: Comparison of Output Estimation With and Without Observer.

The faster observer is not considered while designing the observer as it acts as a deadbeat controller and is not numerically stable. Whereas for comparatively slower poles, the response is relatively slow. Figure 5.7 shows the comparison between the output estimation with and without an observer for Tank 1. Finally, the observer is designed using poles $(0.55, 0, 66, 0.7, 0.8)$, replicates the actual process, and gives the desired response, which can be observed in the Figure 5.7, where the observer traces the actual dynamics of the water level for Tank 1. After 20 samples, the output states of both observer and the actual process become equal, and the error becomes zero.

Therefore observer gain (L) can be defined as:

$$L = \begin{pmatrix} 0.2804 & 0 & 0 & 0 \\ 0 & 0.1618 & 0 & 0 \\ 0.2081 & 0 & 0.0710 & 0 \\ 0 & 0.2857 & 0 & 0.1683 \end{pmatrix} \tag{5.12}$$

5-3-2 Modeling of Four Tank System

Equation 2.4 represents total uncertainty affecting the system, when probabilistic model is taken into account. Therefore, uncertainty propagation for four tank can be defined as:

$$\begin{pmatrix} \gamma_1(k) \\ \gamma_2(k) \\ \gamma_3(k) \\ \gamma_4(k) \end{pmatrix} = \begin{pmatrix} 1-\frac{h}{T_1} & 0 & h\frac{A_3}{A_1T_3} & 0 \\ 0 & 1-\frac{h}{T_2} & 0 & h\frac{A_4}{A_2T_4} \\ 0 & 0 & 1-\frac{h}{T_3} & 0 \\ 0 & 0 & 0 & 1-\frac{h}{T_4} \end{pmatrix} \begin{pmatrix} r_{x_{x_1}}(k) \\ r_{x_{x_2}}(k) \\ r_{x_{x_3}}(k) \\ r_{x_{x_4}}(k) \end{pmatrix} + \begin{pmatrix} \tilde{A}_{11} & \tilde{A}_{12} & \tilde{A}_{13} & \tilde{A}_{14} \\ \tilde{A}_{21} & \tilde{A}_{22} & \tilde{A}_{23} & \tilde{A}_{24} \\ \tilde{A}_{31} & \tilde{A}_{32} & \tilde{A}_{33} & \tilde{A}_{34} \\ \tilde{A}_{41} & \tilde{A}_{42} & \tilde{A}_{43} & \tilde{A}_{44} \end{pmatrix} \begin{pmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \end{pmatrix} + \begin{pmatrix} \tilde{B}_{11} & \tilde{B}_{12} \\ \tilde{B}_{21} & \tilde{B}_{22} \\ \tilde{B}_{31} & \tilde{B}_{32} \\ \tilde{B}_{41} & \tilde{B}_{42} \end{pmatrix} \begin{pmatrix} \nu_1(k) \\ \nu_2(k) \end{pmatrix} + \begin{pmatrix} \chi(k) \\ \chi(k) \\ \chi(k) \\ \chi(k) \end{pmatrix} + \begin{pmatrix} L_1\xi(k) \\ L_2\xi(k) \\ L_3\xi(k) \\ L_4\xi(k) \end{pmatrix}$$

where, A_{ij} represents parametric uncertainty affecting states in state matrix A. B_{ij} represents parametric uncertainty affecting input in input matrix B. χ and ξ corresponds to process and measurement noise respectively. $r_{x_{x_i}}$ corresponds to residual generated between output and measurement value (i.e., $r_x = x - \hat{x}$).

On expanding the uncertainty propagation for four tank system can be written as:

$$\begin{aligned}
\gamma_{x_1}(k) &= -\frac{(1-h)r_{x_{x_1}}}{T_1} + \frac{A_3 r_{x_{x_3}} h}{A_1 T_3} + \chi + L_1\xi + \tilde{A}x_1 + \tilde{A}x_2 + \tilde{A}x_3 + \tilde{A}x_4 + \tilde{B}\nu_1 + \tilde{B}\nu_2 \\
\gamma_{x_2}(k) &= -\frac{(1-h)r_{x_{h_2}}}{T_2} + \frac{A_4 r_{x_{x_4}} h}{A_2 T_4} + \chi + L_2\xi + \tilde{A}x_1 + \tilde{A}x_2 + \tilde{A}x_3 + \tilde{A}x_4 + \tilde{B}\nu_1 + \tilde{B}\nu_2 \\
\gamma_{x_3}(k) &= -\frac{(1-h)r_{x_{x_3}}}{T_3} + \chi + L_3\xi + \tilde{A}x_1 + \tilde{A}x_2 + \tilde{A}x_3 + \tilde{A}x_4 + \tilde{B}\nu_1 + \tilde{B}\nu_2 \\
\gamma_{x_4}(k) &= -\frac{(1-h)r_{x_{x_4}}}{T_4} + \chi + L_4\xi + \tilde{A}x_1 + \tilde{A}x_2 + \tilde{A}x_3 + \tilde{A}x_4 + \tilde{B}\nu_1 + \tilde{B}\nu_2
\end{aligned} \tag{5.13}$$

To implement the MPBUP algorithm for uncertainty propagation in the dynamical model defined in Equation 5.13, it is compared with standard bilinear equation model in Equation 3.1. In comparison the input (b_i) and output (a_i) for the dynamical model to implement algorithm can be defined as:

$$b = \begin{bmatrix} \tilde{A} & \tilde{B} & x_1 & x_2 & x_3 & x_4 & r_{x_{h_1}} & r_{x_{h_2}} & r_{x_{h_3}} & r_{x_{h_4}} & v_1 & v_2 & \chi & L\xi_1 & L\xi_2 & L\xi_3 & L\xi_4 \end{bmatrix}$$
$$a = \begin{bmatrix} \gamma_{x_1} & \gamma_{x_2} & \gamma_{x_3} & \gamma_{x_4} \end{bmatrix}$$

Assumption 1: For parametric uncertainty $\tilde{A}$ and $\tilde{B}$ to reduce the number of input node and numerical complexity of the algorithm, $\tilde{A}$ and $\tilde{B}$ are assumed as input node rather than element wise (A_{ij} and B_{ij}) as input node.

Assumption 2: For $\tilde{A}$, $\tilde{B}$, χ, and ξ are stochastic variables whose distribution is unknown, but the knowledge of their mean and covariance concerning all other components is known.

Among the input nodes, Inputs ν_i are assumed to be known and deterministic (i.e., constant). For the state inputs x_1, x_2, x_3, x_4 its mean and covariance is known from previous time step. For the residual $r_{x_{x_1}}$, $r_{x_{x_2}}$, $r_{x_{x_3}}$, $r_{x_{x_4}}$ its mean and covariance generated from the previous

time step using MPBUP algorithm is taken into account. For parametric uncertainty and noise (i.e., $\tilde{A}$, $\tilde{B}$, χ and ξ) assumption is taken into account. Size of the coefficient matrix (φ) based upon the number of input ($b = 17$) and output ($a = 4$) therefore, it can be defined as $\varphi \in \mathbb{R}^{4\times 378}$, as [$n = 4$, $m = 17$, $n_f = (n+m)\times(m+1) = (17+4)\times(17+1) = 378$]. Therefore the dynamics of the model in Equation 5.13 in terms of standard bilinear with reference to Equation 3.1 model can be defined as:

$$\begin{aligned}
a_1 &= \kappa_{1,(7)}b_7 + \kappa_{1,(9)}b_9 + \kappa_{1,(13)}b_{13} + \kappa_{1,(14)}b_{14} + \omega_{1,(1,3)}b_1b_3 + \omega_{1,(1,4)}b_1b_4 + \omega_{1,(1,5)}b_1b_5 \\
&\quad + \omega_{1,(1,6)}b_1b_6 + \omega_{1,(2,11)}b_2b_{11} + \omega_{1,(2,12)}b_2b_{12} \\
a_2 &= \kappa_{2,(8)}b_8 + \kappa_{2,(10)}b_{10} + \kappa_{2,(13)}b_{13} + \kappa_{2,(15)}b_{15} + \omega_{2,(1,3)}b_1b_3 + \omega_{2,(1,4)}b_1b_4 + \omega_{2,(1,5)}b_1b_5 \\
&\quad + \omega_{2,(1,6)}b_1b_6 + \omega_{2,(2,11)}b_2b_{11} + \omega_{2,(2,12)}b_2b_{12} \\
a_3 &= \kappa_{3,(9)}b_9 + \kappa_{3(13)}b_{13} + \kappa_{3,(16)}b_{16} + \omega_{3,(1,3)}b_1b_3 + \omega_{3,(1,4)}b_1b_4 + \omega_{3,(1,5)}b_1b_5 + \omega_{3,(1,6)}b_1b_6 \\
&\quad + \omega_{3,(2,11)}b_2b_{11} + \omega_{3,(2,12)}b_2b_{12} \\
a_4 &= \kappa_{4,(10)}b_{10} + \kappa_{4(13)}b_{13} + \kappa_{4,(17)}b_{17} + \omega_{4,(1,3)}b_1b_3 + \omega_{4,(1,4)}b_1b_4 + \omega_{4,(1,5)}b_1b_5 + \omega_{4,(1,6)}b_1b_6 \\
&\quad + \omega_{4,(2,11)}b_2b_{11} + \omega_{3,(2,12)}b_2b_{12}
\end{aligned} \tag{5.14}$$

Input Node	Variable	Output Node	Variable
b_1	$\tilde{A}$	a_1	γ_1
b_2	$\tilde{B}$	a_2	γ_2
b_3	x_1	a_3	γ_3
b_4	x_2	a_4	γ_4
b_5	x_3		
b_6	x_4		
b_7	$r_{x_{x_1}}$		
b_8	$r_{x_{x_2}}$		
b_9	$r_{x_{x_3}}$		
b_{10}	$r_{x_{x_4}}$		
b_{11}	ν_1		
b_{12}	ν_2		
b_{13}	χ		
b_{14}	$L_1\xi$		
b_{15}	$L_2\xi$		
b_{16}	$L_3\xi$		
b_{17}	$L_4\xi$		

Table 5-3.2: Relation between Variables and Input-Output Nodes in Four Tank System.

Coefficients	Coefficient Matrix Elements	Constants	Coefficients	Coefficient Matrix Elements	Constants
$\kappa_{1,(7)}$	$\varphi_{(1,7)}$	$\frac{1-h}{T_1}$	$\kappa_{1,(9)}$	$\varphi_{(1,9)}$	$\frac{A_3}{A_1T_3}$
$\kappa_{1,(13)}$	$\varphi_{(1,13)}$	1	$\kappa_{1,(14)}$	$\varphi_{(1,14)}$	1
$\kappa_{2(8)}$	$\varphi_{(2,8)}$	$\frac{1-h}{T_2}$	$\kappa_{2,(10)}$	$\varphi_{(2,10)}$	$\frac{A_4}{A_2T_4}$
$\kappa_{2,(13)}$	$\varphi_{(2,13)}$	1	$\kappa_{2,(15)}$	$\varphi_{(2,15)}$	1
$\kappa_{3,(9)}$	$\varphi_{(3,9)}$	$\frac{1-h}{T_3}$	$\kappa_{3,(13)}$	$\varphi_{(3,13)}$	1
$\kappa_{3,(16)}$	$\varphi_{(3,16)}$	1	$\kappa_{4,(10)}$	$\varphi_{(4,10)}$	$\frac{1-h}{T_4}$
$\kappa_{4,(13)}$	$\varphi_{(4,13)}$	1	$\kappa_{4,(17)}$	$\varphi_{(4,17)}$	1
$\omega_{1,(1,3)}$	$\varphi_{(1,20)}$	1	$\omega_{1,(1,4)}$	$\varphi_{(1,21)}$	1
$\omega_{1,(1,5)}$	$\varphi_{(1,22)}$	1	$\omega_{1,(1,6)}$	$\varphi_{(1,23)}$	1
$\omega_{1,(2,11)}$	$\varphi_{(1,45)}$	1	$\omega_{1,(2,12)}$	$\varphi_{(1,46)}$	1
$\omega_{2,(1,3)}$	$\varphi_{(2,20)}$	1	$\omega_{2,(1,4)}$	$\varphi_{(2,21)}$	1
$\omega_{2,(1,5)}$	$\varphi_{(2,22)}$	1	$\omega_{2,(1,6)}$	$\varphi_{(2,23)}$	1
$\omega_{2,(2,11)}$	$\varphi_{(2,45)}$	1	$\omega_{2,(2,12)}$	$\varphi_{(2,46)}$	1
$\omega_{3,(1,3)}$	$\varphi_{(3,20)}$	1	$\omega_{3,(1,4)}$	$\varphi_{(3,21)}$	1
$\omega_{3,(1,5)}$	$\varphi_{(3,22)}$	1	$\omega_{3,(1,6)}$	$\varphi_{(3,23)}$	1
$\omega_{3,(2,11)}$	$\varphi_{(3,45)}$	1	$\omega_{3,(2,12)}$	$\varphi_{(3,46)}$	1
$\omega_{4,(1,3)}$	$\varphi_{(4,20)}$	1	$\omega_{4,(1,4)}$	$\varphi_{(4,21)}$	1
$\omega_{4,(1,5)}$	$\varphi_{(4,22)}$	1	$\omega_{4,(1,6)}$	$\varphi_{(4,23)}$	1
$\omega_{4,(2,11)}$	$\varphi_{(4,45)}$	1	$\omega_{4,(2,12)}$	$\varphi_{(4,46)}$	1

Table 5-3.3: Relation between the Coefficient Matrix Terms and Constants in Four Tank System.

Table 5-3.2 represents the relation between variables used in Equation 5.14 and,the various input and output nodes in the dynamical model. Table 5-3.3 represents the relationship between various Coefficients terms (i.e., κ, ψ, ω, θ), Coefficient matrix (φ) elements and various constant involved in the dynamical model of the four tank system with respect to input and output acting in the system.

Finally, now system dynamics in the bilinear form are written in terms of standard SOP format with reference to Equation 3.2. The system dynamics are converted to SOP format to draw the bipartite graph and implement the algorithm.

$$\begin{aligned}
a_1 &= \varphi_{(1,7)}f_7 + \varphi_{(1,9)}f_9 + \varphi_{(1,13)}f_{13} + \varphi_{(1,14)}f_{14} + \varphi_{(1,20)}f_{20} + \varphi_{(1,21)}f_{21} + \varphi_{(1,22)}f_{22} + \varphi_{(1,23)}f_{23} \\
&\quad + \varphi_{(1,45)}f_{45} + \varphi_{(1,46)}f_{46} \\
a_2 &= \varphi_{(2,7)}f_7 + \varphi_{(2,9)}f_9 + \varphi_{(2,13)}f_{13} + \varphi_{(2,15)}f_{15} + \varphi_{(2,20)}f_{20} + \varphi_{(2,21)}f_{21} + \varphi_{(2,22)}f_{22} + \varphi_{(2,23)}f_{23} \\
&\quad + \varphi_{(2,45)}f_{45} + \varphi_{(2,46)}f_{46} \\
a_3 &= \varphi_{(3,7)}f_7 + \varphi_{(3,9)}f_9 + \varphi_{(3,13)}f_{13} + \varphi_{(3,16)}f_{16} + \varphi_{(3,20)}f_{20} + \varphi_{(3,21)}f_{21} + \varphi_{(3,22)}f_{22} + \varphi_{(3,23)}f_{23} \\
&\quad + \varphi_{(3,45)}f_{45} + \varphi_{(3,46)}f_{46} \\
a_4 &= \varphi_{(4,7)}f_7 + \varphi_{(4,9)}f_9 + \varphi_{(4,13)}f_{13} + \varphi_{(4,17)}f_{17} + \varphi_{(4,20)}f_{20} + \varphi_{(4,21)}f_{21} + \varphi_{(4,22)}f_{22} + \varphi_{(4,23)}f_{23} \\
&\quad + \varphi_{(4,45)}f_{45} + \varphi_{(4,46)}f_{46}
\end{aligned} \tag{5.15}$$

f_i represent the relationship between the input and output node. Table 5-3.4 represents the relation between the factor used in the Equation 5.15 and, input and output nodes of the dynamical model.

Function Term	Input-Output Term	Function Term	Input-Output Term
f_7	$b_7 = r_{x_{x_1}}$	f_8	$b_8 = r_{x_{x_2}}$
f_9	$b_9 = r_{x_{x_3}}$	f_{10}	$b_{10} = r_{x_{x_4}}$
f_{14}	$b_{14} = L_1\xi$	f_{15}	$b_{15} = L_2\xi$
f_{16}	$b_{16} = L_3\xi$	f_{17}	$b_{17} = L_4\xi$
f_{13}	$b_{13} = \chi$	f_{20}	$b_1b_3 = \tilde{A}x_1$
f_{21}	$b_1b_4 = \tilde{A}x_2$	f_{22}	$b_1b_3 = \tilde{A}x_3$
f_{23}	$b_1b_3 = \tilde{A}x_4$	f_{45}	$b_2b_{11} = \tilde{B}\nu_1$
f_{46}	$b_2b_{12} = \tilde{B}\nu_2$		

Table 5-3.4: Relation between Factors and Input-Output Nodes in Four Tank System

Therefore, the procedure to convert any dynamical model to standard SOP to implement the algorithm is similar to procedure described for the $2nd$ order SSM.

5-3-3 Algorithm Implementation

Based upon the dynamics in Equation 5.15, node set can be divided into two set such as:

$$\begin{aligned}
\text{Variable Node}(\mathcal{V}) &= \{b_1, b_2, b_3, \cdots b_{17}, a_1, a_2, a_3, a_4\} \\
\text{Factor Node}(\mathcal{F}) &= \{f_7, f_8, f_9, f_{10}, f_{14}, f_{15}, f_{16}, f_{17}, f_{13}, f_{20}, f_{21}, f_{22}, f_{23}, f_{45}, f_{46}\}
\end{aligned}$$

Figure 5.8 shows the bipartite graph generated for the four tank system in MATLAB. The factor node set ($\mathcal{F}$) and variable node set ($\mathcal{V}$) are shown in red and blue colour respectively.

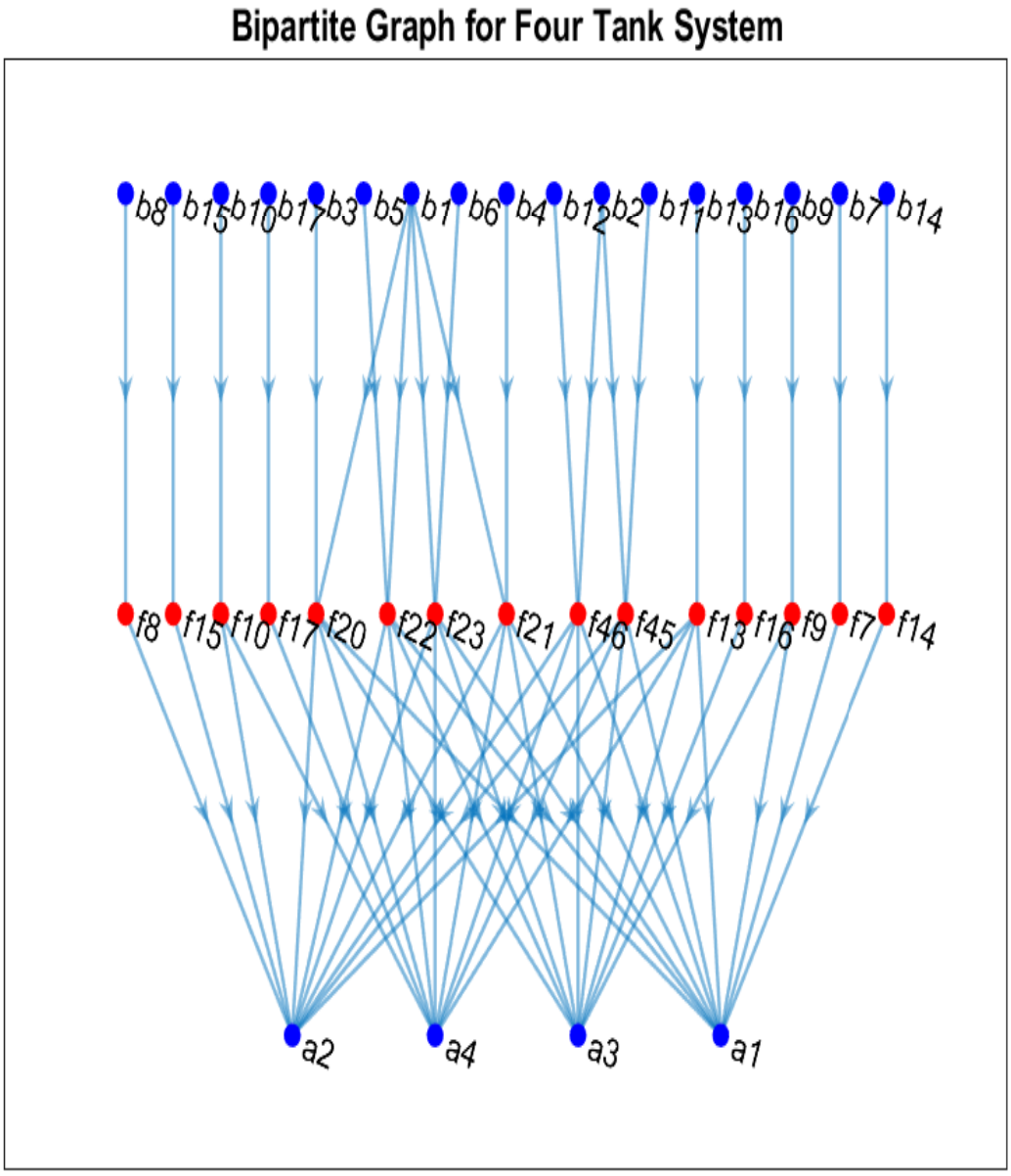


Figure 5.8: Bipartite Graph for Four Tank System.

Finally, MPBUP is algorithm is implemented on the four tank system to determine total uncertainty propagated at each time step. Algorithm propagates mean and covariance of the input node (i.e. b_1, b_2, b_3, $\cdots$ b_{17}) to determine the mean and covariance of the output node (i.e., a_1, a_2, a_3, a_4). As discussed coefficient matrix ($\varphi \in \mathbb{R}^{4\times 378}$). Nonzero elements of coefficient matrix (φ) are expressed in Table 5-3.3. Table 5-3.5 shows the mean and covariance

of the input node given in the algorithm as input.

Input Nodes	Mean	Covariance
b_1	5	0.25
b_2	5	0.5
b_3	14	0.1
b_4	11	0.2
b_5	7	0.2
b_6	4	0.7
b_7	1	0.25
b_8	1	0.25
b_9	1	0.25
b_{10}	1	0.25
b_{11}	2	0.6
b_{12}	2	0.6
b_{13}	3	0.5
b_{14}	0.3043	0.05
b_{15}	1.6840	0.05
b_{16}	-0.0297	0.05
b_{17}	0.2646	0.05

Table 5-3.5: Assumed Values of Mean and Covariance of the Input Node for Four Tank System.

At the end of the algorithm, output mean ($\bar{d}$) and covariance matrix (Σ_d) is generated for the vector (d). The size of output mean is ($\bar{d} \in \mathbb{R}^{(1\times17)}$) and of output covariance matrix is ($\Sigma_d \in \mathbb{R}^{(17\times17)}$), which are quite complicated to represent. As we are only interested in knowing mean and covariance matrix for the output node, therefore Equation 5.16 represent the mean and covariance of the output nodes (i.e.,a_1, a_2, a_3 and a_4) for the time instant $t = 1$, as output mean and covariance is calculated using the initial condition.

$$\begin{aligned} \bar{d} &= \begin{bmatrix} 204.3428 & 205.7428 & 203.7413 & 203.9498 \end{bmatrix} \\ \Sigma_d &= \begin{pmatrix} 152.8332 & 152.6000 & 152.6401 & 152.6000 \\ 152.6000 & 152.8300 & 152.6000 & 152.6448 \\ 152.6401 & 152.6000 & 152.7986 & 152.6000 \\ 152.6000 & 152.6448 & 152.6000 & 152.7813 \end{pmatrix} \end{aligned} \tag{5.16}$$

Output mean and covariance matrix obtained through the algorithm at the time instant $t = 1$ is propagated through the next time step using the evolution of the state dynamics.

5-4 Validation of Four Tank System

Similar to previous experiment, validation test is also performed on the implementation of MPBUP algorithm on four tank system to validate the output mean and covariance generated for the output nodes (i.e.,a_1, a_2, a_3, a_4) found in the equation 5.16. Compared to the previous experiment, a similar setup is used. Input given in the algorithm is a random number of particular distributions, and the corresponding output is generated. The output of various samples is used to obtain the empirical mean.

5-4-1 MC Mean Validation

MC simulation is conducted to validate the output mean obtained for output node (a_1, a_2, a_3 and a_4) for network in Figure 5.8. Analytical mean is computed through the algorithm in Equation 5.16 and is compared against the empirical mean obtained under various sample size. As discussed in the previous section, input for the four tank system (ν_i) is known and deterministic (i.e., constant). MC simulation is conducted by giving a random numbers to various input nodes except (ν_1 and ν_2). These random numbers are normally distributed with their mean and variance defined in Table 5-3.5, and the corresponding output is generated for various sample sizes.

	Sample Size	a_1	a_2	a_3	a_4
Original Mean		204.3428	205.7428	203.7413	203.9894
Empirical Mean	10	204.0325	205.4325	204.4310	203.6790
Empirical Mean	100	204.3779	205.7779	203.7763	204.0244
Empirical Mean	1000	204.3293	205.7293	203.7278	203.9759
Empirical Mean	10000	204.3392	205.7392	203.7377	203.9858
Empirical Mean	100000	204.3433	205.7433	203.7418	203.9899
Empirical Mean	500000	204.3435	205.7435	203.7420	203.9901
Empirical Mean	1000000	204.3428	205.7428	203.7413	203.9895

Table 5-4.1: Analytical and Empirical Mean Value for the Output Node a_1, a_2, a_3 and a_4 in Four Tank System.

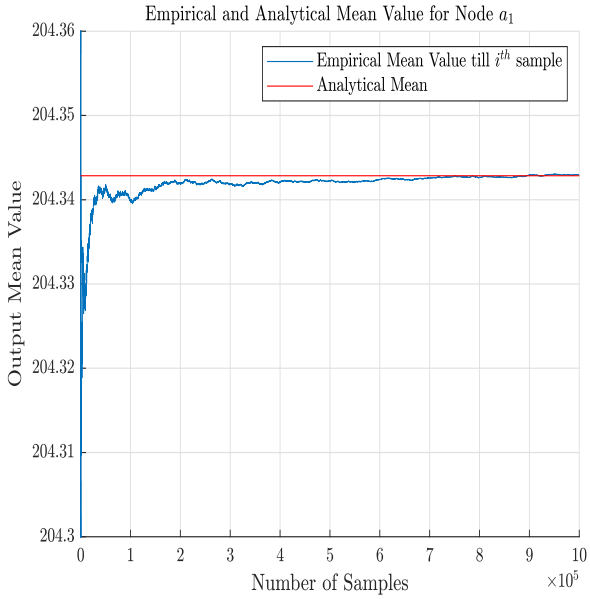
Table 5-4.1 shows the comparison between analytical and empirical mean for various sample size. It is seen there is a convergence between analytical and empirical mean as sample size increases. Around 1000000 samples both analytical and empirical mean have become identical. Therefore validation experiment show convergence between the predicted value through the algorithm and those generated under a variety of uncertain knowledge.

	Sample Size	Error (a_1)	Error ($a_2\}$	Error (a_3)	Error (a_4)
Original Mean					
Empirical Mean	10	0.3103	0.3103	-0.6869	0.3104
Empirical Mean	100	-0.0351	-0.0351	-0.0350	-0.0351
Empirical Mean	1000	0.0135	0.0135	0.0135	0.0135
Empirical Mean	10000	0.0036	0.0036	0.0036	0.0036
Empirical Mean	100000	-0.0005	-0.0005	-0.0005	-0.0005
Empirical Mean	500000	-0.0007	-0.0007	-0.0007	-0.0007
Empirical Mean	1000000	0.0000	0.0000	0.0000	0.0000

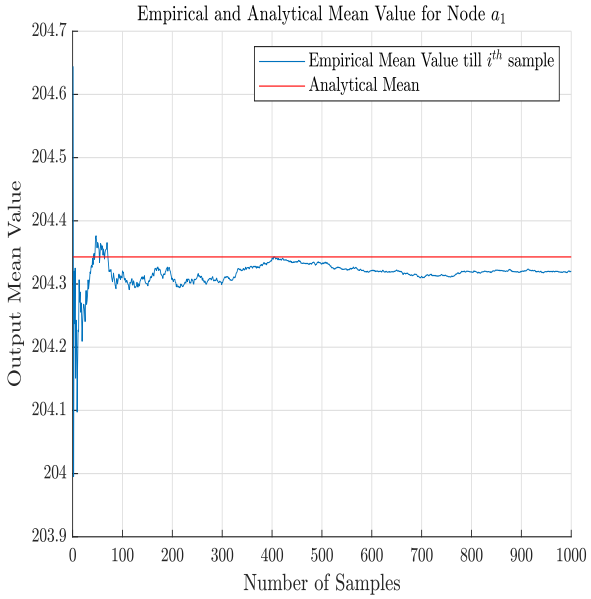
Table 5-4.2: Error between Analytical and Empirical Mean Value for Output Node a_1, a_2, a_3 and a_4 in Four Tank System.

Table 5-4.2 shows the error between the analytical and empirical mean evaluated under various sample size for the output node a_1, a_2, a_3 and a_4. We can see that around 1000000 samples

the error becomes zero indicating the convergence between analytical and empirical mean value.

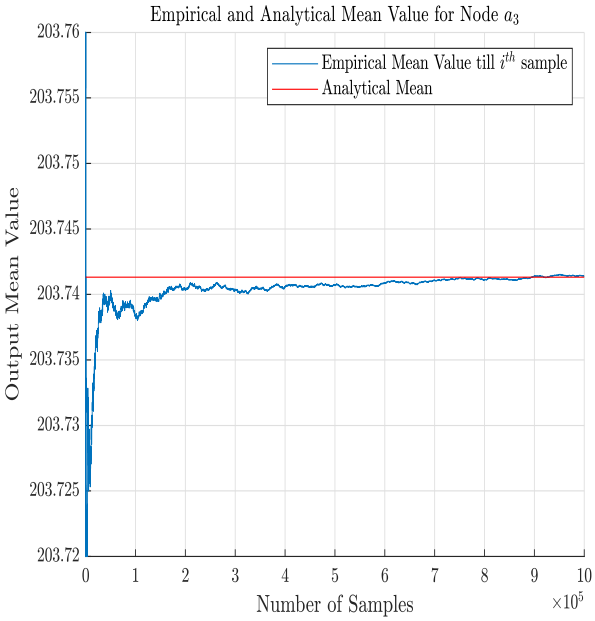


(a) Empirical and Analytical Mean Value.

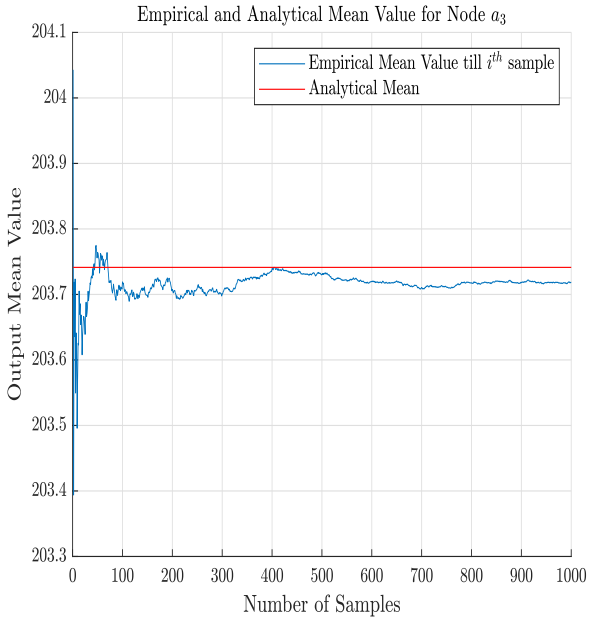


(b) Empirical and Analytical Mean Value for 1000 Samples.

Figure 5.9: Comparison between Empirical and Analytical Mean Value Propagation for Node a_1 in Four Tank System for MC Simulation.



(a) Empirical and Analytical Mean Value.



(b) Empirical and Analytical Mean Value for 1000 Samples.

Figure 5.10: Comparison between Empirical and Analytical Mean Value Propagation for Node a_3 in Four Tank System for MC Simulation.

Compared to analyses done for various other dynamical models, similar analyses is conducted for four tank system. Figures 5.9 and 5.10 depicts the comparison between the behavior of the empirical mean value generated for the output node a_1 and a_3 respectively. The empirical mean values are generated under large random input samples (shown as blue line), which are normally distributed, and the predicted analytical mean value (shown as red line). The empirical mean value for i^{th} sample is calculated by evaluating the mean value till i^{th} samples at every i^{th} instant. In comparison, the analytical mean value is the predicted value generated by the algorithm.

The Figures in the left 5.9a and 5.10a shows the magnified view along y axis. Its shows the comparison between empirical and analytical mean value propagation. We can see that around 1000000 samples, both analytical and empirical mean values converge, and error becomes negligible for output node a_1 and a_2 by LLN. Convergence confirms that analytical and empirical values become equal under various uncertain conditions acting on the system for a large sample size.

Figures in the right 5.9b, and 5.10b shows the initial distortion in the empirical mean compared to the analytical mean value for a small sample size of 1000 samples. There is always a considerable error between the analytical and empirical mean value for a small sample size. To prevent redundant results, a comparison between the analytical and empirical mean value is only provided for output nodes a_1 and a_3 respectively. A similar kind of behavior is also observed for the output nodes a_2 and a_4. Figures 5.9 and 5.10 are used to support the result obtained in the Tables 5-4.1 and 5-4.2 and to validate the output mean value generated through the MPBUP algorithm by seeing the convergence between analytical and empirical mean value.

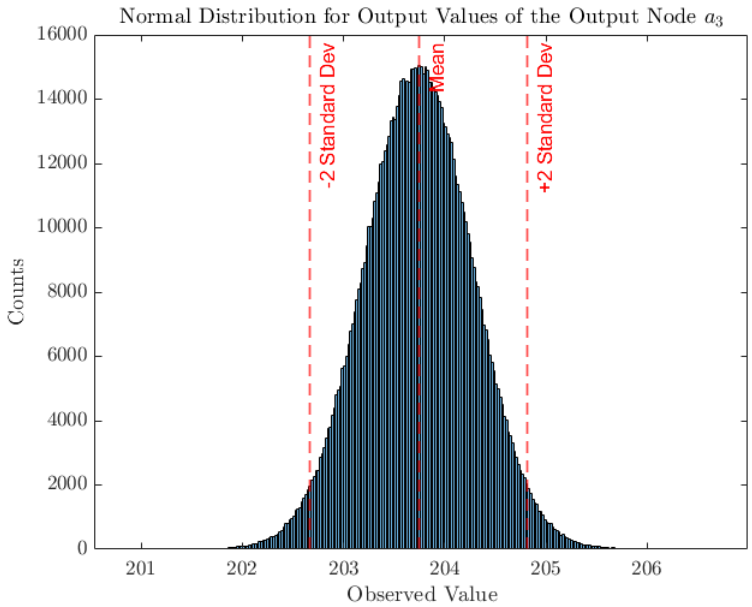


Figure 5.11: Distribution of Output Sample Values Generated for the Output Node a_3 using MC Simulation in Four Tank System.

Figure 5.11 shows the distribution of output sample values generated for node a_2 under

MC simulation for 1000000 random input samples. Figure 5.11 also shows the mean value (i.e., 203.7414) generated for the output samples and 95% confidence limit (using red dotted line). We can see that the output sample value generated for node a_1 also follows a normal distribution similar to the input. Regarding the optimality of the algorithm, it is seen that output values lie within a confidence interval of 95% (i.e., between 202.6698 and 205.8129). Therefore it can be concluded that under the uncertain or random conditions, there are 95% chances that output generated by the algorithm will contain the actual value of node a_2. Additionally, we can also say that the algorithm will not generate any extreme values under uncertain conditions. For other output node (i.e., a_1, a_3 and a_4) a similar behavior is observed.

5-4-2 MC Covariance Validation

Similar to mean, MC simulation is also used to validate the covariance matrix generated for the output nodes in Figure 5.8. A similar setup is used to validate the output covariance matrix generated through the algorithm. The analytical covariance matrix generated through the algorithm is compared against the empirical mean generated for various sample sizes under random input numbers is shown in Equation 5.17.

$$\Sigma_d = \begin{pmatrix} 152.8332 & 152.6000 & 152.6401 & 152.6000 \\ 152.6000 & 152.8300 & 152.6000 & 152.6448 \\ 152.6401 & 152.6000 & 152.7986 & 152.6000 \\ 152.6000 & 152.6448 & 152.6000 & 152.7813 \end{pmatrix} \tag{5.17}$$

In the validation experiment, it is seen around 1000000 samples of the random number; there is a convergence between the analytical and empirical output covariance matrix generated through the algorithm, thus showing a comparable performance of the MPBUP algorithm. The covariance matrix obtained under various sample size during MC simulation can be found in the Appendix C.

Through various analyses conducted, it can be concluded that the MPBUP algorithm can model a complicated dynamical model for the uncertainty quantification at each time step and can be further used to propagate it to the next time step ahead. Moreover, the algorithm also works well, considering various uncertain conditions acting on the system.

5-5 Threshold bounding through MPBUP Algorithm

In the previous section, how to implement the MPBUP algorithm on the basic network, four tank system, etc., is discussed for uncertainty propagation. One of the MPBUP algorithm applications is to determine robust probabilistic threshold in the FDD sector, where uncertainty propagation is considered. In Section 2-4-1 probabilistic based approach for FDD is discussed. In a probabilistic-based approach, an estimator is designed that replicates the process's actual dynamics, and a difference between actual and predicted output value is found called residual. Residual is used to determine the robust threshold taking all the parametric uncertainty, process noise, and measurement noise into account. With reference to Section 2-4-1, the residual equation can be defined as:

$$r_x(k+1) = LCr_x(k) + \gamma(k)$$
$$r_y(k) = \bar{C}r_x(k) + \xi(k)$$

γ corresponds to total uncertainty propagated through the network and ξ corresponds to process noise. It can be seen r_y is a measurable quantity and it can used for the fault detection. r_y can be used to define a robust threshold (ε_α) as:

$$\mathcal{E}_\alpha \triangleq \left\{ r_y \in \mathbb{R}^n \mid \sqrt{(r_y - \bar{r}_y)\, \Sigma_{r_y}^{-1}\, (r_y - \bar{r}_y)} \leq \frac{n}{\alpha} \right\}, \alpha \in (0, 1]$$

Therefore to compute robust threshold we need to know about the mean ($\bar{r}_y$) and the covariance (Σ_{r_y}) of r_y. At time instant $k = 0$ the residual r_y is a deterministic quantity and it can be computed using $r_x(0)$ whereas at the future instant $r_y(k+1)$ is a stochastic quantity as it depends random variable $r_x(k)$, $\gamma(k)$ and $\xi(k)$. A detailed explanation of the residual computation is explained in the subsequent section.

5-5-1 Probabilistic Threshold Computation

As discussed in previous section $r_y(k)$ can be used to compute robust threshold at each time step in a dynamical system. To compute the threshold we need to know mean ($\bar{r}_y$) and covariance (Σ_{r_y}) at each time step. At $k = 0$, r_y is deterministic quantity which depends upon $r_x(0)$ (In order to purse that we assume the knowledge of initial mean and covariance $\bar{r}_x(0) \triangleq \mathbb{E}[r_x(0)]$ and $\Sigma_{r_x} \triangleq \mathrm{Cov}[r_x(0)]$ is known respectively), whereas in order to calculate for next time instant we need to iteratively propagate $r_x(k+1)$ which in turn depend upon the total uncertainty (γ_k). This will require the computation of mean ($\bar{\gamma}(k)$) and covariance ($\Sigma_\gamma(k)$) which in turn depends upon the mean and covariance of the individual uncertainty sources. Therefore to compute the overall uncertainty propagation in the model at each time instant MPBUP algorithm can be used.

To analyze the computation of threshold in probabilistic model a detailed explanation is as follow. For time instant ($k = 0$) we define a vector $\varepsilon(0) \triangleq \mathrm{col}(r_x(0), \gamma(0))$. While it is unknown and not measurable we can compute its mean and covariance as :

$$\begin{aligned} \bar{\varepsilon}(0) &\triangleq \mathbb{E}[\varepsilon(0)] = \mathrm{col}(\bar{r}_x(0), \bar{\gamma}(0)) \\ \Sigma_\varepsilon(0) &\triangleq \mathrm{diag}(\Sigma_{r_x}(0), \Sigma_\gamma(0)) \end{aligned}$$

One advantage of using $\varepsilon(0)$ is r_x and γ are independent as uncertainty had no chance to influence the state dynamics yet. A block diagonal matrix $A_\varepsilon \triangleq \mathrm{diag}(A_0, I)$ is used. Therefore we can simply write equation for propagation in time as:

$$\begin{aligned} \bar{\varepsilon}(k+1) &= A_\varepsilon \bar{\varepsilon}(k) \\ \Sigma_\varepsilon(k+1) &= A_\varepsilon \Sigma_\varepsilon A_\varepsilon^T \end{aligned} \tag{5.18}$$

Using equation 5.18 at start index $k = 0$ it is possible to compute mean and covariance of r_x at time $k + 1 = 1$ and residual $r_y(1)$ can be obtained, which allow to determine set based threshold $\varepsilon_\alpha(k+1)$. In order to proceed with next time instant we need to compute $\bar{\gamma}(1)$ and $\Sigma_\gamma(1)$ which in turn depend upon the uncertainty propagation and where our MPBUP algorithm will come handy.

5-5-2 Threshold Bounding for Four Tank System

In the previous section, a detailed explanation is given on bounding the threshold under the probabilistic conditions. Then, a similar approach is followed to determine the probabilistic threshold for the four tank system using the MPBUP algorithm under uncertainty propagation.

For time instant $k = 0$, vector ε is used to find mean and covariance for residual $r_x(1)$, which is used to determine mean and covariance of residual $r_y(1)$ to find the robust threshold. From next time instant for $k = 1$, stochastic uncertainty also influences the state dynamics. As a result, γ also has to be taken into account. To propagate uncertainty into the system MPBUP algorithm is used to determine the mean and covariance of uncertainty propagation at various time steps.

To determine the threshold following initial conditions were assumed for the four tank system:-

The initial level of the tanks (x) can be described:

$$x = \begin{bmatrix} 14 & 11 & 7 & 4 \end{bmatrix}$$

The level of the tanks are described in terms of cm. The input given into the tank can be described as:

$$\nu = \begin{bmatrix} 50 & 50 \end{bmatrix}$$

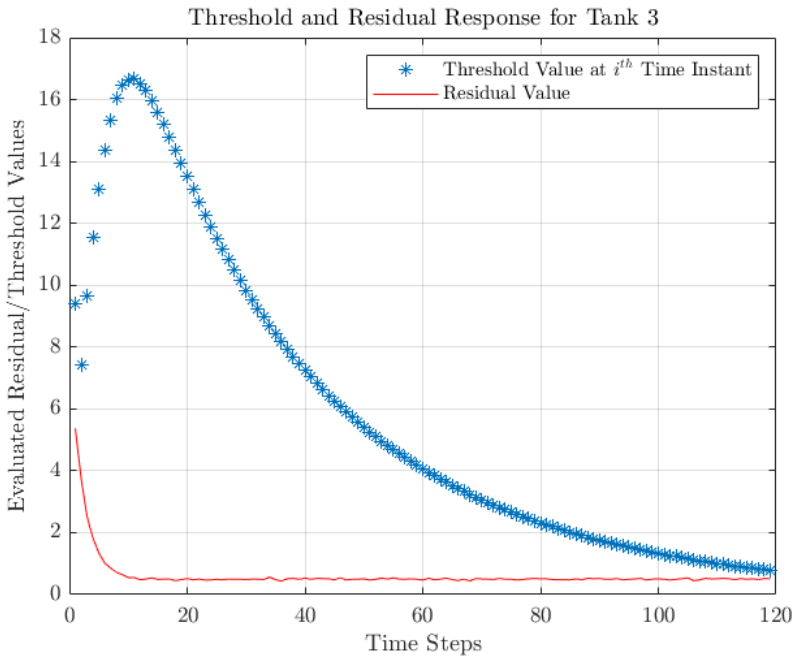


Figure 5.12: Comparison between Threshold and Residual Value Generated for Tank 3.

Figure 5.12 shows the threshold and residual value generated for the Tank 3 in the four tank system from the time instant $k = 1$. Uncertainty present in the system is propagated through

the MPBUP algorithm, which is eventually used to find the threshold value at each step. From Figure 5.12 we can see that the residual value is always lower than the threshold value as the system is in healthy condition. We can also see that the threshold value found using the MPBUP algorithm for uncertainty propagation and Mahalabohnis distance has a large gap compared to the residual initially, but the gap between the residual and threshold decreases as time progresses. Mahalanobis distance takes output residual, its mean, and covariance into account. The output residual is calculated based upon the error generated between the states of the output estimator and the actual process and uncertainty present in the model at each time step. Initially, at the start, there is a large error present between the states of the output estimator and the actual process. As the system propagates further, the error decreases. As a result threshold value starts moving toward residual based upon the dynamics of the model.

The figure also shows that the threshold value generated through the algorithm is dynamic (i.e., as the process is propagating through the time step, the threshold value is also getting updated based upon the evolution of state dynamics and various uncertainty acting on the system). It is also not conservative and static like the deterministic-based approach leading to low FAR. The threshold bound found above in Figure 5.12 taking various parametric uncertainties in the model, process noise, and measurement noise into consideration at each time. Thus the threshold value founding using the probabilistic based approach and MPBUP algorithm is able to bound arbitrary shaped residual.

Therefore, the MPBUP algorithm can successfully propagate various uncertainties into the system, further used to determine the robust threshold. A similar kind of behavior graph can be seen for the other tanks. (Note: Threshold and residual values calculated are not exact as many assumptions were considered in model parameters, parametric uncertainties, etc. The primary purpose of the graph is to show how uncertainty can be propagated using the MPBUP algorithm to find a robust threshold value at each time instant and showing the comparison between residual and threshold).

Chapter 6

MPBUP Algorithm Analysis

In the previous chapters, a detailed discussion is given about the MPBUP algorithm, its implementation, validation, and how to apply the algorithm on real-time system to find a robust threshold. Before we conclude the algorithm, a few more aspects are discussed in the following chapter.

6-1 Analysis of the MPBUP Algorithm

MPBUP algorithm is a novel algorithm that is designed and implemented in Chapter 3. The algorithm is further analyzed in terms of time taken to reach convergence between output value generated through the MPBUP algorithm and empirical mean value generated through MC simulation. Time taken for n number of simulation but at the same time also increasing the graph complexity in terms of factors and variables, and variety of input data types that can be implemented into the algorithm for uncertainty propagation.

Analysis 1:

A simulation is carried out to benchmark the time taken for the convergence between the analytical and empirical mean value (i.e., mean error between analytical and empirical value becomes zero) for the basic network in (Figure 4.1), 2nd order SSM in (Figure 5.1) and four tank system (Figure 5.8) under the similar conditions.

No of Nodes	Basic Network	2nd order SSM	Four Tank System
Factor Nodes	2	4	15
Variable Nodes	3	6	21
Total Nodes	5	10	36

Table 6-1.1: Total Number of Nodes in Various graph

Table 6-1.1 shows the number of factor and variable nodes present in various network graphs. Table 6-1.2 shows the comparison between the time taken to reach convergence by various network graphs for n sample size under similar conditions such as input data type, distribution of the data, etc. In Table 6-1.2 simulation time evaluated under various samples are shown in Minutes.

No of Samples	Basic Network	$2nd$ order SSM	Four Tank System
1	0.4437	1.0262	1.1605
10	3.9710	4.0915	8.7187
100	5.5300	6.4565	18.3353
1000	23.4661	29.6416	137.4520
10000	161.6160	269.3302	910.4156
100000	1340.0000	2373.2000	9227.8000
1000000	15575.0000	30008.0000	130140.0000
2000000	-	58040.0000	-

Table 6-1.2: Simulation Time for Comparison for n samples.

We can see through the Table 6-1.2 as the graph complexity increases, the time for the convergence between the analytical and empirical also increases drastically. In the table, we can see that in around 1000000 samples, the mean error in the basic network and four tank system become zero, whereas for the SSM the mean error becomes zero around 2000000 samples. An important aspect that can be analyzed for the large samples is that the increase in the simulation time is proportional to the total number of nodes. For example, at 1000000 random input samples normally distributed, the simulation time for the basic network consisting of 5 nodes is 15575 secs. On the other hand, in the SSM simulation time is 30008 secs consisting of 10 nodes, which is twice the number of nodes compared to the basic network, and simulation time is also twice the time compared to the basic network. Similarly, for the four-tank system, when compared to the basic network and SSM, a similar trend is observed.

Figures 6.1 and 6.2 shows the comparison between the basic network, $2nd$ order SSM and the four tank system based upon the three criteria number of nodes, simulation time, and the number of random samples. From the bar graph in Figures 6.1 and 6.2 we can conclude that firstly, simulation time increases linearly with the increases in the number of samples, which is quite expected behavior. Secondly, we can also conclude in terms of numerical complexity; we can see that as the number of the nodes increases, the run time for the simulation also increases at a constant sample range. Therefore we can conclude that the run time increases linearly based on the number of nodes present in the bipartite graph. In Figures 6.1 and 6.2 number of nodes 5, 10 and 36 indicates total number of nodes present in the basic network, SSM and four tank system respectively.

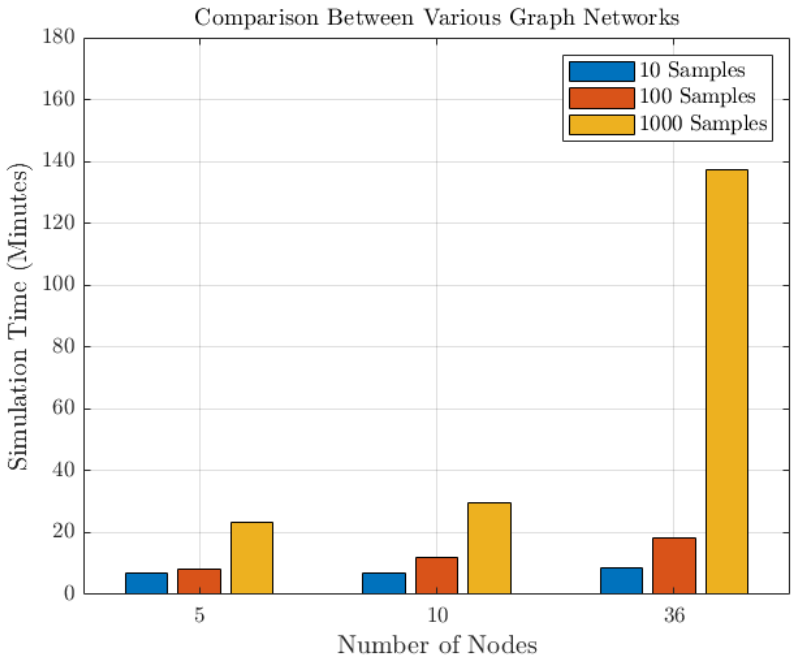


Figure 6.1: Comparison between Number of Nodes, Simulation Time and Number of Samples in Various Graph Networks.

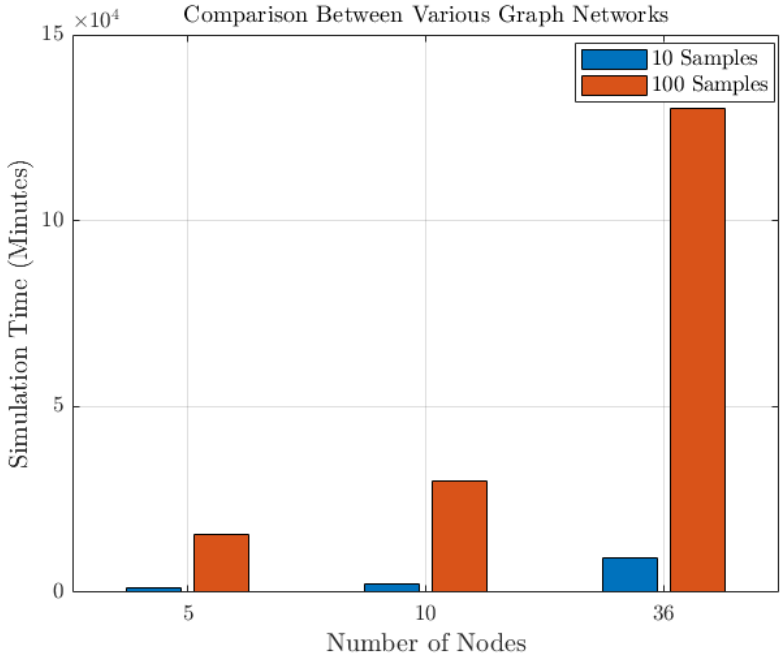


Figure 6.2: Comparison between Number of Nodes, Simulation Time and Number of Samples in various Graph Networks.

Analysis 2 :

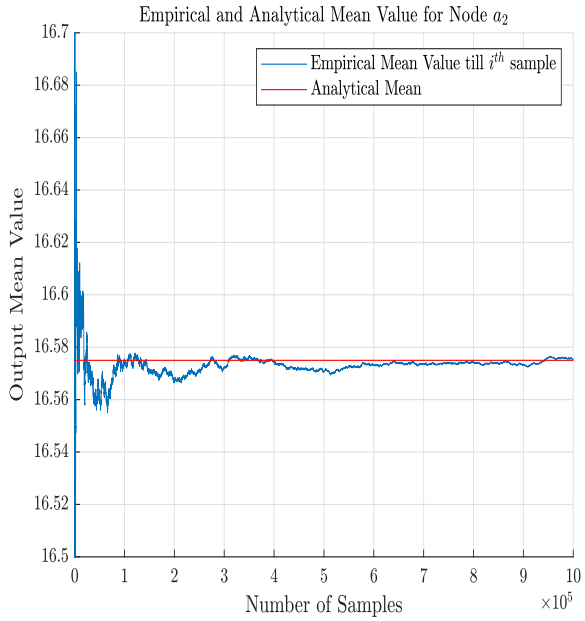
In the previous section, the MPBUP algorithm is implemented, and a validation test is performed using random input numbers, which usually are distributed. In this section, a similar type of validation test is performed using other kinds of distribution data such as Poisson Distribution and Uniform Distribution to analyze whether the MPBUP works under different kinds of input distribution data also.

MC validation test is now performed under Poisson Distribution and Uniform Distribution for the Basic Network in the Figure 4.1. The test is performed to validate the output mean generated through the MPBUP algorithm. A similar kind of setting is used compared to the previous validation experiment, where random inputs are given to the input node (i.e., b_1 for the network in Figure 4.1). In this experiment, random inputs are in Poisson Distribution or Uniformly Distributed.

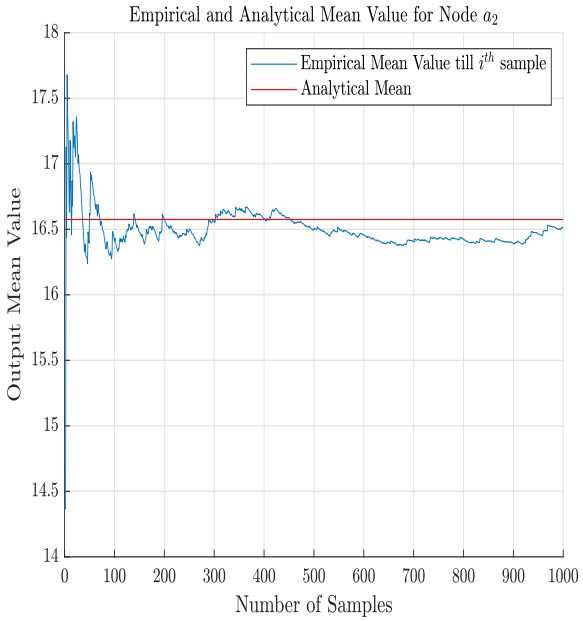
Table 6-1.3 shows the comparison between the analytical mean and empirical mean generated for output node (a_1 and a_2) for various sample size under Poisson distribution. The table 6-1.3 also shows the mean error propagated between analytical and empirical mean value for the output node (a_1 and a_2) for various sample sizes. Around 1000000 random sample, there is a convergence between the analytical and empirical mean. It is also seen that the mean error between analytical and empirical mean becomes zero at convergence. The validation experiment shows the comparable performance of the algorithm under the Poisson distribution.

	No of Samples	a_1	a_2	Error (a_1)	Error (a_2)
Analytical Mean		3.2500	16.5750		
Empirical Mean	10	3.3150	16.8513	0.0650	0.2763
Empirical Mean	100	3.2565	17.0446	0.0065	0.4696
Empirical Mean	1000	3.2513	16.6866	0.0013	0.1116
Empirical Mean	10000	3.2270	16.5424	-0.0230	-0.0326
Empirical Mean	100000	3.2496	16.5861	-0.0004	0.0111
Empirical Mean	500000	3.2502	16.5762	0.0002	0.0012
Empirical Mean	1000000	3.2500	16.5751	0.0000	0.0001

Table 6-1.3: The Analytical and Empirical Mean Generated for Node a_1 and a_2 under Poisson Distribution



(a) Empirical and Analytical Mean Value



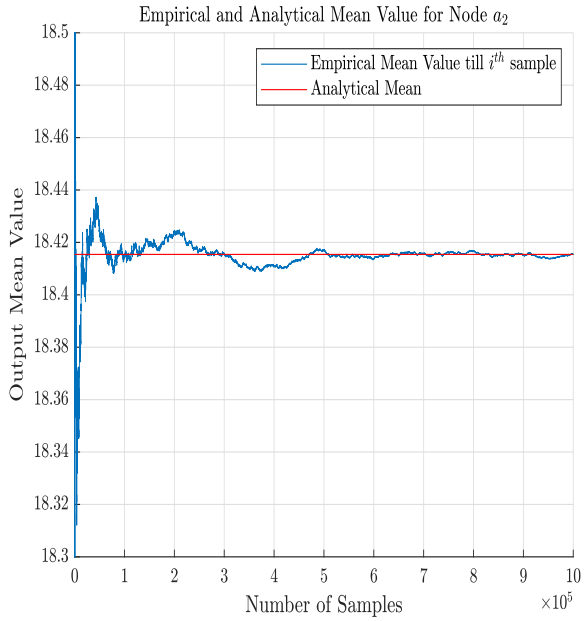
(b) Empirical and Analytical Mean Value for 1000 Samples

Figure 6.3: Comparison between Empirical and Analytical Mean Value Propagation for Node a_2 in Basic Network for MC Simulation under Poisson Distribution.

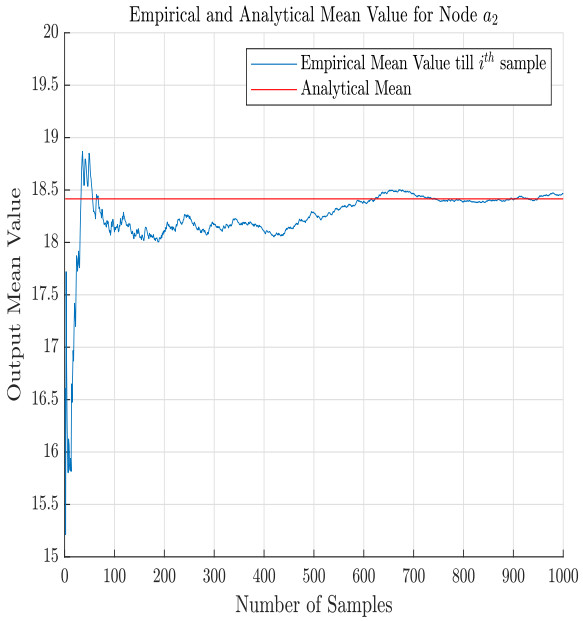
Figure 6.3 shows the comparison between the behavior of the empirical mean value generated for the output node a_2 under random input of sample size 1000000 (shown as blue line) and the predicted analytical mean value (shown as red line). We can see both analytical and empirical mean values converge, and error becomes negligible. The Figure in the left 6.3a shows the magnified view along the y axis of the empirical mean value propagation compared to the analytical mean. We can see that in Figure 6.3a both analytical and empirical values converge to each other for a large sample size. Figure in the right 6.3b shows the initial distortion between analytical and empirical value for a small sample size of 1000 samples.

Similar to Poisson distribution, MC validation is also conducted for the MPBUP algorithm using a uniform distribution. Thus, the inputs given to the input nodes are uniformly distributed. Even under the uniform distribution, a convergence is seen between analytical mean and empirical mean generated around 5000000 samples (To avoid repetitiveness, only error propagated through Network is shown for Uniform Distribution).

Figure 6.4 shows the comparison between the empirical mean value (shown as blue line) and analytical mean value (shown as red line). We can see that at 1000000 samples error between analytical and empirical value becomes significantly less. (To avoid the problem of scaling in the graph, mean values till 1000000 samples are only shown). The Figure in the left 6.4a shows the empirical mean value propagation compared to the analytical mean. Figure in the right 6.4b shows the initial distortion between analytical and empirical value for a small sample size.



(a) Empirical and Analytical Mean Value.



(b) Empirical and Analytical Mean Value for 1000 Samples.

Figure 6.4: Comparison between Empirical and Analytical Mean Value Propagation for Node a_2 in Basic Network for MC Simulation under Uniform Distribution.

Therefore, through this analysis, it can be concluded that the output generated through the MPBUP algorithm shows the comparison between the predicted value and empirical values with other kinds of distribution also. As a result, we can conclude that the performance of the MPBUP algorithm is not affected irrespective of the distribution of the input nodes.

6-1-1 Advantages and Disadvantages

Based on various experimentation, analysis, and application of the MPBUP algorithm in threshold detection, the advantages and disadvantages of the algorithm can be summarized.

Advantages:

- MPBUP algorithm helps propagate various stochastic uncertainty affecting the system into the next step to determine a robust threshold ultimately leading to low FAR.
- Use of bipartite graph in the algorithm avoids the risk of using algebraic loops in the system.
- Uncertainty in the graph is propagated through stochastic moments, which can easily be propagated through the algorithm at each time step.
- Calculation of stochastic moment of the various interconnected random variable without a proper model is difficult to generalize, whereas in uncertainty propagation model where one component γ_i is solved at a time by proceeding step by step through the graph.

- Uncertainty affecting the model is comparatively easy to visualize, hence it can be used to detect the root cause of the process.
- It is a dynamic and online process. As a result, the corrective measure can be taken instantly.
- Threshold found using the algorithm is dynamic and changes the bound value with respect to change in the time. As a result, threshold values take into account the evolution of the state dynamic and various uncertainties at each time step.
- Algorithm can also be used to propagate complex probability distribution of input variable into the system. (such as exponential, Poisson, etc.)
- Compared to other algorithms where also mean and covariance is evaluated, it is difficult to generalize its matrix and vector terms calculation involving mean and covariance. The use of a bipartite graph solves the mean and covariance propagation problem by proceeding step by step.

Disadvantages:

- Complete Knowledge of the model is required, including all the parameters in the model, the relation between various parameters to construct a bipartite graph for the system.
- Complexity of the graph increases as the number of nodes in the graph increases, leading to the rise in the algorithm's run time, which can be further analyzed through the Figure 6.1 and 6.2 respectively. Therefore we can conclude that the numerical complexity of the algorithm is $O(n)$ based upon the number of nodes.
- Initialization of the algorithm is complicated in terms of converting the dynamics of the system to the class of bilinear equation and further to standard SOP form to implement the algorithm.
- Using the Multiplicative property to convert the polynomial system to the bilinear system is tedious and time-consuming. Therefore, an alternative approach should be explored.

6-2 Comparison with Other Algorithm

During the literature study, various other algorithms were also studied, which are used to evaluate the threshold for the FDD [1], [18]. In this section, a comparative study is conducted among the various algorithms (including the Probabilistic based approach using the MPBUP algorithm), which can be used to evaluate robust threshold-based. The comparison among the various algorithm is based upon parameters or aspects which plays an important role in selecting an FDD algorithm. Table 6-2.1 shows the comparison among various algorithm based upon various criteria.

Criteria	Traditional Deterministic Based	Filters	Polynomial Chaos	Bayesian Network	Belief Propagation	Probabilistic MPBUP Based
Type of Technique	Observer Based	Observer Based	Mathematical Based	Graphical Based	Graphical Based	Observer and Graphical Based
Mathematical Model	Precise Model	KF: Precise Model. EKF and UKF: Approximate Model	Input - Output Data Set	Input-Output Data Set	Precise Model	Precise Model
Model Parameter	Exactly Known	Eactly Known	Unknown	Exactly Known	Known	Known and Time Varying
Linear/Non Linear	Linear	KF: Linear EKF and UKF : Both	Both	Both	Both	Both
Process/ Measurement Noise	Gaussian	Gaussian	Gaussian, Exponential, Polynomial	Gaussian and Non Gaussian	Gaussian and Non Gaussian	Gaussian and Non Gaussian
Threshold	Limit Based	Limit Based	Limit Based	Set Based	Set Based	Set Based
Robustness	Not Robust	Not Robust	Robust	Robust	Robust	Robust
Uncertainty	Not Taken	Taken	Taken	Taken	Taken	Taken
Uncertainty Propagation Complexity	-	Difficult	Numerically Difficult	Easy	Easy	Easy
Prior Healthy Data Set	Not Needed	Needed	Not Needed	Needed	Needed	Not Needed
Computational Complexity	High	KF: High EKF: High UKF : Low	High	Low	Low	Depends on Model Size
Offline/Online	Both	Both	Offline	Offline	Offline	Both
Fault Handling Capacity	One Fault	Multiple Fault	Multiple Fault	Multiple Fault	Multiple Faults	Multiple Fault
Fault Isolation	No	KF: No EKF: No UKF: Yes	Yes	No	No	Yes but difficult
Unknown Fault	No	Yes	Yes	Yes	Yes	Yes
Parameter Necessary	All Variables defining system dynamics	EKF: Jacobian and Hessian Matrix. UKF : Sigma Points	Collocation points and Histogram Partition	Training Data Set	Initial Prior Probability of all varaibles	Exact Input Output Relation, Moments of Input Variable

Table 6-2.1: Comparison Among Various Fault Detection Algorithms

6-3 Application of Algorithm

MPBUP algorithm is developed to propagate uncertainty into the system at each time step. During the thesis, the main application where the MPBUP algorithm is applied in a dynamical model to determine the robust threshold for the fault detection. It is used to quantify the fault at each step as part of a probabilistic-based approach, but the application of the MPBUP

algorithm is not just limited to fault detection. The main principle of the algorithm is to quantify uncertainty, which has been a significant challenge in various fields for the past many years. As uncertainties can be present in any model or system in terms of noisy experimental data, uncertainty is present in the model due to uncertain parameters, model error, boundary conditions, and numerical complexity such as rounding off, discretization, bug error, etc. Due to the following, there is always a reason need for an algorithm for uncertainty quantification [38]. Some of the applications where the MPBUP algorithm can be applied in comparison to other algorithms for uncertainty propagation are:

- It can be used in weather prediction by the meteorological department in terms of various input parameters evolved in the model such as atmospheric density, the velocity of the wind, temperature pressure, thermal conductivity, specific heat capacity of the air, etc. Other than input parameters, there can be errors or bias in the measurement sensor, etc. In the MPBUP algorithm weather prediction model can be modeled as a bipartite graph, and various uncertain parameters can be modeled to see their overall effect [33].
- MPBUP algorithm also finds its application in pharmaceutical industries as testing various medicine or vaccines are often too dangerous or expensive for human tests or large scale population. Uncertainty algorithm can be implemented on test subjects rather than full populations, taking various constraints into account such as age, side effect, etc. into consideration [25].
- Uncertainty quantification algorithm also finds its application in nuclear engineering as harsh radioactive, thermal, and chemical environment. As in nuclear reactor, core limits the measurement performance, nondestructive evaluation, and safety regulation, and various uncertain conditions cannot be taken into account [4].

Chapter 7

Conclusion

7-1 Conclusion

Uncertainty propagation and quantification have been a significant challenge in my many fields of areas. One such challenge is in fault detection and diagnosis, determining a robust threshold bound taking various uncertainty affecting the dynamical system into account. Unfortunately, there is limited knowledge available in terms of different stochastic uncertainty affecting the system, propagate the overall effect of various uncertainty affecting the next time. Many other algorithms have been developed in the past for the propagation of uncertainty. However, most developed algorithms either have high numerical complexity or fail to incorporate the loops in the dynamical system. Therefore, during this study, a new algorithm (Message Passing Bilinear Uncertainty Propagation) is proposed for the uncertainty propagation.

MPBUP is a graphical-based approach inspired by Belief Propagation. In the algorithm, various uncertainty affecting a system is modeled as a bipartite graph. The uncertainties are iteratively propagated from various input nodes in the graph to output nodes to derive their overall effect at each time step. In the algorithm, uncertainty through each node is propagated in terms of mean and covariance. Bohrnstedt and Goldberger's formula were used to find mean and covariance for the product of the random numbers. During my research, the algorithm was successfully developed and tested in MATLAB on various types of bipartite graphs to get a better understanding of the algorithm and also to refine the algorithm. After the algorithm was developed, a validation test was conducted to evaluate algorithm performance for output mean and covariance generated using Monte Carlo Simulation. Through MC simulation, a random input following normal distribution was given as an input into the system, and corresponding output was generated, which were used to derive an empirical mean. Comparison between both analytical and empirical means was conducted to validate the algorithm. Other than that, various other analyses were also conducted in terms of mean error propagation, output data distribution, etc.

Initially, the MPBUP algorithm was restricted to a bilinear system as Bohrstedt and Goldberger's formula for mean and covariance can only be used for bilinear terms. The use of the

bipartite graph reduces the problem related to taking the loop into account. Therefore as a part of my research, I proposed an approach to extend the MPBUP algorithm for the polynomial system using a multiplicative property, where polynomial terms were converted to the bilinear terms. Then the algorithm can be successfully applied. This approach was successfully implemented on the polynomial graph network, and the MC validation was performed to validate the output mean and covariance matrix generated.

After designing the algorithm, the main challenge is to apply the algorithm to a dynamical model (i.e., SSM). A systematic approach has been defined as a part of my research. A detailed procedure is given on converting the SSM into standard SOP expression to define the coefficient matrix, input, outputs and draw the bipartite graph, which are the main parts of the algorithm's initialization. Finally, the MPBUP algorithm is implemented, and a validation test was performed for the output mean and the covariance matrix. The algorithm is extended for the actual real-time system (i.e., Four Tank System). A similar procedure was followed for the state-space model, and output means and covariance matrix were generated. MC simulation was performed to validate the result. The main challenge is applying the algorithm and iteratively propagating the algorithm to find a robust threshold bound for the probabilistic-based approach. The algorithm was implemented as a part of a probabilistic-based technique to propagate the uncertainty at each time iteratively and a healthy robust threshold bound is found. Threshold found using MPBUP algorithm into account behaved as expected. In the healthy condition, it bounded the residual generated for the output states. Additionally, it is not conservative like deterministic bounds, as the threshold value is dynamic. It changes based upon the evolution of the states and propagation of uncertainty at each step.

Finally, the various analysis of the algorithm was carried out in terms of simulation time, complexity, type of input data that can be applied into the algorithm and comparative study was conducted with respect to other algorithms. During the whole process, various constraints or bottlenecks found in the algorithm were also discussed, such as precise mathematical model is needed, complexity in of terms graph design, etc., are also discussed. In conclusion, I have summarized answers to the research question formulated in Chapter 1

7-2 Future Work

The current research is focused on designing the framework and validating whether the algorithm can be used to propagate uncertainty and whether a robust threshold bound can be found using the algorithm. However, the comparison between the various algorithm was only based on theoretical analysis. Thus, future work could focus on applying the algorithm to real industrial applications and benchmarking it with other algorithms in terms of numerical complexity, run time, robust threshold bound, fault detection, etc., to validate the algorithm further.

Additionally, filters are also used for FDD. Extended Kalman filter (EKF) is used to propagate uncertainty into the system at each time step. To propagate to uncertainty at each time step covariance matrix is required, which can be generated using the MPBUP algorithm. Hence a study can be done on how to integrate EKF with the MPBUP algorithm. Finally, this study discussed how to apply the MPBUP algorithm to a polynomial system, and a validation experiment is also performed. Research can be conducted on applying the MPBUP algorithm

using Multiplicative property on a real-time system and further validating the approach. Multiplicative property is used for converting the polynomial system to a bilinear system. However, using this process can bit tedious and time-consuming, so other areas should be explored on implementing the MPBUP algorithm on the polynomial dynamical model.

Appendix A

Monte Carlo Simulation

Monte Carlo Simulation is a mathematical technique, which is used to estimate the possible outcomes of an uncertain event. It involves a computational algorithm which rely on the random sampling events to obtain the numerical results. The main principle of MC simulation is use randomness to solve problem that may be deterministic in principle.

MC simulation finds its large application in understanding the impact of risk, uncertainty in prediction and forecasting the model. MC experimentation are virtually used in variety of domain such as, In engineering domain it is used to validate an any research output by performing the same experimentation under the variety of uncertain conditions. It is also used in finance sector to evaluate the investment in the project in terms of business unit, corporate levels, etc. It is used to model the project schedules to determine overall outcome of the model under the various uncertain events including the best and the worst case scenarios. It is also used for portfolio evaluation, Option Analysis, etc [34].

A-1 Working

Working of the MC simulation can be summarised by building the model of the system. Model replicates the dynamics of the system. All the possible factors (i.e. input, external disturbance) which influence the dynamics of the system are inherited by uncertain random number which follows a possible distribution. It then recalculates the result over and over, each time using different sets of random values. Process is repeated thousands and thousands of times to produce a large number of possible outcomes. Following the principle of large number all the possible outcomes can be approximated by taking the empirical mean (i.e. Mean value found by summing all the possible outcomes generated using the random number). The empirical mean generated through the algorithm is compared against the analytical mean to validate the results [23]. Therefore MC Simulation are used for long term prediction due to their accuracy. As the number of input sample increases the range of output sample also grows, allowing to project outcome further in time with more accuracy.

Other than validation various other analysis can be drawn out such for the MC simulation such as:

- Based on the distribution of output data from MC simulation through randomly generated input, various analysis can be made in terms of the spread of the data around the analytical value in terms of taking best and worst case into consideration [28].
- Mean error and standard deviation of each sample can be calculated, comparing against the analytical for accessing the robustness of the parametric inference under various condition. It also helps to investigate the complete range of risk involved with each risky input variable.
- It can also used to access the statistical distribution of the output data generated to characterize the output variation.

A-2 Advantages and Disadvantages

The main advantages and disadvantages of using MC Simulation can be summarised as

Advantages

- It is an useful mathematical tool used for analyzing uncertain scenarios and providing probabilistic analysis under variety of different situation.
- Provides a satisfactory approximate result to computationally expensive mathematical problem.
- It can be used for both deterministic and stochastic problem.
- It is difficult to model the differ combination of values for different input values to see the effect of different scenario. MC simulation helps in understanding the which input have which value together certain outcome [17].

Disadvantages

- It is time consuming to generate a large number of samples to get desired output.
- Results obtained through MC simulation are only approximation of true solution, not the exact solution.

Appendix B

Mean and Covariance of Product of Random Variable

B-1 The Variance of a Product

x and y be jointly distributed random variables with Expectation $\mathrm{E}[x]$ and $\mathrm{E}[y]$, Variance $\mathrm{V}(x)$ and $\mathrm{V}(y)$ and Covariance $C(x, y)$ respectively. Exact variance of the product xy can be found as:-

By definition of variance for the product xy it can be described as

$$\mathrm{V}(xy) = \mathrm{E}[xy - \mathrm{E}(xy)]^2 \tag{B.1}$$

Let $\Delta x = x - \mathrm{E}(x)$ and $\Delta y = y - \mathrm{E}(y)$

$$\begin{aligned} xy &= [\Delta x + \mathrm{E}(x)][\Delta y + \mathrm{E}(y)] \\ &= (\Delta x)(\Delta y) + (\Delta x)\,\mathrm{E}(y) + (\Delta y)\,\mathrm{E}(x) + \mathrm{E}(x)\,\mathrm{E}(y) \end{aligned} \tag{B.2}$$

Taking Expectation of equation B.2

$$\begin{aligned} \mathrm{E}(xy) &= \mathrm{E}[(\Delta x)(\Delta y)] + \mathrm{E}(x)\,\mathrm{E}(y) \\ &= \mathrm{C}(x, y) + \mathrm{E}(x)\,\mathrm{E}(y) \end{aligned} \tag{B.3}$$

so, subtracting equation B.2 and B.3, we get:

$$xy - \mathrm{E}(xy) = (\Delta x)(\Delta y) + (\Delta x)\,\mathrm{E}(y) + (\Delta y)\,\mathrm{E}(x) - \mathrm{C}(x, y) \tag{B.4}$$

square and taking expectation, we get:

$$\begin{aligned} \mathrm{V}(xy) = \mathrm{E}^2(x)\,\mathrm{V}(y) + \mathrm{E}^2(y)\,\mathrm{V}(x) + \mathrm{E}[(\Delta x)^2(\Delta y)^2] + 2\,\mathrm{E}(x)\,\mathrm{E}[(\Delta \mathrm{x})(\Delta \mathrm{y})^2] + 2\,\mathrm{E}(y)\,\mathrm{E}[(\Delta \mathrm{x})^2(\Delta \mathrm{y})] + \\ 2\,\mathrm{E}(x)\,\mathrm{E}(y)C(x, y) - C^2(x, y) \end{aligned} \tag{B.5}$$

where, $\mathrm{E}(\Delta y)^2 = \mathrm{V}(y)$ and $\mathrm{E}(\Delta x)^2 = \mathrm{V}(x)$. The main assumption for variance of the product xy in equation B.5 is $\mathrm{E}(x)$ and $\mathrm{E}(y)$ are non-zero. This sollution even hold if $\mathrm{E}(x) = 0$ and $\mathrm{E}(y) = 0$.

If x and y are bivariate normally distributed and third moments vanish and $\mathrm{E}[(\Delta x)^2(\Delta y)^2] = \mathrm{V}(x)\,\mathrm{V}(y) + 2\,\mathrm{C}^2(x, y)$, Therefore B.5 can be rewritten as

$$\mathrm{V}(xy) = \mathrm{E}^2(x)\,\mathrm{V}(y) + \mathrm{E}^2(y)\,\mathrm{V}(x) + 2\,\mathrm{E}(x)\,\mathrm{E}(y)\,\mathrm{C}(x, y) + \mathrm{V}(x)\,\mathrm{V}(y) + \mathrm{C}^2(x, y) \tag{B.6}$$

If x and y are uncorrelated such that $C(x, y) = 0$, then the equation B.5 can be rewritten as

$$\mathrm{V}(xy) = \mathrm{E}^2(x)\,\mathrm{V}(y) + \mathrm{E}^2(y)\,\mathrm{V}(x) + E\left[(\Delta x)^2(\Delta y)^2\right] + 2E(x)E\left[(\Delta x)(\Delta y)^2\right] + 2E(y)E\left[(\Delta x)^2(\Delta y)\right]. \tag{B.7}$$

B-2 The Covariance of Product

Let x, y, u and v are jointly distributed random variable. The two products xy and uv by definition their covariance is described as

$$\mathrm{C}(xy, uv) = \mathrm{E}[xy - \mathrm{E}(xy)][uv - \mathrm{E}(uv)] \tag{B.8}$$

let $\Delta x = x - \mathrm{E}(x)$, $\Delta y = y - \mathrm{E}(y)$, $\Delta u = u - \mathrm{E}(u)$, $\Delta v = v - \mathrm{E}(v)$. Multiply the expression in equation B-1by the corresponding expression for $uv = \mathrm{E}(uv)$ and take the expectation. Typical terms in product include $(\Delta x)(\Delta u)\,\mathrm{E}(y)\,\mathrm{E}(v)$ whose expectation is $\mathrm{C}(x, u)\,\mathrm{E}(y)\,\mathrm{E}(v)$ and $(\Delta x)\,\mathrm{E}(y)\,\mathrm{C}(u, v)$, whose expectation is 0.

$$\begin{aligned} C(xy, uv) =& E(x)E(u)C(y, v) + E(x)E(v)C(y, u) + E(y)E(u)C(x, v) + +E(y)E(v)C(x, u) \\ &+ E[(\Delta x)(\Delta y)(\Delta u)(\Delta v)] + E(x)E[(\Delta y)(\Delta u)(\Delta v)] + E(y)E[(\Delta x)(\Delta u)(\Delta v)] \\ &+ E(u)E[(\Delta x)(\Delta y)(\Delta v)] + E(v)E[(\Delta x)(\Delta y)(\Delta u)] - C(x, y)C(u, v) \end{aligned} \tag{B.9}$$

If we get $x = u$ and $y = v$ then equation B.9 reduces to B.5, such that $\mathrm{C}(xy, xy) = \mathrm{V}(xy)$.

If we set $u = 1$ such that $\mathrm{E}(u) = 1$ and $\Delta u = 0$, Therefore equation B.9 can be rewritten as

$$\mathrm{C}(xy, v) = \mathrm{E}(x)\,\mathrm{C}(y, v) + \mathrm{E}(y)\,\mathrm{C}(x, v) + \mathrm{E}[(\Delta x)(\Delta y)(\Delta v)] \tag{B.10}$$

Under the multivariate condition all the third moment vanishes, while $\mathrm{E}[(\Delta x)(\Delta y)(\Delta u)(\Delta v)] = \mathrm{C}(x, y)\,\mathrm{C}(u, v) + \mathrm{C}(x, u)\,\mathrm{C}(y, v) + \mathrm{C}(x, v)\,\mathrm{C}(y, u)$, Therefore equation B.9 reduces to

$$\begin{aligned} \mathrm{C}(xy, uv) =& \mathrm{E}(x)\,\mathrm{E}(u)\,\mathrm{C}(y, v) + \mathrm{E}(x)\,\mathrm{E}(v)\,\mathrm{C}(y, u) + \mathrm{E}(y)\,\mathrm{E}(u)\,\mathrm{C}(x, v) + \mathrm{E}(y)\,\mathrm{E}(v)\,\mathrm{C}(x, u) \\ &+ \mathrm{C}(x, u)\,\mathrm{C}(y, v) + \mathrm{C}(x, v)\,\mathrm{C}(y, u) \end{aligned} \tag{B.11}$$

Appendix C

MC Simulation Covariance Validation

MC simulation is conducted to validate the output mean and covariance matrix generated through the MPBUP algorithm for the various models that were studied and analyzed during the study. In this section, the empirical mean for covariance matrix is generated under various sample sizes by the MPBUP algorithm for the different network graphs are discussed. The setup to conduct the validation experiment for the output covariance matrix is already discussed in the previous sections.

C-1 Basic Network- MC Covariance validation

With reference to the section 4-3-2. The analytical covariance matrix and the empirical covariance matrix generated under various sample sizes is discussed.

Analytical Mean:

$$\Sigma_d = \begin{pmatrix} 0.1056 & 0.7631 & 0.1625 & 0.1625 & 1.0563 \\ 0.7631 & 1.0567 & 1.1741 & 1.1741 & 1.0567 \\ 0.1625 & 1.1741 & 0.2500 & 0.2500 & 1.6250 \\ 0.1625 & 1.1741 & 0.2500 & 0.2500 & 1.6250 \\ 1.0563 & 1.0567 & 1.6250 & 1.6250 & 1.4625 \end{pmatrix}$$

Empirical Mean:

$$\Sigma_d(10) = \begin{pmatrix} 0.1056 & 0.7870 & 0.1625 & 0.1625 & 1.0892 \\ 0.7870 & 1.0567 & 1.2107 & 1.2107 & 1.0567 \\ 0.1625 & 1.2107 & 0.2500 & 0.2500 & 1.6757 \\ 0.1625 & 1.2107 & 0.2500 & 0.2500 & 1.6257 \\ 1.0892 & 1.0567 & 1.6257 & 1.6257 & 1.4625 \end{pmatrix}$$

$$\Sigma_d(100) = \begin{pmatrix} 0.1056 & 0.7662 & 0.1625 & 0.1625 & 1.0605 \\ 0.7662 & 1.0567 & 1.2107 & 1.2107 & 1.0567 \\ 0.1625 & 1.2107 & 0.2500 & 0.2500 & 1.6315 \\ 0.1625 & 1.2107 & 0.2500 & 0.2500 & 1.6315 \\ 1.0605 & 1.0567 & 1.6315 & 1.6315 & 1.4625 \end{pmatrix}$$

$$\Sigma_d(1000) = \begin{pmatrix} 0.1056 & 0.7615 & 0.1625 & 0.1625 & 1.0540 \\ 0.7615 & 1.0567 & 1.1716 & 1.1716 & 1.0567 \\ 0.1625 & 1.1716 & 0.2500 & 0.2500 & 1.6216 \\ 0.1625 & 1.1716 & 0.2500 & 0.2500 & 1.6216 \\ 1.0540 & 1.0567 & 1.6216 & 1.6216 & 1.4625 \end{pmatrix}$$

$$\Sigma_d(10000) = \begin{pmatrix} 0.1056 & 0.7633 & 0.1625 & 0.1625 & 1.0565 \\ 0.7633 & 1.0567 & 1.1744 & 1.1744 & 1.0567 \\ 0.1625 & 1.1744 & 0.2500 & 0.2500 & 1.6254 \\ 0.1625 & 1.1744 & 0.2500 & 0.2500 & 1.6254 \\ 1.0565 & 1.0567 & 1.6254 & 1.6254 & 1.4625 \end{pmatrix}$$

$$\Sigma_d(100000) = \begin{pmatrix} 0.1056 & 0.7631 & 0.1625 & 0.1625 & 1.0563 \\ 0.7631 & 1.0567 & 1.1741 & 1.1741 & 1.0567 \\ 0.1625 & 1.1741 & 0.2500 & 0.2500 & 1.6250 \\ 0.1625 & 1.1741 & 0.2500 & 0.2500 & 1.6250 \\ 1.0563 & 1.0567 & 1.6216 & 1.6216 & 1.4625 \end{pmatrix}$$

$$\Sigma_d(1000000) = \begin{pmatrix} 0.1056 & 0.7631 & 0.1625 & 0.1625 & 1.0563 \\ 0.7631 & 1.0567 & 1.1741 & 1.1741 & 1.0567 \\ 0.1625 & 1.1741 & 0.2500 & 0.2500 & 1.6250 \\ 0.1625 & 1.1741 & 0.2500 & 0.2500 & 1.6250 \\ 1.0563 & 1.0567 & 1.6250 & 1.6250 & 1.4625 \end{pmatrix}$$

Around 1000000 samples convergence is seen between the empirical and analytical covariance matrix values.

C-2 State Space Network- MC Covariance validation

With reference to the section 5-2-2. The analytical covariance matrix and the empirical covariance matrix generated under various sample sizes is discussed.

Analytical Mean:

$$\Sigma_d = \begin{pmatrix} 10.75 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10.75 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \end{pmatrix}$$

Empirical Mean:

$$\Sigma_d(10) = \begin{pmatrix} 10.2588 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5.9438 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \end{pmatrix}$$

$$\Sigma_d(100) = \begin{pmatrix} 10.6158 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10.2223 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \end{pmatrix}$$

$$\Sigma_d(1000) = \begin{pmatrix} 10.2957 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10.2924 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \end{pmatrix}$$

$$\Sigma_d(10000) = \begin{pmatrix} 10.7833 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10.6866 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \end{pmatrix}$$

$$\Sigma_d(100000) = \begin{pmatrix} 10.7689 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10.7509 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \end{pmatrix}$$

$$\Sigma_d(1000000) = \begin{pmatrix} 10.7423 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10.7363 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \end{pmatrix}$$

$$\Sigma_d(2000000) = \begin{pmatrix} 10.7500 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10.7500 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 & 0.5 \end{pmatrix}$$

Around 2000000 samples convergence is seen between the empirical and analytical covariance matrix values.

C-3 Four Tank System - MC Covariance Validation

With reference to the section 5-4-2. The analytical covariance matrix and the empirical covariance matrix generated under various sample sizes is discussed. Analytical:

$$\Sigma_d = \begin{pmatrix} 152.8332 & 152.6000 & 152.6401 & 152.6000 \\ 152.6000 & 152.8300 & 152.6000 & 152.6448 \\ 152.6401 & 152.6000 & 152.7986 & 152.6000 \\ 152.6000 & 152.6448 & 152.6000 & 152.7813 \end{pmatrix}$$

Empirical:

$$\Sigma_d(10) = \begin{pmatrix} 152.3654 & 152.1322 & 152.1723 & 152.1322 \\ 152.1322 & 152.3622 & 152.3122 & 152.1770 \\ 152.1723 & 152.1322 & 152.3308 & 152.1322 \\ 152.1322 & 152.1770 & 152.1322 & 152.3136 \end{pmatrix}$$

$$\Sigma_d(100) = \begin{pmatrix} 152.8702 & 152.6370 & 152.6771 & 152.6370 \\ 152.6370 & 152.8670 & 152.6320 & 152.6817 \\ 152.6771 & 152.6370 & 152.8356 & 152.6370 \\ 152.6370 & 152.6817 & 152.6370 & 152.8183 \end{pmatrix}$$

$$\Sigma_d(1000) = \begin{pmatrix} 152.9560 & 152.7228 & 152.7629 & 152.7228 \\ 152.7228 & 152.9528 & 152.7228 & 152.7676 \\ 152.7629 & 152.7228 & 152.9214 & 152.7228 \\ 152.7228 & 152.7676 & 152.7228 & 152.9041 \end{pmatrix}$$

$$\Sigma_d(10000) = \begin{pmatrix} 152.7528 & 152.5196 & 152.5597 & 152.5196 \\ 152.5196 & 152.7496 & 152.5156 & 152.5643 \\ 152.5597 & 152.5196 & 152.7182 & 152.5196 \\ 152.5196 & 152.5643 & 152.5196 & 152.7009 \end{pmatrix}$$

$$\Sigma_d(100000) = \begin{pmatrix} 152.8778 & 152.6446 & 152.6847 & 152.6446 \\ 152.6446 & 152.8796 & 152.6446 & 152.6893 \\ 152.6847 & 152.6446 & 152.8432 & 152.6446 \\ 152.6446 & 152.6893 & 152.6446 & 152.8259 \end{pmatrix}$$

$$\Sigma_d(1000000) = \begin{pmatrix} 152.8332 & 152.6000 & 152.6401 & 152.6000 \\ 152.6000 & 152.8300 & 152.6000 & 152.6448 \\ 152.6401 & 152.6000 & 152.7986 & 152.6000 \\ 152.6000 & 152.6448 & 152.6000 & 152.7813 \end{pmatrix}$$

Around 1000000 samples convergence is seen between the empirical and analytical covariance matrix values.

Bibliography

[1] L. An and N. Sepehri. Hydraulic actuator circuit fault detection using extended kalman filter. *Proceedings of the 2003 American Control Conference, 2003.*

[2] M. Blanke, M. Kinnaert, J. Lunze, and M. Staroswiecki. *Diagnosis and Fault-Tolerant Control*. Springer, 2016.

[3] G. W. Bohrnstedt and A. S. Goldberger. On the exact covariance of products of random variables. *Journal of the American Statistical Association*, 64(328):1439–1442, 1969.

[4] L L Briggs and Nuclear Engineering Division. Uncertainty quantification approaches for advanced reactor analyses. 3 2009.

[5] H. Budman, Y. Du, and T.A. Duever. Generalized polynomial chaos-based fault detection and classification for nonlinear dynamic processes. *Industrial amp; Engineering Chemistry Research,*, 55:2069–2082, 2016.

[6] B. Cai, L. Huang, and M. Xie. Bayesian networks in fault diagnosis. *IEEE Transactions on Industrial Informatics*, 13(5):2227–2240, 2017.

[7] J. Chen and S. Nielsen. Model-based methods for fault diagnosis. *Transactions of The Institute of Measurement and Control - TRANS INST MEASURE CONTROL*, 17:73–83, 04 1995.

[8] J. Chen and R.J. Patton. *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Kluwer Academic, 1999.

[9] X. Chen. A new generalization of chebyshev inequality for random vectors. 07 2007.

[10] S. Dey and J.A. Stori. A bayesian network approach to root cause diagnosis of process variations. *International Journal of Machine Tools and Manufacture*, 45(1):75–91, 2005.

[11] S. Ding. *Model-Based Fault Diagnosis Techniques: Design Schemes, Algorithms and Tools*. Springer, 2015.

[12] R. Ferrari, H. Dibowski, and S. Baldi. A message passing algorithm for automatic

synthesis of probabilistic fault detectors from building automation ontologies. *IFAC-PapersOnLine*, 50(1):4184–4190, 2017.

[13] P.M. Frank. Enhancement of robustness in observer-based fault detection. *International Journal of Control*, 59(4):955–981, 1994.

[14] E. P. Gatzke, E. S. Meadows, C. Wang, and F. J. Doyle. Model based control of a four-tank system. *Computers and amp; Chemical Engineering*, 24(2-7):1503–1509, 2000.

[15] N. Hajji, A. Benamor, S. Maraoui, and K. Bouzrara. Centralized sliding mode control with input delay of complex system: Application to a four tank system. *2017 International Conference on Control, Automation and Diagnosis (ICCAD)*, pages 082–086, 2017.

[16] R. Isermann. *Fault-Diagnosis Systems: An Introduction From Fault Detection to Fault Tolerance*. Springer, 2011.

[17] A.M. Johansen. Monte carlo methods. In P. Peterson, E. Baker, and B. McGaw, editors, *International Encyclopedia of Education (Third Edition)*, pages 296–303. Elsevier, Oxford, third edition edition, 2010.

[18] S.J. Julier and J.K. Uhlmann. New extension of the kalman filter to nonlinear systems. *Signal Processing, Sensor Fusion, and Target Recognition VI*, 1997.

[19] B.A. Kumar, R. Jeyabharathi, S. Surendhar, S. Senthilrani, and S. Gayathri. Control of four tank system using model predictive controller. In *2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN)*, pages 1–5, 2019.

[20] W. Li, H. Li, S. Gu, and T. Chen. Process fault diagnosis with model- and knowledge-based approaches: Advances and opportunities. *Control Engineering Practice*, 105:104637, 2020.

[21] R. Liang, F. Liu, and J. Liu. A belief network reasoning framework for fault localization in communication networks. *Sensors*, 20(23):6950, 2020.

[22] H.B. Liu, C. Jiang, and Z. Xiao. Efficient uncertainty propagation for parameterized p-box using sparse-decomposition-based polynomial chaos expansion. *Mechanical Systems and Signal Processing*, 138, 2020.

[23] Jiming Liu and M.C. Desmarais. A method of learning implication networks from empirical data: Algorithm and monte-carlo simulation-based validation. *IEEE Transactions on Knowledge and Data Engineering*, 9(6):990–1004, 1997.

[24] R.De Maesschalck, D.J. Rimbaud, and D.L. Massart. The mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50(1):1–18, 2000.

[25] L. H. Mervin, S. Johansson, E. Semenova, K. A. Giblin, and O. Engkvist. Uncertainty quantification in drug design. *Drug Discovery Today*, 26(2):474–489, 2021.

[26] A. Mesbah, S. Streif, R. Findeisen, and R. D. Braatz. Active fault diagnosis for nonlinear systems with probabilistic uncertainties. *IFAC Proceedings Volumes*, 47(3):7079–7084, 2014.

[27] D. Miljković. Fault detection methods: A literature survey. In *2011 Proceedings of the 34th International Convention MIPRO*, pages 750–755, 2011.

[28] C.Z. Mooney. *Monte Carlo simulation.* Sage, 116 edition, 2003.

[29] S. Oladyshkin and W. Nowak. Data-driven uncertainty quantification using the arbitrary polynomial chaos expansion. *Reliability Engineering amp; System Safety*, 106:179–190, 2012.

[30] R.J. Patton. Fault-tolerant control: The 1997 situation. *IFAC Proceedings Volumes*, 30(18):1029–1051, 1997.

[31] R.J. Patton and J. Chen. A review of parity space approaches to fault diagnosis. *IFAC Proceedings Volumes*, 24(6):65–81, 1991.

[32] R.J. Patton, J. Chen, and S.B. Nielsen. Model-based methods for fault diagnosis: Some guidelines. *Transactions of the Institute of Measurement and Control*, 17(2):73–83, 1995.

[33] Y. Qian, C. Jackson, F. Giorgi, B. Booth, Q. Duan, C. Forest, D. Higdon, Z.J. Hou, and G. Huerta. Uncertainty quantification in climate modeling and projection. *Bulletin of the American Meteorological Society*, 97(5):821–824, 2016.

[34] S. Raychaudhuri. Introduction to monte carlo simulation. In *2008 Winter Simulation Conference*, pages 91–100, 2008.

[35] I. Rish. Distributed systems diagnosis using belief propagation. In *In Allerton Conference on Communication, Control and Computing*, 2005.

[36] V Rostampour, R Ferrari, and T Keviczky. A set based probabilistic approach to threshold design for optimal fault detection. *2017 American Control Conference (ACC)*, 2017.

[37] S. Simani, C. Fantuzzi, and R.J. Patton. *Model-Based Fault Diagnosis in Dynamic Systems using Identification Techniques.* Springer, 2003.

[38] Ralph C. Smith. *Uncertainty quantification: Theory, Implementation, and Applications.* SIAM, 2014.

[39] S. Streif, F. Petzke, A. Mesbah, R. Findeisen, and R. D. Braatz. Optimal experimental design for probabilistic model discrimination using polynomial chaos. *IFAC Proceedings Volumes*, 47(3):4103–4109, 2014.

[40] E.E. Tiniou, P.M. Esfahani, and J. Lygeros. Fault detection with discrete-time measurements: An application for the cyber security of power networks. In *52nd IEEE Conference on Decision and Control*, pages 194–199, 2013.

[41] G. Verdier and A. Ferreira. Adaptive mahalanobis distance and nearest neighbor rule for fault detection in semiconductor manufacturing. *IEEE Transactions on Semiconductor Manufacturing*, 24(1):59–68, 2011.

[42] Z. Wang, L. Wang, K. Liang, and Y. Tan. Enhanced chiller fault detection using bayesian network and principal component analysis. *Applied Thermal Engineering*, 141:898–905, 2018.

[43] C. Yang, J. Liu, Y. Zeng, and G. Xie. Real-time condition monitoring and fault detection of components based on machine-learning reconstruction model. *Renewable Energy*, 133:433–441, 2019.

[44] J. Yu and M. Rashid. A novel dynamic bayesian network-based networked process monitoring approach for fault detection, propagation identification, and root cause diagnosis. *AIChE Journal*, 59(7):2348–2365, 2013.

Glossary

List of Acronyms

FAR	False Alarm Rate
FDD	Fault Detection and Diagnosis
FTC	Fault Tolerant Control
MDR	Missed Detection Rate
FDI	Fault Detection and Identification
FD	Fault Detection
MCC	Multivariate Control Charts
MD	Mahalanobis Distance
PC	Polynomial Chaos
PDF	Probability Density Function
BN	Bayesian Networks
BP	Belief Propagation
MPBUP	Message Passing Bilinear Uncertainty Propagation
SOP	Sum of Product
MC	Monte Carlo
LLN	Law of Large Numbers
SSM	State Space Model
MIMO	Multiple-Input and Multiple-Output
EKF	Extended Kalman filter

List of Symbols

α	User Defined Constant
$\bar{d}$	Mean Vector of d
$\bar{r_y}$	Output Residual Mean
$\chi(k)$	Process Noise
η_i	Water Diverted from one tank to another
$\gamma(k)$	Total Uncertainty Acting in the System

$\hat{x}(k)$	Estimated Value of the State
κ_l	Input Coefficient Vector in MPBUP algorithm
κ_l	Input Coefficient Vector in MPBUP algorithm
$\mathcal{J}(q)$	Transversed Edge Set
$\mathcal{V}(q)$	Visited Node Set
ν_i	Manipulated Input acting in the System
ω_l	Input Coefficient Matrix in MPBUP algorithm
ω_l	Input Coefficient Matrix in MPBUP algorithm
ψ_l	Input-Output Coefficient Matrix in MPBUP algorithm
ψ_l	Input-Output Coefficient Matrix in MPBUP algorithm
Σ_d	Covariance Matrix of d
Σ_{r_y}	Output Residual Covariance
θ_l	Output Coefficient Vector in MPBUP algorithm
θ_l	Output Coefficient Vector in MPBUP algorithm
$\tilde{A}$	State Matrix Uncertainty
$\tilde{B}$	Input Matrix Uncertainty
$\tilde{C}$	Output Matrix Uncertainty
ε_α	Probabilistic Threshold Bound
φ	Coefficient Matrix in MPBUP Algorithm
ς_i	Pump gain in the Four Tank System
$\xi(k)$	Measurement Noise
d	Vector containing Concatenation of all the Nodes
f_h	Factor Nodes in MPBUP Algorithm
r_x	Residual in States
r_y	Residual in Output States
x_i	Level of Water in Each Tank
A	State Matrix
a	Output acting in the MPBUP algorithm
A_i	Area of the tank
a_{p_i}	Area of the Pipe
B	Input Matrix
b	Input acting in the MPBUP algorithm
C	Output Matrix
h	Sampling Time
L	Observer Gain