The Dissertation Committee for Fu Zhang
certifies that this is the approved version of the following dissertation:

Simulation of Dynamic Systems with Uncertain Parameters

Committee:

Raul G. Longoria, Supervisor

Matthew Campbell

Richard H. Crawford

Eric P. Fahrenthold

Maruthi R. Akella

Simulation of Dynamic Systems with Uncertain Parameters

by

Fu Zhang, B.S., M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2004

Dedicated to my families.

In this world full of uncertainties, their love to me is definitely certain.

Acknowledgments

There are many people that have played important role in supporting me in writing this dissertation. First and foremost is Dr. Raul Longoria, who not only served as my advisor but also encouraged and challenged me throughout my academic program. He patiently guided me through the dissertation process and help me to pursue my ideas.

I would like to express my thanks to all other members of my doctoral committee for their invaluable comments and suggestions. I appreciate the insights of Dr. Matthew Campbell on optimization techniques, Dr. Maruthi R. Akella's opinions on Boundary Theorem. I especially thank Prof. Eric P. Fahrenthold for nominating me as the recipient of H. Grady Rylander Longhorn Mechanical Engineering Club Excellence in Teaching Fellowship when I was working as his teaching assistant. I also thank his suggestions on where to apply my research results. I thank Prof. Richard H. Crawford for his invaluable comments on regression techniques. I especially thank him for his generous help to my wife and me.

Mr. Robert Thelen from the Center for Electromechanics (CEM) at University of Texas at Austin has been such a great support for me when I was working as a graduate research assistant. The other ALPS team members, Mr. Doug Wardell, Dr. Angelo Gattozzi, Mr. Matthew Caprio, I would like

to thank them all. I had a great time working with you and learned a lot from them.

Ms. Cindy Raman and Ms. Ruth Schwab from Mechanical Engineering Department and Ms. Jo Ann Richmond from CEM are especially thanked for their care and help. My friends, Gilberto Lopez, Seyoon Kim, Chinmaya Patil and Jarrett Goodell, thanks for your help and support. I had a lot of fun with you all.

I would also like to thank Ford Motor Company for providing me the Ford Motor Research Fellowship to support my study on dual power systems.

Finally, I am most grateful to my families for their endless love, care, and support.

Simulation of Dynamic Systems with Uncertain Parameters

Publication No. _____

Fu Zhang, Ph.D.
The University of Texas at Austin, 2004

Supervisor: Raul G. Longoria

This dissertation describes numerical methods for representation and simulation of dynamic systems with time invariant uncertain parameters. Simulation is defined as computing a boundary of the system response that contains all the possible behaviors of an uncertain system. This problem features many challenges, especially those associated with minimizing the computational cost due to global optimization. To reduce computational cost, an approximation or surrogate of the original system model is constructed by employing Moving Least Square (MLS) Response Surface Method for non-convex global optimization. For more complicated systems, a gradient enhanced moving least square (GEMLS) response surface is used to construct the surrogate model more accurately and efficiently. This method takes advantage of the fact that parametric sensitivity of an ODE system can be calculated as a by-

product with less computational cost when solving the original system. Furthermore, global sensitivity analysis for monotonic testing can be introduced in some cases to further reduce the number of samples. The proposed method has been applied to two engineering applications. The first is hybrid system verification by reachable set computing/approximation. First, the computational burden of using polyhedron for reachable set approximation is reviewed. It is then proven that the boundary of a reachable set is formed only by the trajectories from the boundary of an initial state region. This result reduces the search space from $\mathbb{R}^n$ to $\mathbb{R}^{n-1}$. Finally, the GEMLS method proposed is integrated with oriented rectangular hull for reachable set representation and an approximation with improved accuracy and efficiency can be achieved. Another engineering application is model-based fault detection. In this case, a fault free system is modeled as a parametric uncertain system whose parameters belong to a given bounded set. The performance boundary of a fault free system can be acquired by using the proposed approach and then employed as an adaptive threshold. A fault is defined when system parameters do not belong to the set due to malfunction or degradation. Once such a fault occurs, the monitored system performance will extend beyond the normal system boundary predicted.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The use of numerical models for the simulation of physical system has greatly impacted our approach to engineering and science. However, modeling of physical systems is often complicated by the presence of uncertainties, resulting from lack of information, incomplete scientific understanding, errors in measurement, or manifested in the different predictions from different modeling systems, etc.

This dissertation is concerned with representation and simulation of dynamic systems with uncertain parameters. In this chapter, the problem is first stated in a mathematical form, followed by background and motivation. The goals and contributions are also included, and an outline is presented at the end of the chapter.

1.1 Problem Statement

This dissertation posits to study the behavior of dynamic systems with uncertain parameters. Consider the Initial Value Problem (IVP) of a dynamic

system represented by the Ordinary Differential Equations (ODE),

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, t) \\ \mathbf{y} = \mathbf{g}(\mathbf{x}, \mathbf{u}) \\ \mathbf{x}(0) = \mathbf{x}_0 \end{cases} \tag{1.1}$$

$$\mathbf{x} \in \mathbb{R}^n \ , \boldsymbol{\lambda} \in \mathbb{R}^m \ , \ \mathbf{u} \in \mathbb{R}^p \ , \ \mathbf{y} \in \mathbb{R}^q.$$

In this formation, $\mathbf{x}$ is an n-dimensional state vector, with the initial state vector, $\mathbf{x}_0$. The m-dimensional uncertain parameter vector is $\boldsymbol{\lambda}$, $\mathbf{u}$ is the p-dimensional input vector, and $\mathbf{y}$ is the q-dimensional output vector. Both $\mathbf{f}$ and $\mathbf{g}$ are vector-valued functions.

Definition 1.1 A *determined system* is a system described by Eq. (1.1), in the case where parameter $\boldsymbol{\lambda}$ and initial conditions $\mathbf{x}_0$ are precisely known.

Definition 1.2 A *trajectory* is the path of output $y_j, j = 1, 2, ..., q$, of a determined system in the state space, ranging from t_0 to t_f.

Definition 1.3 A *parametric uncertain system*[1] is a system described by Eq. (1.1), in the case where the parameters $\boldsymbol{\lambda}$ or initial conditions $\mathbf{x}_0$ are bounded,

$$\begin{array}{l} \lambda_i \in \left[\underline{\lambda_i}, \overline{\lambda_i}\right], i = 1, 2, ..., m \\ x_0^j \in \left[\underline{x_0^j}, \overline{x_0^j}\right], j = 1, 2, ..., n \end{array} \tag{1.2}$$

where $\underline{x}$ is the lower bound and $\overline{x}$ is the upper bound. Also, this study is limited to *time invariant parametric uncertain systems*, an uncertain parametric system with $\dot{\boldsymbol{\lambda}} = \mathbf{0}$.

[1] In this dissertation, *parametric uncertain system* strictly refers to a system defined by **Definition 1.3**. *System with uncertain parameters* refers to a more general concept.

Definition 1.4 The parameters of an ODE system construct a *parameter space*. The bounded uncertain parameter, $\boldsymbol{\lambda}$, and initial condition, $\mathbf{x}_0$, represent a *hyper rectangle* in parameter space and state space, respectively. For a determined system, these two hyper rectangles are two points.

Obviously, for parametric uncertain systems, there are an infinite number of output trajectories formed by all possible system parameter values. An arbitrary trajectory can be denoted as $\mathbf{y}(t, \forall\boldsymbol{\lambda})$.

At a given instant of time, t_s, any output of a parametric uncertain system should be bounded by,

$$\min(y_k(t_s, \forall\boldsymbol{\lambda})) \leq y_k(t_s, \forall\boldsymbol{\lambda}) \leq \max(y_k(t_s, \forall\boldsymbol{\lambda})), k = 1, 2, ..., q$$

Definition 1.5 The *lower and upper envelope* of a parametric uncertain dynamic system is defined as,

$$\begin{aligned} \underline{y_k} &= \bigcup_{t\in[0,T]} \min\left(y_k\left(t, \forall\lambda\right)\right) \\ \overline{y_k} &= \bigcup_{t\in[0,T]} \max\left(y_k\left(t, \forall\lambda\right)\right) \\ k &= 1, 2, ..., q \end{aligned} \tag{1.3}$$

Geometrically speaking, $\underline{y_k}$ and $\overline{y_k}$ represent the lower and upper curves that are projections of the boundary of the union of all possible trajectories, to the y_k plan, as shown in Figure-1.1.

Definition 1.6 *Simulation* is defined, in the context of parametric uncertain system, as the process of computing the envelope $\underline{y_k}, \overline{y_k}$.

Boundaries and envelopes represent all the possible trajectories of an uncertain system by a single image. The boundaries split the set of possible

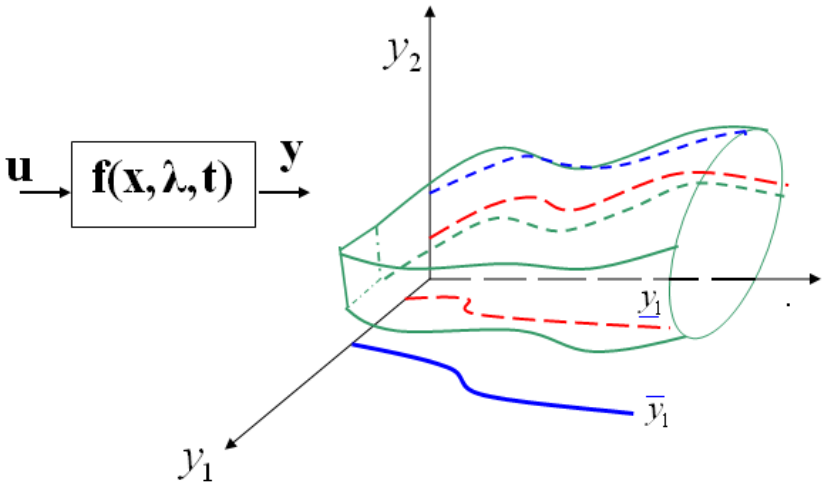


Figure 1.1: Simulation of dynamic system with uncertain parameters

values of a specific output variable at a time point into two subsets: the allowed values and the forbidden ones, according to the model and the input applied to the system [4]. The generation of the exact boundary and envelopes are not realistic goals in most practical cases. It is difficult or even impossible to determine these measures. Hence the problem to be studied in this dissertation is *how to accurately and effectively find approximation of* $\underline{y_k}$ *and* $\overline{y_k}$*, denoted as* $\underset{\sim}{y_k}$ *and* $\widetilde{y_k}$. An illustration of this problem is shown in Figure-1.1.

1.2 Background and Motivation

A fundamental task in engineering and science is the construction and simulation of models for representing real systems. For continuous dynamic

systems, ODE models are a common representation. Conventionally, such models consist of a system of differential equations that describe the trajectory of state variables over time. Making predictions from such determined models has become straightforward and efficient, given the advancement and availability of numerical ODE solvers. However, the value of this approach to modeling and simulation depends on the accuracy of the ODE for representing the physical system of interest. For many problems, accurate ODEs may be difficult to find due to the existence of uncertainties [45]. For realistic systems, one can account for uncertainties of various types in the mathematical model of the system. Uncertainty may occur in the parameters describing the mathematical model, or in the formulation of the mathematical model used to describe the system of interest, etc. This research is motivated by the following issues:

1. *Design/simulation dilemma*: Modeling and simulation have been widely used as design and decision making tools for decades. In most cases, the designer who uses simulation as a design tool faces a dilemma: while many of the simulations must be done in the early design stage, parameters needed for simulation are not completely known. In this stage, in most cases, the engineers may have a good understanding of the physical principles involved in modeling the system, however, they will not have precise knowledge of the values of each parameter, as the process of design is what helps determine the parameters of the system. Thus, design/simulation by nature is usually an iterative trial and and error

process. Many iterations may be needed. However, if one simulation is computationally expensive, an iteration process is not desirable. This is well illustrated in engineering applications such as power systems design [14, 36], circuit design [28], vehicle design [13, 34], etc. The author has encountered this problem while involved in the Advanced Locomotive Propulsion System (ALPS) project [84].

2. *Uncertainties in real world.* Assume that a design is finalized and the designer has found a set of parameters for a given component. However, any real component may have manufacturing tolerance, or the properties of the materials may have variability. Therefore, it is unlikely that the fabricated component/system will exactly reflect the original design. As such, a modeling and simulation tool that has the ability to represent and predict the behavior of the component with tolerance presents a more useful tool for engineers.

3. *Design/simulation for uncertainties.* Design is riddled with uncertainties. For certain parameters in a model, their value may not be a discrete value but a bounded set. For example, the inertia of a vehicle is an important factor for the design of the suspension system. Yet, the number of passengers in the car may vary from 1 to 5. Consequently, the total mass of the vehicle is left uncertain but bounded. The suspension must work well in this range. Therefore, it will be better to use a simulation tool that is able to handle such uncertain parameters.

4. *Hybrid system verification.* A *hybrid system* may be categorized as a switching system. Such systems will switch among several continuous systems represented by ODEs. For such systems, the continuous state space is divided into several operational regions. Switching between different continuous system is determined by whether the state of the system enters into these regions. Verification of such a system determines if a system will enter a certain region, given an uncertain initial state region. This problem can be studied by *reachable set computing* and will be discussed in detail in Chapter 4.

5. *Fault detection.* The performance of a determined system in 'good' operating condition can ideally be represented by a state trajectory. However, due to different uncertainties, the acceptable performance of a system is best represented by a boundary. Whenever the monitored system's performance is outside this boundary, a fault may have occurred. A fault detection problem can thus be modeled by simulation of a parametric uncertain system. The details of this application will be discussed in Chapter 5.

In each of the above cases, inherent uncertainties are common to the modeling and simulation task. Thus there is a strong motivation for the development of efficient methods for simulating dynamic systems with uncertain parameters.

1.3 Goal of This Research

The goal of this research is to formulate methods for the simulation of *parametric uncertain dynamic systems*. This dissertation focuses on parametric uncertainties because in most cases other uncertainties can be transformed to parametric uncertainties. The proposed methods should be a trade-off between efficiency and accuracy. These methods should be able to deal with large uncertainties, certain number (< 10) of uncertain parameters, and be easily applied for practical applications. Specifically, the goals of this research are to find methods that increase accuracy without losing too much computational efficiency and are easily integrated with commercial simulation software. This dissertation also examines classes of engineering applications that can be solved by this method.

1.4 Summary of Key Contributions of This Research

This dissertation presents a hybrid method for solving the problems described above based on Response Surface Method (RSM). The possible contributions of this research are:

1. A hybrid numerical method for parametric uncertain system simulation that integrates response surface method, sensitivity analysis, monotonic testing and gradient enhanced RSM for enhanced computational efficiency and accuracy. This method is described in detail in Chapter 3.

2. The introduction of sensitivity bands and monotone intervals for para-

metric uncertain dynamic systems, and a novel theorem that simplifies the monotonic testing of a dynamic system to reduce the computational cost of simulation. The detail is in Section 3.6.

3. The derivation and discussion of gradient enhanced moving least square response surface method for function approximation with better accuracy and efficiency. The details are in Section 3.5.3.

4. A boundary theorem that can reduce the search space of reachable set computing problem from $\mathbb{R}^n$ to $\mathbb{R}^{n-1}$. The details are in Section 4.4.3.

5. The applications of the developed method for existing engineering problems, such as hybrid system verification (Chapter 4), fault detection (Chapter 5), to illustrate its capabilities.

1.5 Guide to This Dissertation

The content of this dissertation is arranged as follows: Chapter 2 extensively reviews existing methods used for simulation of dynamic systems with uncertain parameters, and an evaluation is given at the end of the chapter. Based on the review of existing methods, Chapter 3 presents details of the proposed method, which is the core of this research. Chapter 4 presents hybrid system verification problem. First the boundary theorem is derived and then the proposed method integrated with oriented rectangular hull (ORH) for reachable set computing is presented. Chapter 5 presents fault detection

by employing the proposed method. Chapter 6 summarizes the dissertation and suggests future work.

Chapter 2

Literature Review

In this chapter, existing and past research work on categorization, representation and propagation of uncertainty in modeling and simulation are reviewed.

2.1 Uncertainty Categorization

Uncertainties exist in every stage of the modeling and simulation task. They can be categorized into different types for different applications.

2.1.1 Aleatory and Epistemic Uncertainty

One of the most widely recognized distinctions in uncertainty types is between *aleatory* and *epistemic* uncertainty [62]. The term *aleatory uncertainty* is used to describe the inherent variation associated with the physical system or the environment under consideration. Sources of aleatory uncertainty can commonly be modeled as 'random' quantities. *Aleatory* uncertainty is also called *variability, inherent uncertainty or natural uncertainty* in some literature. For example, in air pollution systems, the turbulent atmosphere and unpredictable emission-related parameters are types of *aleatory* uncer-

tainties [42]. They can be represented as random distributions that can take on values in an established or known range, but for which the exact value will vary by chance from time to time. The mathematical representation most commonly used for *aleatory* uncertainty is a probability distribution. Propagation of these distributions through a modeling and simulation process is well developed and is described in many texts [62].

Epistemic uncertainty derives from some level of ignorance of the system or the environment. The term *epistemic* uncertainty is used to describe any lack of knowledge or information in any phase or activity of the modeling process. As a result, an increase in knowledge or information can lead to a reduction in this kind of uncertainty. Examples of sources of epistemic uncertainty are when there is little or no experimental data for a **fixed (but unknown)** physical parameter. As opposed to *aleatory* uncertainty, the mathematical presentation of *epistemic* uncertainty has proven to be much more of a challenge. In fact, it is believed that the preeminent issue in uncertainty analysis of systems is the representation and propagation of epistemic uncertainty [62].

Uncertainty associated with model formulation and application can also be classified as 'reducible' and 'irreducible'. *Aleatory* or Natural uncertainty is 'inherent' or irreducible, while *Epistemic* uncertainty is reducible.

2.1.2 Parametric and Model Uncertainty

For any particular physical system of interest that is mathematically modeled, we can distinguish between parametric uncertainty and model uncertainty.

- Model uncertainty: Mathematical models are necessarily simplified representations of the phenomena being studied, and a key aspect of the modeling process is the judicious choice of model assumptions. In fact, sometimes modeling is rather subjective, depending on the modeling team and its skills and therefore one often refers to it as to the 'art of modeling' [69]. The optimal model will provide the greatest simplifications, while providing an adequately accurate representation of the processes affecting the phenomena of interest. Hence, the structure of mathematical models employed to represent a dynamic systems is often a key source of uncertainty [42]. In addition to the significant approximations often inherent in modeling, sometimes competing models may be available. Isukapalli and Georgopoulus [42] have categorized model uncertainty in the following way:

    1. Model structure uncertainty: Uncertainty arises when there are alternative sets of scientific or technical assumptions for developing a model. For example, a simplified DC/AC inverter can be modeled by using ideal switches or by using an average state model of the switching devices. In general, the results of the results from these

different models are very close. In this case, one can be confident that the decision is robust in the face of uncertainty. However, in certain cases, these two models may generate some different conclusions, other models may be used to evaluate the results given by these two models.

2. Model detail uncertainty: All models are, to some extend, simplifications of real physical systems. Such as simple linear model of a very complicated nonlinear system or models that only consider the low order dynamics that are of interest. For example, for an inverter, for different purpose, models (for simple to complex) such as behavioral model, average model, detailed device level model can be used. Uncertainties from simplified models can sometimes be characterized by comparison of their predictions to those of more detailed models.

Model uncertainty is fundamentally *epistemic* in nature. Further detail categorization of model uncertainty can be found in [42].

- Parametric Uncertainty. Uncertainties in model parameter estimates stem from a variety of sources. For example, measurement of parameters are often associated with strong uncertainties, especially for products with small geometric features, such as VLSI chip or MEMS devices. Other type of parametric uncertainties arise from the design/simulation dilemma: at an early design stage, simulations are needed to aid design

decisions while the parameters of the design are not yet known. Further, from a simulation point of view, some parameters are 'inherent' uncertain. For example, the mass of a vehicle depends on the number of passengers in the car. Parametric uncertainly can be a mixture of *epistemic* and *aleatory* uncertainties.

In fact, parametric uncertainty can be treated as a more fundamental uncertainty, compared to model uncertainty. Model uncertainty can be a result of parametric uncertainty. For example, if more parameters (information) are available, a higher fidelity model can be built to avoid model structure uncertainty.

2.2 Representation of Parametric Uncertainties

The representations of parametric uncertainties are strongly related to the uncertainty types associated with the parameters. Some of the representations of uncertainty that occur in modeling and simulation of physical systems include:

1. Pure qualitative representation [49, 50]. In certain conditions, due to lack of information, a parameter may only be described qualitatively, such as 'pretty big' or 'small' etc.

2. Strong statistical information [62]. Sometimes large quantities of experimental data are available, sufficient to derive or convincingly verify a

particular statistical model of the uncertain parameter. In this case, the uncertain parameter can be modeled by its probability density function (pdf). For example, the mass of a component may be uncertain due to manufacturing errors, so mass can be modeled by a normal distribution function with its mean value and variation.

3. Intervals [4, 34, 78]. Sometimes the upper and lower bounds on parametric values can be provided, typically from expert input or design constraints. For example, the number of passengers in car could range from 1 to 5 and thus the mass of the car will range from the lower limit to upper limit. This representation is very common in design problems, in which most unknown parameters are represented in intervals. Systems with interval uncertain parameters are called *interval systems* or *semi-qualitative system* [4, 64].

4. Mixed representation. In [62], Oberkampf et al., pointed out that more commonly, real problems typically present a mixture of all the uncertain sources across different parameters. Moreover, there may be multi-information sources for one parameters [62]. For example, one parameter might be independently estimated from several experts and each one gives a different estimation. Thus, there is the challenge of aggregating these disparate representations into a single representation, which might have a hybrid or mixed mathematical form.

The uncertain parameters represented by the methods reviewed above can be further divided into two categories: time variant or time invariant. Table 2.1 summarizes the most common representations of parametric uncertainties. The methods used to propagate time variant and time invariant uncertainties are very different, and the later case will is more challenging [62].

Table 2.1: Representation of uncertain parameters

Representation of uncertain parameter λ	
Time invariant λ	Time variant λ(t)
λ is known qualitatively	λ(t) is known qualitatively
λ is a random variable with specified pdf	λ(t) is a stochastic process whose probabilistic structure is known
λ is a random variable with specified pdf having uncertain parameters such as mean and variance	λ(t) is a stochastic process whose probabilistic structure may contain uncertain parameters
λ belongs to an interval	λ(t) belongs to an interval
Mixed representation	Mixed representation

In most engineering design/simulation stages, strong statistical information or large quantities of experimental data may not be available, and the probabilistic structure of the uncertain parameters is not generally known. Thus, interval representation is perhaps the most appropriate representation. In this case, the uncertain parameters form a hyper rectangular shape in parameter space. A more general and flexible representation of uncertain parameters is a set, which can be any shape in the parameter space. In this research, the focus is on interval/set representation of time invariant parametric uncertainties.

2.3 Propagation of Parametric Uncertainties

The most important step in uncertainty analysis is the propagation of the parametric uncertainties. This is to study of how uncertainties in system parameters can impact the output of the system. Various uncertainty propagation methods are reported in literatures and these depend on the nature and representation of the parametric uncertainty.

2.3.1 Monte-Carlo Simulation

Monte-Carlo Simulation (MCS) is the most widely used means for uncertainty analysis, with applications ranging from aerospace engineering to zoology. It is a technique which has had a great impact in many different engineering fields. This technique derives its name from the casinos in Monte-Carlo - a Monte-Carlo simulation uses random numbers to model some sort of a process. This technique works particularly well when the process is one where the underlying probabilities are known but the results are more difficult to determine [42]. Monte-Carlo methods can be loosely described as sample based statistical simulation methods, where statistical simulation is defined in general terms to be any method that utilizes sequences of random numbers to perform the simulation. Monte-Carlo methods have been used for centuries, but only in the past several decades has the technique gained the status of a full-fledged numerical method capable of addressing the most complex applications [1]. For an ODE system with uncertain parameters, if the uncertain parameters are represented by their pdfs, then Monte-Carlo methods can be

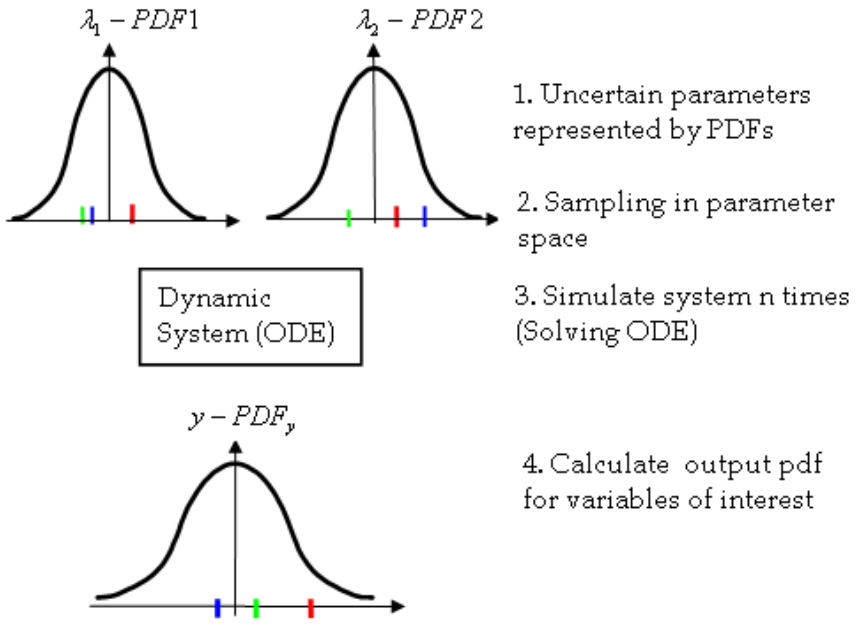


Figure 2.1: Monte-Carlo simulation of uncertain dynamic systems

applied to simulate the system. The Monte-Carlo simulation can proceed by random sampling from the pdf's of the uncertain parameters and many simulations (solving ODEs numerically) are then performed (multiple 'trials' or 'histories'). The final output pdf is then calculated. Figure-2.1 illustrates the idea of Monte-Carlo simulation as applied to an ODE systems with uncertain parameters represented by pdfs. If the uncertain parameters are represented as intervals, then uniform distributions can be used to represent an interval, but the output will not necessary be uniform distribution and some conversions from pdf to an interval should be done. The primary components of a complete Monte Carlo simulation method include the following [1]:

1. Probability distribution functions (pdf's) - the uncertain parameters should be described by a set of pdf's. If the parameters are not represented as pdfs, they should be converted to pdfs.

2. Random number generator - a source of random numbers uniformly distributed on the unit interval must be available.

3. Sampling rule - a prescription for sampling from the specified pdf's, assuming the availability of random numbers on the unit interval, must be given.

4. Scoring (or tallying) - the outcomes must be accumulated into overall tallies or scores for the quantities of interest.

5. Error estimation - an estimate of the statistical error (variance) as a function of the number of trials and other quantities must be determined.

6. Variance reduction techniques - methods for reducing the variance in the estimated solution to reduce the computational time for Monte Carlo simulation.

7. Parallelization and vectorization - algorithms to allow Monte Carlo methods to be implemented efficiently on advanced computer architectures.

When applying the Monte-Carlo method to a dynamic system with ODE solver, it is worth to note that one treats the uncertainty as a time invariant or time variant value. For a time variant uncertain parameter, if the

relationship between the variation of the parameter versus time is not known, then Monte-Carlo method may not be directly applied.

Monte Carlo method has certain advantages:

1. Monte carlo methods involve running the model at a set of sample points and establishing the relationship between inputs and outputs using these sampled results. Since each run is independent of the others, Monte Carlo simulations can be easily parallelized.

2. Monte Carlo methods do not require access to model equations or even the model code and thus is completely general. Since it is completely general it is frequently used to calibrate and validate other methods as benchmarks. Also, existing system solving tools such as ODE solvers can be applied directly.

3. Monte Carlo methods work for both probabilistic and non-probabilistic problems.

The primary disadvantage of Monte Carlo is quite obvious: since it requires a large number of samples, it is very time consuming, especially when simulation of system take a long time, which is not unusual for many engineering applications. The applicability of Mote-Carlo basically is sometimes limited by computing power available.

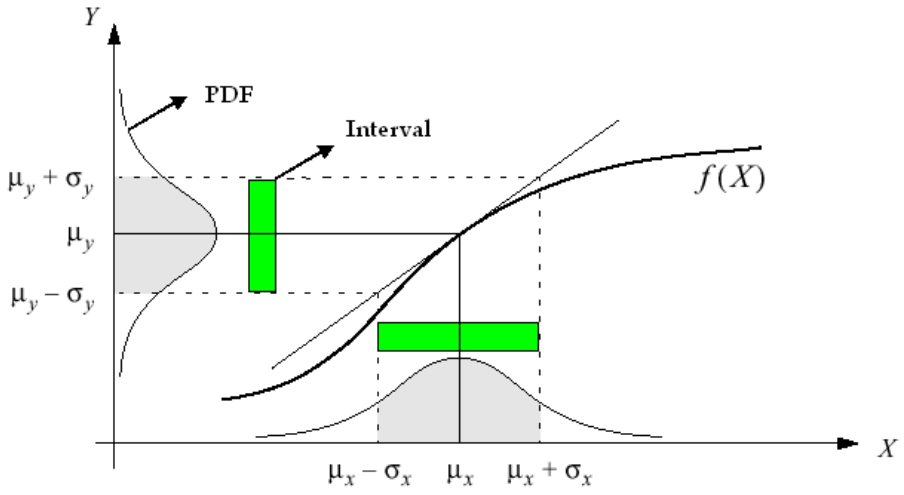


Figure 2.2: First order error propagation, adapted and modified from [5]

2.3.2 First Order Propagation and Error Propagation Law

The idea of this approach is to linearize the system with nominal parameters and thus reduce the computational cost to find the boundary. Figure-2.2 [5] illustrates this idea with a simple case of one output and one unknown parameter. The unknown parameter can be given in terms of intervals or it can be treated as a random variable, and its the mean and variance are assumed given.

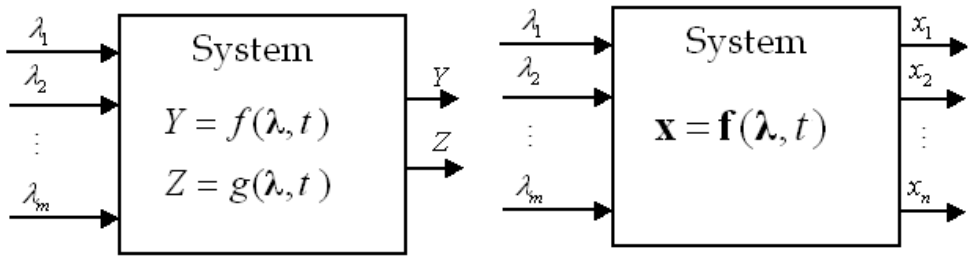
If the uncertain parameters are given as intervals, this method is called First Order Propagation (FOP). It is also called sensitivity based method in some literature [71]. This approach estimates the worst case response using nominal sensitivity and first order Taylor expansion of the original system around a nominal value. Assume the trajectory given by the nominal parameter set $\boldsymbol{\lambda}^N$ is $y(x, u, \lambda_i^N, t), i = 1, 2, ..., n$, and each uncertain parameter λ_i has

disturbance $\delta\lambda_i$, then the system with uncertain parameter $\lambda_i = \lambda_i^N + \delta\lambda_i$ can be approximated as,

$$y(x, u, \lambda_i^N + \delta\lambda_i, t) \approx y(x, u, \lambda_i^N, t) + \sum_{i=1}^{n} \left.\frac{\partial y}{\partial \lambda_i}\right|_{\lambda_i=\lambda_i^N} \delta\lambda_i \tag{2.1}$$

The above equation requires simulating the system with the nominal parameters and calculating the parametric sensitivity coefficient, $\left.\frac{\partial y}{\partial \lambda}\right|_{\lambda=\lambda^N}$. For ODE systems, these two steps can be calculated simultaneously. With little extra cost the sensitivity coefficient can be acquired [24]. Monte-Carlo method can then be applied to Eq. (2.1) to estimate the boundary of the system with less computational cost; or, the upper and lower boundary can be approximated by: $\underline{y_j} \approx y_j(x, u, \lambda_i^N, t) - \left|\sum_{i=1}^{n} \left.\frac{\partial y_j}{\partial \lambda_i}\right|_{\lambda_i=\lambda_i^N}\right| |\delta\lambda_i|$ and $\overline{y_j} \approx y_j(x, u, \lambda_i^N, t) + \left|\sum_{i=1}^{n} \left.\frac{\partial y_j}{\partial \lambda_i}\right|_{\lambda_i=\lambda_i^N}\right| |\delta\lambda_i|$.

If the uncertainties are given in terms of mean and variance, apply the same procedure, and the so called Error Propagation Law (EPL) can be derived as follows [5]:



(a) Error propagation of two output Y, Z (b) Error propagation of n output $x_i, i = 1, 2, ...n$

Figure 2.3: Error propagation law, adapted and modified from [5]

Consider the system with two output y and z shown in Figure-2.3(a), and assume the mean and the variance of λ_i are λ_i^N and δ_i, and the covariance of λ_i, λ_j is δ_{ij}. The mean and variance of the output Y is then,

$$\begin{aligned}
&\mu_Y = f(\lambda_1^N, \lambda_2^N, ..., \lambda_m^N, t) \\
&\delta_Y^2 \approx \sum_{i=1}^{m} \left(\frac{\partial f}{\partial \lambda_i}\bigg|_{\lambda_i=\lambda_i^N} \right)^2 \delta_i^2 + \sum_{i=1}^{m} \sum_{j=1, i\neq j}^{m} \left(\frac{\partial f}{\partial \lambda_i} \frac{\partial f}{\partial \lambda_i} \right)\bigg|_{\lambda_i=\lambda_i^N} \delta_{ij}
\end{aligned} \tag{2.2}$$

The covariance that describes the statistical dependence of Y and Z is,

$$\delta_{YZ} = \sum_{i=1}^{m} \left(\frac{\partial f}{\partial \lambda_i} \frac{\partial g}{\partial \lambda_i} \right)\bigg|_{\lambda_i=\lambda_i^N} \delta_i^2 + \sum_{i=1}^{m} \sum_{j=1, i\neq j}^{m} \left(\frac{\partial f}{\partial \lambda_i} \frac{\partial f}{\partial \lambda_i} \right)\bigg|_{\lambda_i=\lambda_i^N} \delta_{ij}^2 \tag{2.3}$$

If the parameter λ_i, λ_j are independent, the second term, containing their covariance δ_{ij}, disappears from the above equation. In that case, we have the following simplified error propagation law,

$$\begin{aligned}
&\mu_Y = f(\lambda_1^N, \lambda_2^N, ..., \lambda_m^N, t) \\
&\delta_Y^2 = \sum_{i=1}^{m} \left(\frac{\partial f}{\partial \lambda_i}\bigg|_{\lambda_i=\lambda_i^N} \right)^2 \delta_i^2 \\
&\delta_{YZ} = \sum_{i=1}^{m} \left(\frac{\partial f}{\partial \lambda_i} \frac{\partial g}{\partial \lambda_i} \right)\bigg|_{\lambda_i=\lambda_i^N} \delta_i^2
\end{aligned} \tag{2.4}$$

For the general dynamic system with n variables, as shown in Figure-2.3(b), the error propagation law to propagate the error (parametric uncertainties) from the uncertain system parameters to the output variables can be written in matrix form as [5],

$$\begin{aligned}
&\boldsymbol{\mu_X} = \mathbf{f}(\lambda_1^N, \lambda_2^N, ..., \lambda_m^N, t) \\
&\boldsymbol{C_X} = \boldsymbol{J_\lambda} \boldsymbol{C_\lambda} \boldsymbol{J_\lambda^T}
\end{aligned} \tag{2.5}$$

where $\boldsymbol{C_\lambda} = \begin{bmatrix} \delta_1^2 & \delta_{12} & \cdots & \delta_{1m} \\ \delta_{12} & \delta_2^2 & \cdots & \delta_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ \delta_{m1} & \delta_{m2} & \cdots & \delta_m^2 \end{bmatrix}_{m\times m}$ is the covariance matrix of the uncertain parameter. If the parameters are independent, then $\boldsymbol{C_\lambda}$ is a diagonal matrix. In these formulation, $\boldsymbol{J_\lambda} = \begin{bmatrix} \frac{\partial f_1}{\partial \lambda_1} & \frac{\partial f_1}{\partial \lambda_2} & \cdots & \frac{\partial f_1}{\partial \lambda_m} \\ \frac{\partial f_2}{\partial \lambda_1} & \frac{\partial f_2}{\partial \lambda_2} & \cdots & \frac{\partial f_2}{\partial \lambda_m} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial \lambda_1} & \frac{\partial f_n}{\partial \lambda_2} & \cdots & \frac{\partial f_n}{\partial \lambda_m} \end{bmatrix}_{n\times m}$ is the parametric Jacobian matrix, and $\boldsymbol{C_X} = \begin{bmatrix} \delta_{x_1}^2 & \delta_{x_1x_2} & \cdots & \delta_{x_1x_n} \\ \delta_{x_2x_1} & \delta_{x_2}^2 & \cdots & \delta_{x_2x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \delta_{x_nx_1} & \delta_{x_nx_2} & \cdots & \delta_{x_n}^2 \end{bmatrix}_{n\times n}$ is the output covariance matrix.

From the above matrix form, it can be seen that the parametric uncertainty represented by the matrix $\boldsymbol{C_\lambda}$ is propagated to the output through the system parametric Jacobian matrix $\boldsymbol{J_\lambda}$. It shows that, in order to use this approach, the Jacobian matrix must be calculated. As pointed before, for ODE systems, $\boldsymbol{J_\lambda}$ can be calculated with little extra cost.

The advantages of this approach are as follows:

1. Less computational cost. As an alternative to Monte-Carlo simulation, this approach simulates the system based on Eq. (2.1) and Eq. (2.5) only once, and there is no need to simulate the original system equation many times. Although the Jacobian matrix of the system needs to be calculated, for certain systems, including ODE systems, this extra cost

is usually relatively low.

2. Sensitivity information. Since this method calculates the Jacobian matrix, it provides the sensitivity information of the uncertain parameters and can help determine which uncertain parameter will have the largest impact on the output.

The disadvantages of this approach are also quite obvious: because the high order terms are unknown, this approach often leads to overly estimated results, and further it is not predictable whether the results will underestimate or overestimate the actual response bounds [82]. Also, since Taylor linear expansion only works well when the variation is very small, this approach is applicable only when the uncertain parameters have relatively small intervals.

Despite the above disadvantages, this approach has been widely used in many engineering fields, such as for bounding uncertainty in power systems [14, 61] and circuit analysis [71].

2.3.3 Vertex Enumeration

Vertex enumeration simulates all possible extreme cases of parameter variations; i.e, it simulates all the vertex of a hyper-rectangle formed by uncertain parameters represented as intervals. It is computationally less expensive than Monte-Carlo method. However, extreme values of the output may not necessarily occur at these corners of the parameter space [71] and thus the quality of the result is low.

2.3.4 Direct Global Optimization

From the definition of the envelope in Eq. 1.3, and the system equation Eq. 1.1, it is found that the uncertainty propagation problem can be modeled as an optimization problem as follows:

$$\begin{aligned} \underline{\mathbf{y}} &= \bigcup_{t\in[0,T]} \min\left(\mathbf{y}(t)\right) = \bigcup_{t\in[0,T]} \min\left(\mathbf{g}(\mathbf{x}(t),\mathbf{u})\right) \\ \overline{\mathbf{y}} &= \bigcup_{t\in[0,T]} \max\left(\mathbf{y}(t)\right) = \bigcup_{t\in[0,T]} \max\left(\mathbf{g}(\mathbf{x}(t),\mathbf{u})\right) \\ \text{s.t.} &\begin{cases} \mathbf{x}(t) = \mathbf{x}_0 + \int_0^t \mathbf{f}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda},\tau)d\tau \\ \boldsymbol{\lambda} \in [\underline{\boldsymbol{\lambda}}, \overline{\boldsymbol{\lambda}}], \mathbf{x}_0 \in [\underline{\mathbf{x_0}}, \overline{\mathbf{x_0}}] \end{cases} \end{aligned} \tag{2.6}$$

The above equation shows that computation of $\underline{\mathbf{y}}$ and $\overline{\mathbf{y}}$ is an optimization problem. At each time step, the above optimization problem needs to be solved. To solve this problem, the uncertain space construct by uncertain parameter $\boldsymbol{\lambda}$ and uncertain initial condition $\mathbf{x}_0$ must be searched. To evaluate the objective function, Eq. (2.6), a simulation (integration of ODEs) needs to be done and optimization algorithms should be integrated with the ODE solver.

Note that this approach can only be applied to time invariant systems, because time variant systems can not be simulated, unless the way that the parameters change versus time is known. Meanwhile, this method is not recursive since at each time step the integration should be initiated from time zero. The detailed computational cost analysis of this optimization problem will be shown in the next chapter.

The quality of the envelopes are determined by the optimization algorithms. For example, if a local optimum is given by the optimization, the

envelope may be under-bounded.

Different optimization algorithms have been applied to this problem, such as Genetic Algorithms(GA) [25],and interval analysis [4, 26, 78].

2.3.5 Interval Analysis

Although many different optimization algorithms can be used for uncertainties propagation, the most used global optimization algorithm to solve Eq. 2.6 is by interval analysis [4, 26, 64, 78].

When the uncertain parameters or system initial conditions are represented using intervals, the problem of uncertain propagation can become a case of finding the range of an interval function. For ODE systems, the range of the state variables must be evaluated every time step. One method for evaluating the range of the function is to use interval analysis [43, 59].

The foundation of interval analysis is interval arithmetic. It is an arithmetic defined on sets of intervals, rather than sets of real numbers. Modern interval arithmetic began with R. E. Moore's pioneering work [59].

If $x \in [\underline{x}, \bar{x}]$ and $y \in [\underline{y}, \bar{y}]$, then the elementary arithmetic operations are defined as [46]:

$$\begin{aligned} x + y &= [\underline{x} + \underline{y}, \bar{x} + \bar{y}] \\ x - y &= [\underline{x} - \bar{y}, \bar{x} - \underline{y}] \\ x \times y &= [\min(\underline{x}\underline{y}, \underline{x}\bar{y}, \bar{x}\underline{y}, \bar{x}\bar{y}), \max(\underline{x}\underline{y}, \underline{x}\bar{y}, \bar{x}\underline{y}, \bar{x}\bar{y})] \\ \frac{1}{x} &= [\frac{1}{\bar{x}}, \frac{1}{\underline{x}}], \text{if} \underline{x} > 0 \text{ or } \bar{x} < 0 \\ x \div y &= x \times \frac{1}{y} \end{aligned} \tag{2.7}$$

Once the fundamental operations are defined, functions defined on intervals can be defined by natural extension. For example, if $f(x) = x(x-1)$ then,

$$f([0,1]) = [0,1]([0,1]-1) = [0,1][-1,0] = [-1,0],$$

which contains the exact range $[-1/4; 0]$. Interval arithmetic will generate a guaranteed over-bound interval that contains the exact range of the original function. This feature guarantees that the envelope generated by solving Eq. 2.6 is an over-bound envelope. For certain applications, such as hybrid system verification, this feature is desired.

However, as can be found from the example presented, there is an intrinsic problem with interval analysis: the interval found by natural extension is highly over-bounded, since there are multi-incidences (the same variable appears more than once in an expression, in the above example, variable x appeared twice and the calculated interval $[-1, 0]$ is much wider than the real range $[-1/4, 0]$). Therefore, a way to obtain tighter interval is to express the function with the lowest number of multi-incidences. Many techniques that search for the best way to express an interval function with minimum multi-incidence have been developed [43].

There are some simulators based on interval arithmetic such as NSIM (Numerical Simulator using Interval Methods) [45] and NIS (Numerical interval simulation) [80]. NSIM is limited to monotonic functions and NIS uses numeric integration algorithms revised for intervals by calculating the range of

the derivative of the state variables at each integration step, then the maximum and minimum of the derivatives of the variables are used for Euler or Runge-Kutta integration algorithms.

These simulators obtain highly over-bounded envelopes due to ignored multi-incidences. A comprehensive review of these types of simulators can be found in [4].

Recently, an improved interval method called Modal Interval Analysis (MIA) was reported and applied to simulate interval ODEs (ODEs system with parameters defined as intervals) [4, 64]. MIA is an extension of the classical interval theory that includes interesting properties. It gives a formal way to study the optimality of an interval function for the results. It provides tools to calculate the ranges of functions with multi-incidences. For monotonic functions, it can give tighter intervals than classical intervals. However, when the function is not monotonic, a splitting algorithm has to be used. It divides the interval of the variables into small intervals, in which the function is monotonic. By doing so, the bound will be tight but the computational cost will be increased.

2.3.6 Differential Inclusion

A Differential Inclusion (DI) is a relation of the form $\dot{\mathbf{x}} \in \mathbf{f}(\mathbf{x})$, where $\mathbf{f}$ is a set-valued map associating any point $\dot{\mathbf{x}} \in \mathbb{R}^{\mathbf{n}}$ with a set $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^{\mathbf{n}}$. As such, the notion of a differential inclusion generalizes the notion of an ordinary differential equation of the form $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$. Therefore, all problems usually

studied in the theory of ordinary differential equations (existence and continuation of solutions, dependence on initial conditions and parameters, etc.) can be studied for differential inclusions as well. Since a differential inclusion usually has many solutions starting at a given point, new types of problems arise, such as investigation of topological properties of the set of solutions, selection of solutions with given properties, and many others. Differential inclusions play an important role as a tool in the study of various dynamical processes described by equations with a discontinuous or multi-valued right-hand side, and they also are very useful in proving existence theorems in control theory [74].

Naturally, an uncertain dynamic system can be represented as a differential inclusion rather than as differential equations. Let us consider a simple example of a second order system [66],

$$\frac{d^2y}{dt^2} + a \cdot \frac{dy}{dt} + y = b. \tag{2.8}$$

This is a simple ODE model. Introducing notation $x_1 = y, x_2 = dy/dt$ we obtain:

$$\begin{aligned} \frac{dx_1}{dt} &= x_2 \\ \frac{dx_2}{dt} &= b - ax_2 - x_1 \end{aligned} \tag{2.9}$$

In more general notation, the state equation is,

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(a, b, \mathbf{x}), \tag{2.10}$$

where $\mathbf{x}$ is a two-dimensional vector, t is the time, and $\mathbf{f}$ is a vector-valued function. Now suppose that the parameters a and b are uncertain parameters

and that the only information we have is the corresponding intervals where the values may belong, or a permissible (may be quite irregular and variable) set on the 2D-plane where the point (a, b) must belong. Note that we know nothing about a possible probability distribution of these parameters and they are not treated as random variables. Thus, Eq. 2.10 takes the following form,

$$\frac{d\mathbf{x}}{dt} \in \mathbf{f}(a, b, \mathbf{x}), \tag{2.11}$$

where $\mathbf{f}$ is a set. This is a differential inclusion (DI). The above process also suggests how parametric uncertain differential equations may be converted into differential inclusions.

The solution to a DI is a tube in the state space contains all possible system trajectories, and each one is one of the solutions to the uncertainty problem. In this very natural way we see that the uncertainty in dynamic system modeling leads to differential inclusions as a corresponding mathematical tool. These methods have been known for about 70 years, mainly to the control theory society [65]. However, accurate, effective, general differential inclusion solvers are not yet available. Some solvers for simplified differential inclusion, such as those assuming the right side is a regular set (a hyper-rectangle), i.e. $\frac{d\mathbf{x}}{dt} \in \square$[1] are reported [35]. Obviously, this is an over-simplified problem.

One reported general differential solver is to treat solving DI as a classical optimal control problem [65, 66]. Observe that any trajectory that reaches a point on the boundary of the solution tube is optimal in some sense; such

[1] $\square$ is used to represent a hyper-rectangle in this dissertation

trajectories can be calculated using Maximum Principle of Pontryagin in classical optimal control field, which is to solve the Hamilton-Jacobian equation. For the above example, the uncertain parameters a, b can be treated as control inputs whose values are bounded in certain intervals. Finding the boundary of the solution tube can be treated as an optimal control problem: with the constrained input a, b, what is the maximum region the system can reach at certain time? In few words, the DI solver provided in [65, 66] works as follows. The user provides the DI in the form of an equivalent control system. To do this, first parameterize the right-hand size (the set $\mathbf{f}$) using an m-dimensional auxiliary control variable $\mathbf{u}$ (for the example above $m = 2, \mathbf{u} = [a, b]^T$). The DI solver automatically generates the equations of the so-called conjugate vector $\mathbf{p}$, and integrates a set of trajectories, each belonging to the boundary of the solution tube. To achieve this, over each trajectory the Hamiltonian $H(\mathbf{x}, \mathbf{p}, \mathbf{u}, t)$ is maximized. This procedure is similar to that used in dynamic optimization in the optimal control problem. Details of this approach are found in references [65, 66, 81].

The above procedure implicitly assumes the uncertain parameters a, b are time variant (since they are treated as input variables) when solving the Hamilton-Jacobian equations using Maximum Principle. If the uncertain parameters are time invariant, a strong constraint $\frac{da}{dt} = 0, \frac{db}{dt} = 0$ will make solving the Hamilton-Jacobian very difficult. Thus, this approach is only applicable to systems with time variant uncertain parameters. Generally, the solution tube of such system is much larger than the corresponding time in-

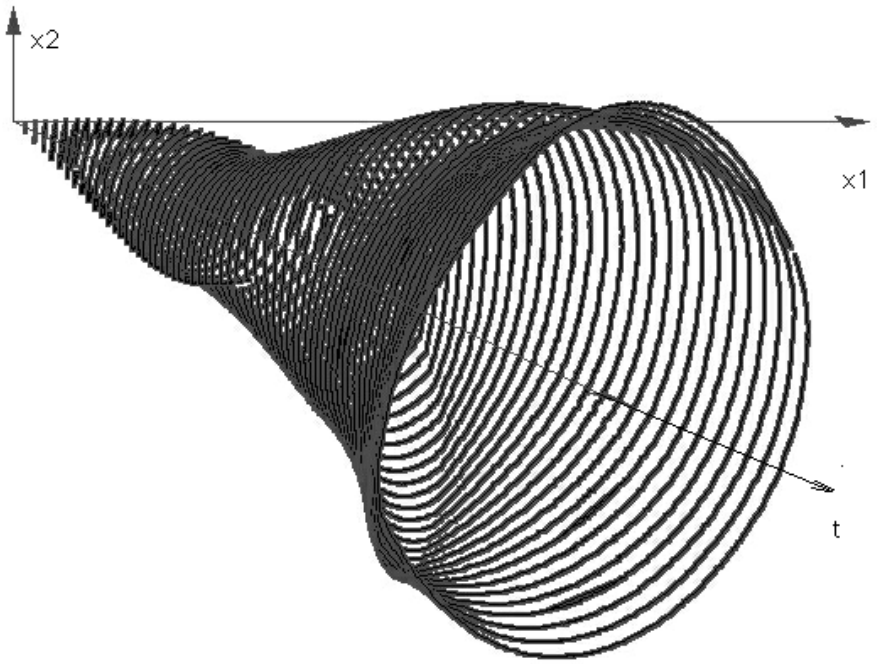


Figure 2.4: The solution tube of time variant differential inclusion, adapted from [66]

variant system. A typical DI solution tube for the time variant system,

$$\begin{aligned} \frac{dx_1}{dt} &= x_2 \\ \frac{dx_2}{dt} &= b - ax_2 - x_1 \end{aligned} \qquad a \in [0.5, 1.5], b \in [-1, 1]$$

is given in Figure-2.4 [66].

2.3.7 Qualitative Model and Simulation

The above mentioned approaches can be seen as semi-quantitative methods. For certain conditions, the system is so uncertain that only qualitative information is known. In this case, one should use a qualitative analysis method.

In short, qualitative analysis is an area of research that seeks ways to model and simulate the everyday, qualitative, non-numerical reasoning humans use to estimate (the range of) possible solutions to some real-world problems with strong uncertainties. This approach has been extensively formulated by Kuipers and his team at the University of Texas at Austin [49].

The idea of qualitative simulation is to build *quantitative models*. A quantitative model is qualitative in two senses [50]. First, the values of variables are described in terms of their relations with a finite set of symbolic landmark values (such as negative, positive), rather than in terms of real numbers. Second, functional relations may be described as monotonic functions (increasing or decreasing over particular ranges) rather than by specifying a functional form. These purely qualitative descriptions can be augmented with semi-quantitative knowledge in the form of real bounding intervals around unknown real values and real-valued bounding envelope functions around unknown real-valued functions (In this case, the model became semi-qualitative).

The value of the derivative is also expressed in a purely qualitative way, such as *increasing*, *decreasing* or *steady*. The qualitative model is represented by *Qualitative Differential Equations* (QDE), which describe the relations among variables. These relations include algebraic operations, derivatives, monotonicity, etc. QDEs are more able than traditional models to express states of incomplete knowledge about continuous mechanisms [49].

Once the QDE is built, certain rules can be applied in the design of simulation algorithms. Details of these rules can be found in [45, 49]. Given

a qualitative description of a state, the result of a qualitative simulator is the qualitative state descriptions that can possibly be direct successors of the current state description. Repeating this process produces a graph of qualitative state descriptions, in which the paths starting from the root are the possible qualitative behaviors. The graph of qualitative states is pruned according to criteria derived from the theory of ordinary differential equations, in order to preserve the guarantee that all possible behaviors are predicted [50], and guarantee to find all possible behaviors consistent with the knowledge in the model. However, the time information of these states is not given since qualitative simulation is not causal. The most widely used and advanced qualitative simulator is called QSIM [45, 49].

The value of qualitative simulation comes from the ability to express natural types of incomplete knowledge of the world, and the ability to derive a provably complete set of possible behaviors in spite of the incompleteness of the model. Qualitative simulation starts with a QDE and a qualitative description of an initial state. Results of it can be used in the design and validation of dynamical systems, such as controllers. A set of qualitative models and their associated predictions can also be unified with a stream of observations to monitor an ongoing dynamical system or to do system identification on a partial model [4, 45, 49]. The disadvantage of this approach is that it is not causal and thus there is no time information in the simulation results which limits its application in practical engineering problem. A qualitative simulation result of parametric uncertain two-tank system was reported in [38].

2.3.8 Simulation with Uncertainties in Frequency Domain

In frequency domain, a time invariant linear dynamic system is characterized by its transfer functions. For a determined, certain system, the transformation can be written as,

$$P(s, \mathbf{q}, \mathbf{r}) = \frac{N(s, \mathbf{r})}{D(s, \mathbf{q})}, \tag{2.12}$$

where $N(s, \mathbf{r})$ and $D(s, \mathbf{q})$ are all polynomials with fixed coefficients, $\mathbf{r} = [r_1, r_2, \ldots, r_m]$ and $\mathbf{q} = [q_1, q_2, \ldots, q_n], m \leq n$. The stability of the system is determined by the characteristic polynomial $D(s, \mathbf{q})$. However, for a system with parametric uncertainties, the transformation function can not be represented by polynomials with fixed coefficients, but interval coefficients, as the following examples shows [7]:

Example 2.1 Torque control of a DC motor [7]. Consider a DC motor driving a load with dumping. The uncertain parameters are motor constant, $K \in [0.2, 0.6] volts/rpm$, and the load moment of inertia, $J_L \in [10^{-5}, 3 \times 10^{-5}] kg - m^3$. The transfer function is,

$$P(s) = \frac{K(J_L s + B_L)}{(Ls + R)(J_m s + J_L s + B_m + B_L) + K^2}.$$

Taking uncertain parameters $q_1 = K$ and $q_2 = J_L$ and fixed parameters $J_m = 2 \times 10^{-3} kg - m^3, B_m = 2 \times 10^{-5} N - m/rpm, L = 10^{-2} H, R = 1\Omega, B_L = 2 \times 10^{-5} N - m/rpm$, an interval transfer function can be formatted that describes this parametric uncertain system as,

$$P(s, \mathbf{q}) = \frac{0.5 q_1 q_2 s + 10^{-5} q_1}{(10^{-5} + 0.005 q_2) s^2 + (0.00102 + 0.5 q_2) s + (2 \times 10^{-5} + 0.5 q_1^2)},$$

with $q_1 \in [0.2, 0.6]$,$q_2 \in [10^{-5}, 3 \times 10^{-5}]$.

The above interval transfer function represents a family of transfer functions. Also, the coefficients of this transfer function are **dependent**, since coefficients of the 2^{nd} order and 1^{st} order term are all related to q_2.

There are two important questions that need to be answered with an interval transfer function and both are related to uncertainty propagation discussed in previous sections.

1. Is the uncertain system stable, given that all the parameters can vary in the intervals? We know that if the poles of a transfer function lie in the strict left half s-plane then the system will be stable. For a system with parametric uncertainties, it is said that the uncertain system is *robust stable*, if for all $\mathbf{q} \in [\mathbf{q}^-, \mathbf{q}^+]$, all roots of $P(s, \mathbf{q}, \mathbf{r})$ lie in the strict left half plane. Obviously, the robust stable question can be converted to finding the boundary of all the roots of the uncertain denominator polynomial $D(s, \mathbf{q})$, or to use Hurwitz criterion for all the possible polynomials determined by the interval polynomials. For example, when q_1 and q_2 vary in their interval, the root locus of the DC motor system is shown in Figure-2.5. It shows that the system is robust stable.

2. Does the uncertain system's performance meet the frequency performance specifications? For the uncertain transfer function $P(s, \mathbf{q}, \mathbf{r})$, the uncertain bode plot is given by,

$$LG(w, q, r) = 20 \log |P(j\omega, \mathbf{q}, \mathbf{r})| .$$

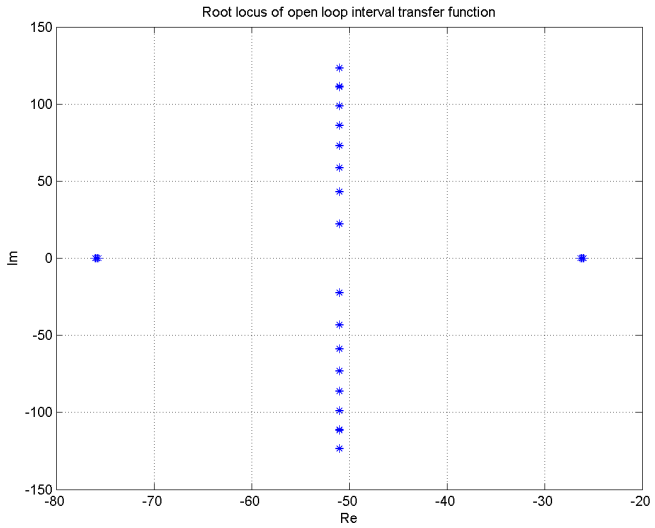


Figure 2.5: Root locus of open loop interval transfer function

Now, a typical performance specification **S** might be as follows: In the frequencies range $\omega_{\min}, \omega_{\max}$, given a band or boundary of the desired performance $\underline{LG}(\omega), \overline{LG}(\omega)$, we seek to guarantee: $\underline{LG}(\omega) \leq LG(\omega) < \overline{LG}(\omega)$, for all $\mathbf{r} \in [\mathbf{r}^-, \mathbf{r}^+], \mathbf{q} \in [\mathbf{q}^-, \mathbf{q}^+]$. When these performance inequalities are met, we can say that the performance specification **S** is robustly satisfied. Again, this problem can be converted to find the boundary of the bode plot, as shown in Figure-2.6. If the boundary of the bode plot is within the specification, then the design is robust.

Clearly, Monte carlo methods can be used to answer the two questions posed. However, it could be too time consuming to calculate the roots and bode plots. For the robust stability problem, Kharitonov's Theorem represented a dramatic breakthrough that initiated research into robust stability problem

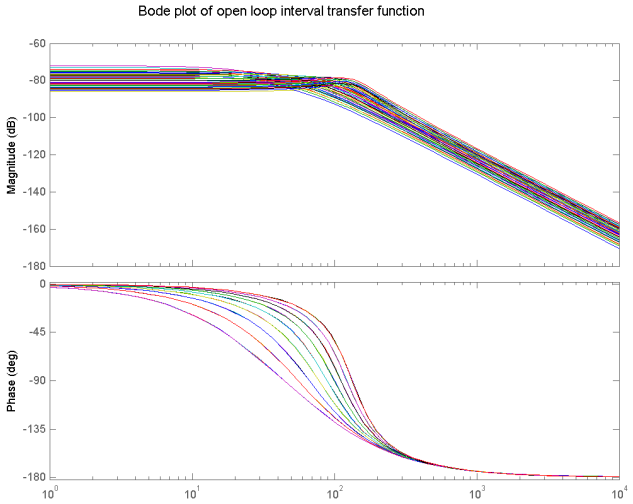


Figure 2.6: Bode plot of open loop interval transfer function of the DC motor system

that greatly reduced the computational cost. This theorem reduces the number of polynomials to be studied to four, a magic number.

Definition *Kharitonov's polynomials*: Associated with the interval polynomial with independent coefficients,

$$p(s,\mathbf{q}) = \sum_{i=0}^{n} [q_i^-, q_i^+] s^i, \tag{2.13}$$

are four Kharitonov's polynomials, defined as,

$$\begin{aligned} K_1(s) &= p^{--}(s) = q_0^- + q_1^- s + q_2^+ s^2 + q_3^+ s^3 + q_4^- s^4 + q_5^- s^5 + \cdots \\ K_2(s) &= p^{-+}(s) = q_0^- + q_1^+ s + q_2^+ s^2 + q_3^- s^3 + q_4^- s^4 + q_5^+ s^5 + \cdots \\ K_3(s) &= p^{+-}(s) = q_0^+ + q_1^- s + q_2^- s^2 + q_3^+ s^3 + q_4^+ s^4 + q_5^- s^5 + \cdots \\ K_4(s) &= p^{++}(s) = q_0^+ + q_1^+ s + q_2^- s^2 + q_3^- s^3 + q_4^+ s^4 + q_5^+ s^5 + \cdots \end{aligned} \tag{2.14}$$

Notice that there are only four Kharitonov's polynomials– independent of the degree of $p(s,\mathbf{q})$.

Kharitonov's theorem [7]. In 1978 the Russian researcher Vladimír Kharitonov proved the following fundamental result: *A continuous-time interval polynomial is robustly stable if its four Kharitonov polynomials are stable.* The coefficients of the polynomial are assumed to be independent.

This theorem means that instead of checking stability of an infinite number of polynomials only the stability of four polynomials need be assessed, and this can be done using the classical Hurwitz criterion. This result was so surprising and elegant that it has been the starting point of a renewed interest in robust control theory [10]. However, this theorem only applies to interval polynomials with independent coefficients.

For interval polynomial with dependent coefficients, a graphical method based on *Zero Exclusion Condition* should be used.

Definition *Kharitonov's rectangle*: Let $P(s, \mathbf{q}) = \sum_{i=0}^{n} [q_i^-, q_i^+] s^i$ be an interval polynomial, then at a fixed frequency ω_0, $P(j\omega_0, \mathbf{q}) = \sum_{i=0}^{n} [q_i^-, q_i^+] (j\omega_0)^i$ describes the set of possible value that $P(j\omega_0, \mathbf{q})$ can assume as $q_i, i = 1, 2, .., n$ ranges over their intervals. $P(j\omega_0, \mathbf{q})$ is called the *Kharitonov rectangle* at frequency ω_0. It can be proved that for an interval polynomial with independent coefficient q_i, $P(j\omega_0, \mathbf{q})$ in z-plane is a rectangle with vertices that are obtained by evaluating the four Kharitonov polynomials $K_1, (s), K_2(s), K_3(s), K_4(s)$ at frequency ω_0; i.e., the vertices of $P(j\omega_0, \mathbf{q})$ are precisely the $K_i(j\omega_0)$. When ω changes from $\omega = 0$ to $\omega = \infty$, the *Kharitonov rectangle* will move in z-plane, as shown in Figure-2.7 [10].

The Zero Exclusion Condition Suppose that an interval polynomial $P(s, \mathbf{q})$ has invariant degree and at least one stable member $p(s, q^0)$. Then it is robustly stable if and only if $z = 0$ is excluded from the Kharitonov rectangle at all nonnegative frequency; i.e.,

$$0 \in P(j\omega, \mathbf{q}), \tag{2.15}$$

for all frequency $\omega \geq 0$. The geometric meaning of this theorem is that if the origin point of z-plane is outside the boundary of the Kharitonov rectangles then the system is robustly stable. For the system in Figure-2.7, since the origin point in outside the boundary, the system is robustly stable.

From the Kharitonov theorem we know that for interval system with independent coefficient, the boundary can be acquired by evaluating only four Kharitonov polynomials. However, for system with dependent coefficients, finding the boundary can be quite difficult. The corner of the Kharitonov rectangle can not be acquired by just evaluating four Kharitonov polynomials. Repeated evaluations of many polynomials may be needed. Some methods, such as level set theory, has been developed to approximate the boundary of such systems [7].

An alternative to interval uncertain systems is spherical uncertain systems, in which the uncertainty set is described as an ellipsoid rather than a box determined by intervals. For such systems, the Kharitonov rectangle will became a Kharitonov ellipse, as shown in Figure-2.8. A tool to find such an ellipse in reported in [40].

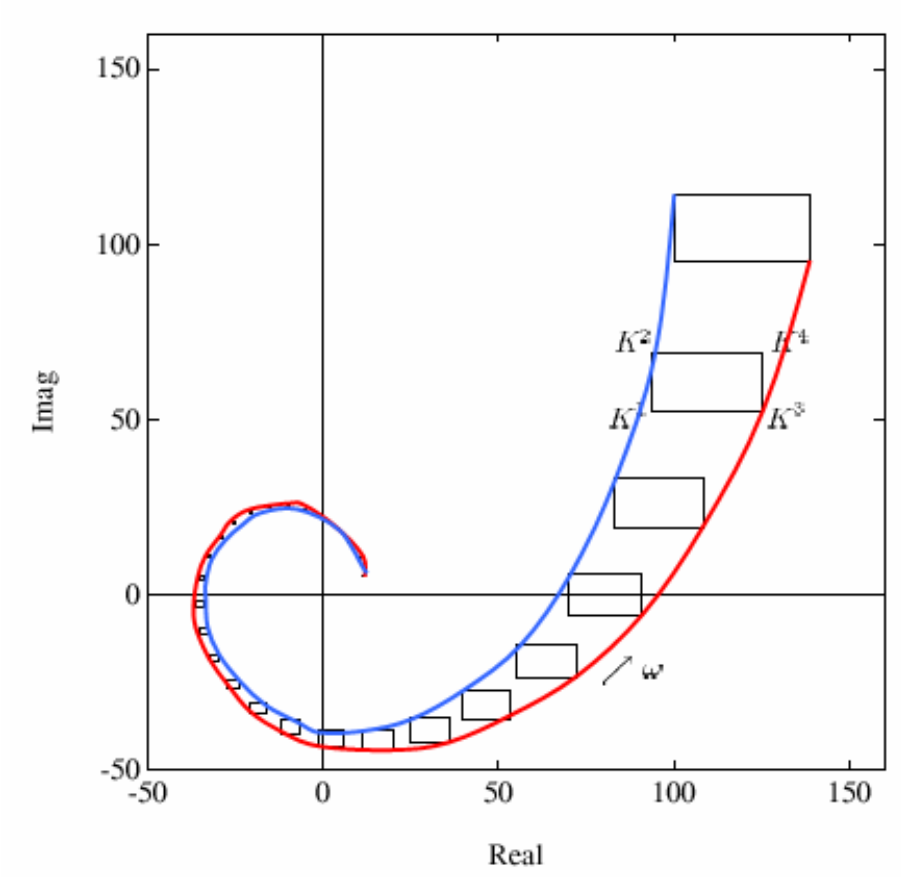


Figure 2.7: Motion of Kharitonov rectangular and zero exclusion condition, adapted from [10]

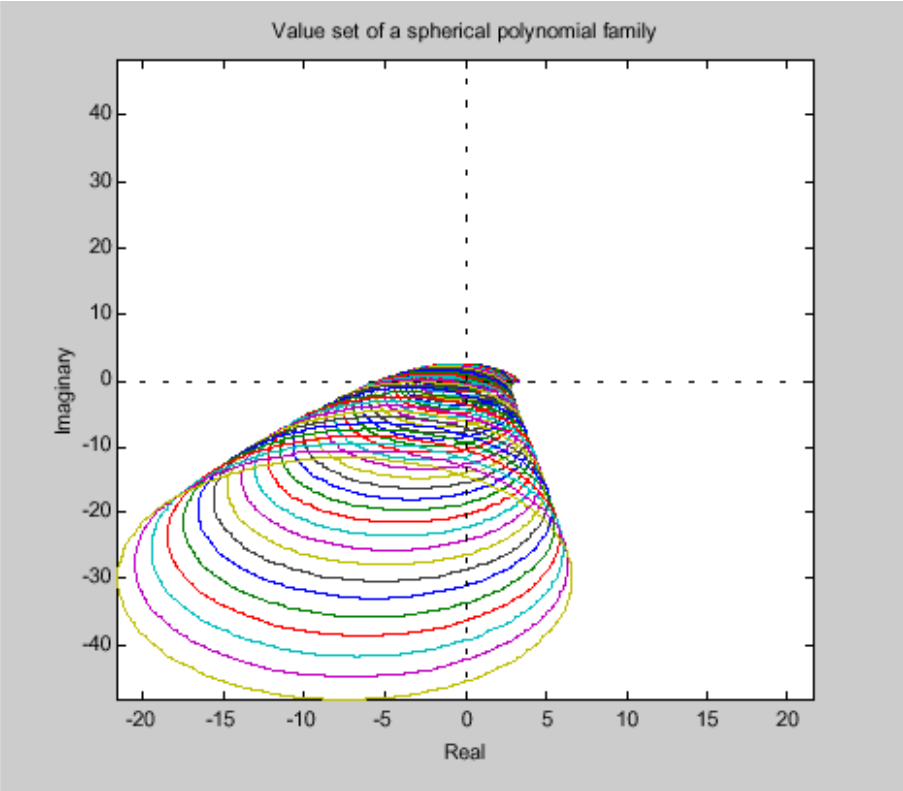


Figure 2.8: Spherical uncertain polynomials families, adapted from [40]

Comparing the frequency domain uncertainty propagation problems and the time domain uncertainty propagation problems reveals many similarities. The ω in frequency is analogous to time t in time domain, and the transfer function is analogous to the ODE in time domain. The problem of frequency uncertainty analysis has also become one of finding the boundaries of certain output, such as the boundary of the bode plot, the boundary of the Kharitonov rectangles etc. In fact, methods used in time domain analysis, such as interval analysis, can also be applied in frequency domain. Methods used in frequency domain, such as level set, can also be used in time domain [58]. In a later chapter, a boundary theorem is proven for the time domain that can be seen as analogous to the Kharitonov theorem for frequency domain.

2.4 Summary

In this chapter, topics related to simulation of dynamic systems with uncertain parameter have been reviewed. Figure-2.9 summaries the content of this review.

Table 2.2 shows the applicability of the different time domain approaches reviewed and the goal motivated by the review. Some conclusions can be drawn from this review as follows:

1. Simulation of systems with uncertain parameters remains a challenging problem, particularly because of the high computational cost.

2. Knowing the boundary of the output of an uncertain system has a wide

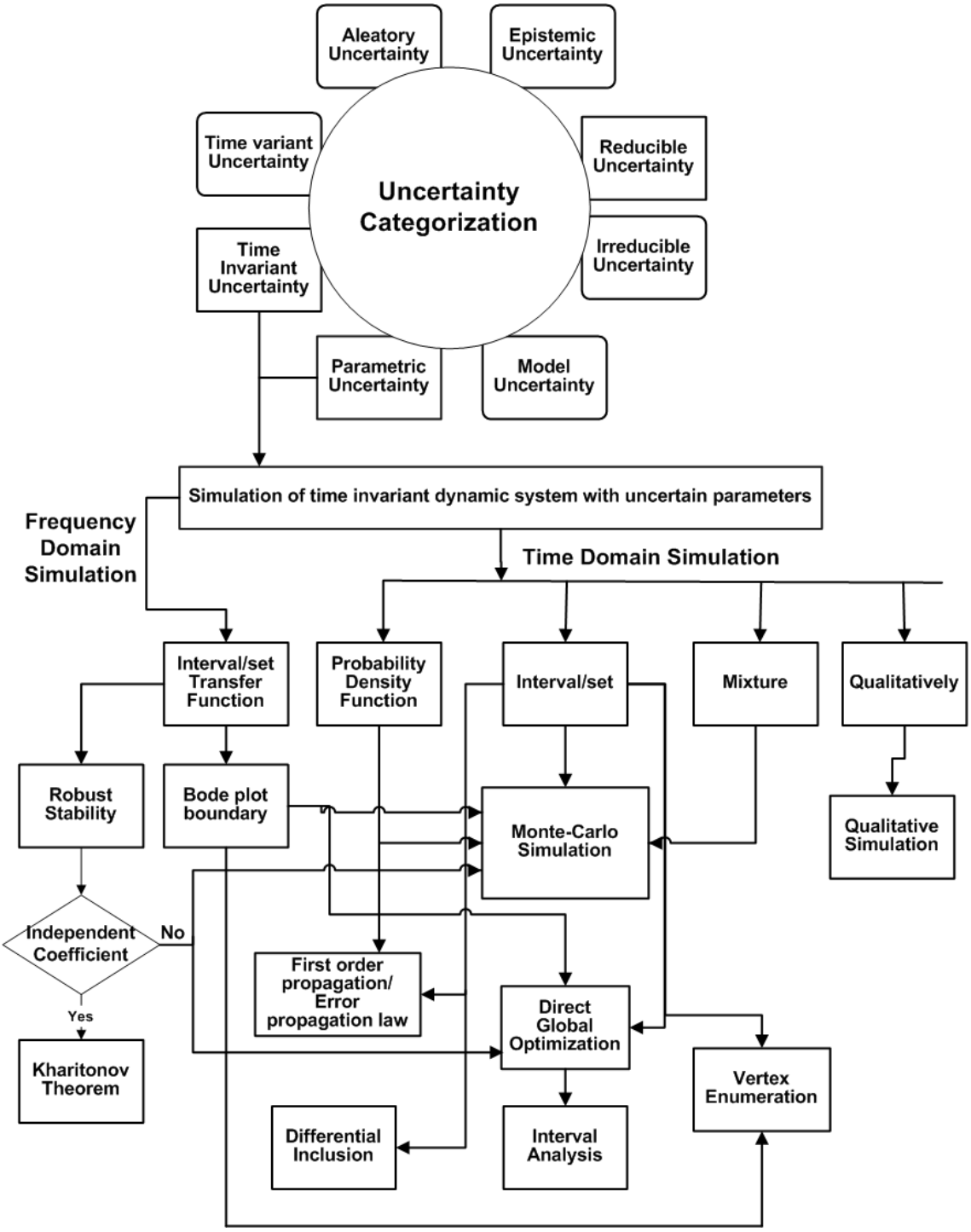


Figure 2.9: Summary of the review

range of applications.

This review motivated the following research goals:

1. Develop a numerical method to simulate (in time domain) parametric uncertain, time invariant, nonlinear dynamic systems, i.e., find the boundary/envelopes of the output variables.

2. The method should calculate boundary/envelope of the output accurately and efficiently. Methods to improve accuracy and reduce computational cost should be pursued.

3. The method should be able to deal with large uncertainties, and to handle a reasonable number (< 10) of uncertain parameters.

4. The method should be easily implemented and existing simulation tools for determined systems should be easily integrated with this method.

Table 2.2: Evaluation of reviewed time domain approaches

Evaluation Criteria	MCS	FOP/ EPL	DGO	IA	DI	QS	VE	New Approach
Computational Cost	High	Low	High	High	Low	Low	Low	medium
Solution quality	High	Depends	High/ Depends	High/ Depends	Low	Low	Low	High/ Depends
Uncertainty Degree	Large	Small	Large	Large	Large	large	Large	Large
Interval	Yes	Yes	yes	yes	Yes	Yes	Yes	yes
Statistic/pdf	Yes	Yes	No	No	No	No	No	yes
Implementation	Easy	Medium	Medium	Difficult	Difficult	Difficult	Easy	easy/ medium
Time invariant	Yes	Yes	Yes	Yes	No?	N/A	Yes	Yes

Chapter 3

The Response Surface Methodology

In this chapter, we describe in detail a proposed methodology for simulation of parametric uncertain system. The goal of this approach is to resolve the trade-off between efficiency and accuracy. To reach this goal, the computational cost of the problem is first analyzed and then the basic idea of the proposed approach is presented. The basic idea is to study an approximation or a surrogate of the original system model, instead of the original system. To construct the surrogate model, the response surface method (RSM) is employed. Since the optimization problem to be solved is generally non-convex, there may be multiple local optima. Conventional RSM using polynomials which provides global approximation is not able to deal with the non-convex problem. Thus a local approximation approach called Moving Least Square (MLS) is used for response surface construction. For more complicated systems, a gradient enhanced moving least square (GEMLS) response surface method is used to solve the global optimization problem more efficiently. This method takes advantage of the fact that parametric sensitivity of an ODE system can be calculated as a by-product when solving the original system with less computational cost. With the help of sensitivity information, the number of samples needed to construct the response surfaces is further decreased, and the

quality of the response surface can be improved. Furthermore, global sensitivity analysis for monotonic testing to further reduce the number of samples is introduced. Finally, numerical examples are provided to show the applicability and effectiveness of the proposed method.

3.1 Computational Cost Analysis

From Chapter 2 we know that the simulation of parametric uncertain dynamic systems can be posed as an optimization problem, as in Eq. 2.6, which is rewritten here for convenience,

$$\begin{aligned} \underline{\mathbf{y}} &= \underset{t\in[0,T]}{\cup} \min\left(\mathbf{y}(t)\right) = \underset{t\in[0,T]}{\cup} \min\left(\mathbf{g}(\mathbf{x}(t),\mathbf{u})\right) \\ \overline{\mathbf{y}} &= \underset{t\in[0,T]}{\cup} \max\left(\mathbf{y}(t)\right) = \underset{t\in[0,T]}{\cup} \max\left(\mathbf{g}(\mathbf{x}(t),\mathbf{u})\right) \\ \text{s.t.} &\begin{cases} \mathbf{x}(t) = \mathbf{x}_0 + \int_0^t \mathbf{f}(\mathbf{x},\mathbf{u},\boldsymbol{\lambda},\tau)d\tau \\ \boldsymbol{\lambda} \in [\underline{\boldsymbol{\lambda}},\overline{\boldsymbol{\lambda}}], \mathbf{x_0} \in [\underline{\mathbf{x_0}},\overline{\mathbf{x_0}}] \end{cases} \end{aligned} \tag{3.1}$$

This formulation shows that the solution requires a method that is a computationally expensive task. Specifically, this arises because of the following aspects:

1. In order to solve Eq. (3.1), the objective function,

$$\mathcal{O} = \begin{cases} \min(\mathbf{g}(\mathbf{x}(t))) \\ \max(\mathbf{g}(\mathbf{x}(t))) \end{cases} \tag{3.2}$$

must be evaluated at each time step. This means that numerical simulations ($\mathbf{x}(t) = \mathbf{x}_0 + \int_0^t \mathbf{f}(\mathbf{x},\boldsymbol{\lambda},\mathbf{u},\tau)d\tau$) need to be embedded into the optimization routine. Since Eq. (3.1) is not generally a convex problem,

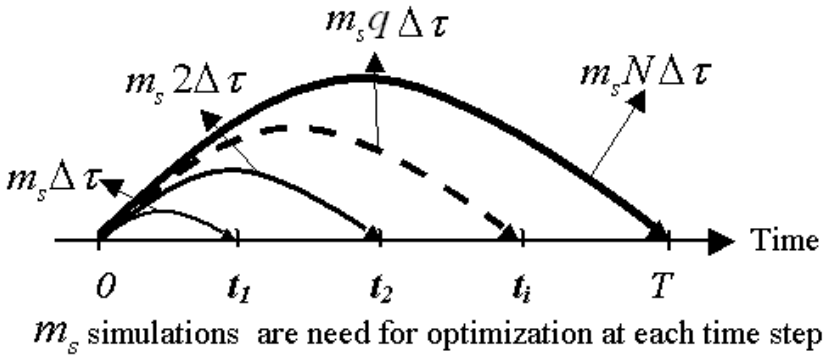


Figure 3.1: Non-recursive simulation

it may have a lot of local maxima. To find the global maxima, simulations may be performed many times. If simulation of the system is already a computationally expensive task, which is often true for practical engineering systems (such as a nonlinear, stiff systems or if the time to be simulated is very long), solving Eq. (3.1) will be even more expensive.

2. Furthermore, Eq. (3.1) implies that the simulations embedded in the optimization routine must always start from $t = 0$, due to the nature of time-invariant dynamic system. This is a non-recursive simulation that is illustrated by Figure-3.1. The advantage of non-recursive simulation is that it can guarantee that the approximation error does not grow with time. On the other hand, it will take longer and longer to simulate the system as time t increases.

The computational cost of solving Eq. (3.1) can be approximated as follows:

Assume the CPU time to simulate the system is linear with the time span $[0, T]$ to be simulated. The CPU time to simulate the Δt time span is denoted as $\Delta\tau$. We also assume that, for each time step, m_s simulations (evaluation of objective function $\mathcal{O}$) are needed in order to solve Eq. (3.1). The total time step is $N = \dfrac{T}{\Delta t}$. Then, the total CPU time is,

$$\begin{aligned} T_{cpu\_flow} &= m_s(\Delta\tau + 2\Delta\tau + \cdots q\Delta\tau + \cdots N\Delta\tau) \\ &= m_s\Delta\tau\frac{N(N+1)}{2} \end{aligned} \tag{3.3}$$

This shows that CPU time grows quadratically with N.

To reduce the computational cost of solving Equation 3.1, this non-recursive simulation is often approximated by a recursive one as,

$$\begin{aligned} &\underset{\sim}{\mathbf{y}} = \bigcup_{t\in[0,T]} \min\left(\hat{\mathbf{y}}(\mathbf{t})\right) = \bigcup_{t\in[0,T]} \min\left(\mathbf{g}(\hat{\mathbf{x}}(\mathbf{t}), \mathbf{u})\right) \\ &\tilde{\mathbf{y}} = \bigcup_{t\in[0,T]} \max\left(\hat{\mathbf{y}}(\mathbf{t})\right) = \bigcup_{t\in[0,T]} \max\left(\mathbf{g}(\hat{\mathbf{x}}(\mathbf{t}), \mathbf{u})\right) \\ &\text{s.t.} \begin{cases} \hat{\mathbf{x}}(t) = \hat{\mathbf{x}}(t-1) + \int_{t-1}^{t} \mathbf{f}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, \tau) d\tau \\ \boldsymbol{\lambda} \in [\underline{\boldsymbol{\lambda}}, \overline{\boldsymbol{\lambda}}], \mathbf{x}_0 \in [\underline{\mathbf{x_0}}, \overline{\mathbf{x_0}}] \end{cases} \end{aligned} \tag{3.4}$$

where $\hat{\mathbf{x}}$ is the approximation of state variable $\mathbf{x}$ at time t. Geometrically speaking, $\hat{\mathbf{x}}$ is the hyper rectangle bounded by $\underset{\sim}{\mathbf{y}}$ and $\tilde{\mathbf{y}}$, as shown in Figure-3.2. Eq. (3.4) can reduce the computational cost by converting a non-recursive simulation to a recursive one, and thus there is no need to simulate the system from time zero. This transformation leads to the following severe drawbacks:

1. Eq. (3.4) is a conservative approximation of Eq. (3.1), because the simulation scheme is assuming the system is time-varying. Generally, the

envelope given by Eq. (3.4) will be much wider than that given by Eq. (3.1) [68].

2. The approximation at time step t_{k-1} will accumulate to the next time step,t_k, by Eq. (3.4), which is called *wrapping effect*, as illustrated in Figure-3.2 [4]. In Eq. (3.4), the system's state variable, $\mathbf{x}(\mathbf{t})$, is rep-

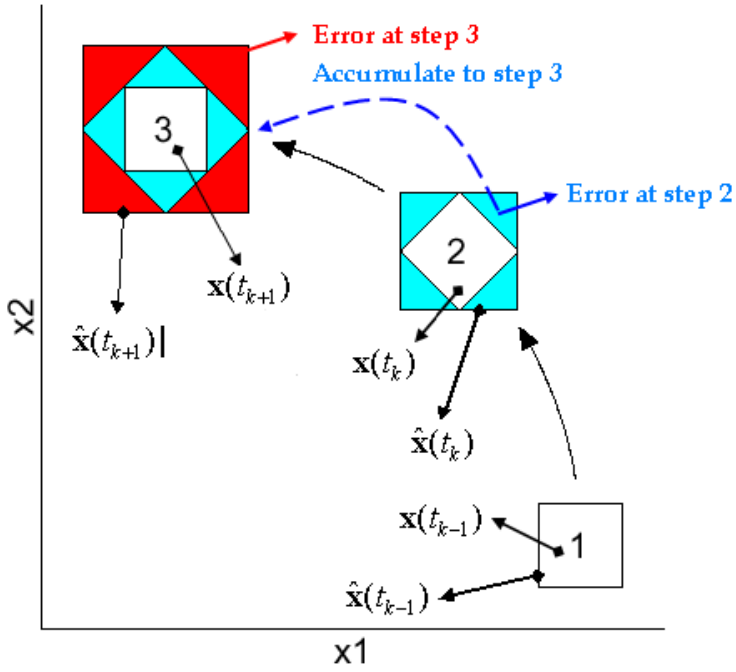


Figure 3.2: The wrapping effect, adapted from [4]

resented by a hyper rectangle. However, it may be that the system's state does not evolve into another hypercube at the next time point. In Figure-3.2, an example is shown in which there are two state variables and, therefore, the state is approximated as a rectangle. This rectangle evolves to a rhombus (it could evolve to any figure in two dimensions) in

the following time step. As the value of each variable is still expressed with an interval, the new state is represented with a new rectangle that includes all possible states (the rhombus) but also spurious states, shown in shadow in the figure. It can be found that error in a previous time step will accumulate into the next step.

3. Because of the wrapping effect, the envelopes of a stable time variant system may be unstable if Eq. (3.4) is used to simplify Eq. (3.1) [21].

To solve the above problems, a method called *sliding window* was proposed [4, 21, 64]. The idea is to assume that the current state is only influenced by a limited number of L previous states, due to the dynamics of the system. L is called the window length. It simplifies the simulation to,

$$\underset{L}{\hat{\mathbf{x}}}(t_k) = \begin{cases} \mathbf{x}_0 + \int_0^{t_k} \mathbf{f}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, \tau) d\tau & 0 \leq k \leq L \\ \hat{\mathbf{x}}(t_{k-L}) + \int_{t_{k-L}}^{t_k} \mathbf{f}(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, \tau) d\tau & k > L \end{cases} \tag{3.5}$$

Eq. (3.5) can be seen as a trade-off between pure time invariant simulation of Eq. (3.1), where $L = \inf$, and a pure time variant simulation of Eq. (3.4), where $L = 1$. For linear parametric uncertain system, there exists a minimum window length L_{min} that will guarantee that a stable simulation will be acquired if the original linear parametric uncertain system is stable [68].

The sliding window method raised more questions than it answered. For example, how to determine L_{min}, what if the system is nonlinear, etc. Also new methods should be developed to reduce the computational cost.

3.2 Surrogate Model for Optimization: The Response Surface Method

As concluded in the previous section, the major computational burden of simulation in parametric uncertain systems is that evaluation of the objective functions is too expensive, due to system dynamics and the non-recursive character of time invariant systems. When optimization of the original objective function is too expensive, an approximation of the original function or a surrogate could be used [7, 8]. Evaluation of the surrogate takes less time than evaluation of the original objective functions. One method to build such surrogate is called Response Surface Method (RSM), which has long been used for design, optimization and other applications [11, 31, 60].

RSM can be defined as a collection of statistical and mathematical techniques useful for developing, improving, and optimizing processes. The most extensive use of RSM can be found in the industrial applications, in situations where several input variables influence some performance measure, called the response, in a way that is difficult or impossible to describe with a rigorous mathematical formulation. In these situations it might be possible to derive an expression for the performance measure based on the response values obtained from experiments at some particular combination of the input variables. The expression of the performance measure obtained through experiments is called response surface (RS) [60]. With the development of the computing technology, the experiments cam be done by computer simulations.

The response surface method approximates an unknown objective func-

tion O, with an appropriate empirical model, $\hat{O}$, so that $O(\mathbf{x}) = \hat{O}(\mathbf{x}) + \varepsilon$. The empirical model $\hat{O}$ is called the response surface of O. Once the response surface, $\hat{O}$, is built, it can be used to replace the original function, O, for optimization or other purposes.

3.2.1 General Steps of Using RSM for Simulation of Parametric Uncertain Systems

For the simulation of parametric uncertain system, the objective function Eq. (3.2) is a function of the uncertain parameter $\boldsymbol{\lambda}$ and uncertain initial condition $\mathbf{x}_0$, at a given time t. It can be written as: $\hat{\mathcal{O}}(\boldsymbol{\chi}, t)$, where $\chi = [\boldsymbol{\lambda}, \boldsymbol{x}_0], \boldsymbol{\chi} \in \mathbb{R}^n$ is the total uncertain parameter space to be searched.

A response surface $\hat{\mathcal{O}}$ to approximate $\mathcal{O}$ based on just a few samples can be constructed and used for optimization. The general steps of using RSM for parametric uncertain system simulation are:

1. Take n_s sample points from $\boldsymbol{\chi}$, denoted as $\boldsymbol{\chi}_s^i$, $i = 1, 2, ..., n_s$, where $\boldsymbol{\chi}_s^i = [\boldsymbol{\lambda}_s^i, \mathbf{x}_{s0}^i]$

2. Integrate $\mathbf{x}_s^i(t) = \mathbf{x}_{s0}^i + \int_0^T \mathbf{f}(\mathbf{x}, \boldsymbol{\lambda}_s^i, \mathbf{u}, t)dt$ to get n_s trajectories of state variables in the time span $[0, T]$.

3. At q^{th} time step $t_q, q = 1, 2, ..., N$, do the following:

    a. Calculate $\mathcal{O}(\chi_s^i, t_q) = \mathbf{g}(\mathbf{x}_s^i, t_q)$, $i = 1, 2, ..., n_s$

    b. Construct the response surfaces $\hat{\mathcal{O}}(\chi_s^i, t_q)$ based on $\mathcal{O}(\chi_s^i, t_q), i = 1, 2, ..., n_s$.

c. Solve Eq. (3.1) using $\hat{\mathcal{O}}$ as the objective function, instead of using $\mathcal{O}$.

The above steps are illustrated in Figure-3.3. By using RSM, only n_s simulations from time 0 to T are needed. Since each simulation is still started from time 0, the error will not accumulate from step to step. (i.e. no wrapping effect with this approach). In step 3-c, any appropriate global optimization technique can be used, such as genetic algorithms or differential evolution [75]; or simply divide $\boldsymbol{\chi}$ into m_{rsm} grids, each grid is small enough and one cane exclusively evaluate all the values of $\hat{\mathcal{O}}(\chi, t)$ at each grid and find the minimum and maximum, since $\hat{\mathcal{O}}(\chi, t)$ is cheap to evaluate.

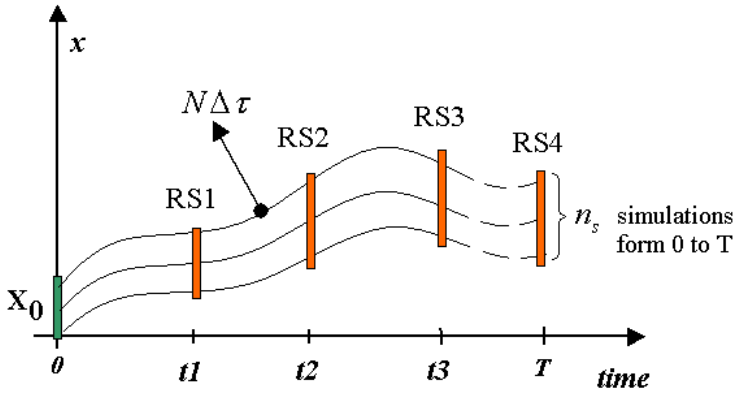


Figure 3.3: Using RSM for simulation of parametric uncertain systems

3.2.2 Time Saving by Using RSM

The total CPU time used by RSM for parametric uncertain system simulation can be approximated as follows: assume n_s samples are needed and

at each time step t_q, m_{rsm} evaluations of the response surface (evaluation of $\hat{\mathcal{O}}$) are needed. The time for each evaluation is τ', so the total CPU time is,

$$T_{cpu\_rsm} = n_s N\Delta\tau + m_{rsm} N\tau'. \tag{3.6}$$

From Eq. (3.3),

$$\zeta = \frac{T_{cpu\_flow}}{T_{cpu\_rsm}} = \frac{m_s\Delta\tau\dfrac{N}{2}(N+1)}{n_s N\Delta\tau + N m_{rsm}\tau'}. \tag{3.7}$$

To simplify this equation, assume $\dfrac{N(N+1)}{2} \approx \dfrac{N^2}{2}, \dfrac{m_{rsm}}{m_s} = \alpha$. Also, assume that $m_s = n_s$, which implies that if a certain optimization technique used to solve Eq. (3.1) can find the optima of $\mathcal{O}$ directly by m_s evaluations of $\mathcal{O}$, then m_s samples can be used to construct a response surface to find the optima of $\hat{\mathcal{O}}$ as well. Let $\tau = N\Delta\tau$, which is the CPU time to simulate system from 0 to T. The cost ratio between calculating the real objective function,$\mathcal{O}$, and the surrogate, $\hat{\mathcal{O}}$, is $\beta = \dfrac{\tau}{\tau'}$. Notice that τ' can be treated as a constant, while τ will increases if T increases. With these assumptions, Eq. (3.7) can be simplified to,

$$\zeta = \frac{T_{cpu\_flow}}{T_{cpu\_rsm}} \approx \frac{m_s\tau\dfrac{N}{2}}{m_s\tau + \dfrac{\alpha}{\beta}m_s\tau} = \frac{N}{2\left(1+\dfrac{\alpha}{\beta}\right)}. \tag{3.8}$$

When $\zeta >> 1$, RSM will be a better choice for parametric uncertain system simulation. Eq. (3.8) suggests that RSM can be effectively applied to problems when:

1. N is large, which means time span is long or integration step is small.

2. Generally, large values of α are needed to get accurate results. This means that the search domain should be divided into smaller regions when using RSM to do an exclusive search, leading to increased CPU time. However, for systems where simulation is slow, β will easily be large enough to cancel the effect of increasing α.

From the above analysis, it can be found that RSM will reduce the computational cost for systems whose simulation is very time consuming, while keeping the time invariant character and avoiding the wrapping effect. Compared to the sliding window method, RSM can be applied to general nonlinear systems and there is no need to select window length, L.

3.3 Total Least Square Response Surface

The general steps to construct a response surfaces $\hat{O}$ of a function $O(\boldsymbol{x})$ are:

1. Select input variables called *sample points*, which are denoted as, $\mathbf{x}_s^i \in \mathbb{R}^n, i = 1, 2, ..., n_s$, where n_s is the number of samples points. This step is called *Design of Experiment* (DOE).

2. Acquire $O_s^i = O(\mathbf{x}_s^i)$, which are the corresponding outputs (observations) of the selected sample points by experiments or simulation. The pairing of a sample point $\mathbf{x}_s^i$ and its observation O_s^i is called a *sample*. The process of acquiring a sample is called *sampling*.

3. Fit $\hat{O}$ to the samples by minimizing certain error function, $E = E(O_s^i - \hat{O}_s^i)$.

Different mathematical forms of $\hat{O}$ and E lead to different response surface methods.

The most widely used response surface method is called the total least square (TLS) method, which uses polynomials to from $\hat{O}$. For example, a two variable model represented by a quadratic polynomial can be written as,

$$\hat{O} = a_1 + a_2 x_1 + a_3 x_2 + a_4 x_1^2 + a_5 x_2^2 + a_6 x_1 x_2.$$

In matrix form, we have,

$$\hat{O} = \sum_{k=1}^{n_p} p_k(\mathbf{x}) a_k = \mathbf{pa}, \tag{3.9}$$

where $p_k(\mathbf{x})$ is called the *basis function*, and n_p is the number of terms in the basis. Generally, the basis functions are polynomials, as in this example. For a 2^{nd} order approximation, the basis function $\mathbf{p}(\mathbf{x})$ can be written as,

$$\mathbf{p}(\mathbf{x}) = \mathbf{p}(x_1, x_2) = \left[\begin{array}{cccccc} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \end{array}\right] = \left[\begin{array}{cccccc} 1 & x_1 & x_2 & x_1 x_2 & x_1^2 & x_2^2 \end{array}\right],$$

where $\mathbf{a}$ is the coefficient vector, and $\mathbf{a}^T = \left[\begin{array}{cccc} a_0 & a_1 & \cdots & a_{n_p-1} \end{array}\right]$. In this example, $n_p = 6$.

The coefficient of the polynomials are determined by minimizing the total square error defined as,

$$E_T = \sum_{i=1}^{n_s} \varepsilon_i^2 = \sum_{i=1}^{n_s} (O_s^i - \tilde{O}_s^i)^2 = \sum_{i=1}^{n_s} \left[\sum_{k=1}^{n_p} p_k(\mathbf{x}) a_k - O(\mathbf{x}_s^i) \right]^2. \tag{3.10}$$

In matrix form, Eq. (3.10) can be written as,

$$E_T = (\mathbf{P}_s\mathbf{a} - \mathbf{O}_s)^T(\mathbf{P}_s\mathbf{a} - \mathbf{O}_s), \tag{3.11}$$

where,

$$\mathbf{O}_s^T = \left[\begin{array}{cccc} O\left(\mathbf{x}_s^1\right) & O\left(\mathbf{x}_s^2\right) & \cdots & O\left(\mathbf{x}_s^{n_s}\right) \end{array} \right]_{1\times n_s}, \tag{3.12}$$

is the observation matrix, and,

$$\mathbf{P}_s = \left[\begin{array}{cccc} p_1(\mathbf{x}_s^1) & p_2(\mathbf{x}_s^1) & \cdots & p_{n_p}(\mathbf{x}_s^1) \\ p_1(\mathbf{x}_s^2) & p_2(\mathbf{x}_s^2) & \cdots & p_{n_p}(\mathbf{x}_s^2) \\ \vdots & \vdots & \cdots & \vdots \\ p_1(\mathbf{x}_s^{n_s}) & p_2(\mathbf{x}_s^{n_s}) & \cdots & p_{n_p}(\mathbf{x}_s^{n_s}) \end{array} \right]_{n_s\times n_p}, \tag{3.13}$$

is the basis function matrix.

To minimize the total least square error, E_T, require,

$$\left.\frac{\partial E_T}{\partial \mathbf{a}}\right| = 0 = -2\mathbf{P}_s^T\mathbf{O}_s + 2\mathbf{P}_s^T\mathbf{P}_s\mathbf{a}(\mathbf{x}) \ . \tag{3.14}$$

Let,

$$\begin{aligned} \mathbf{A}_{[n_p\times n_p]} &= \mathbf{P}_s^T\mathbf{P}_s \\ \mathbf{B}_{[n_p\times 1]} &= \mathbf{P}_s^T\mathbf{O}_s \end{aligned} \tag{3.15}$$

then Eq. (3.14) simplifies to,

$$\mathbf{a}(\mathbf{x}) = \left[\begin{array}{cccc} a_1(\mathbf{x}) & a_1(\mathbf{x}) & \cdots & a_{n_p}(\mathbf{x}) \end{array} \right] = \mathbf{A}^{-1}\mathbf{B} \tag{3.16}$$

From the above procedures we can find that the response surface model constructed by total least square method is a global model. That is, once the coefficient vector $\mathbf{a}$ is determined by the sampled data, it can be used to approximate the objective function value at any points in the entire domain

where O is defined. The advantage of a global model is that the computational cost is low since one model can be used for all points where function values are to be estimated. However, since the base function in the model is a low order polynomial, such a global model can only be used to approximate functions with very few local maxima. If a function with multi-local maxima is to be approximated, the approximation error will be increased. To avoid this, the order of the polynomial has to be increased, with corresponding computational cost. Because of this, global approximation is often used to approximate a function in a very small domain where only one local maxima exists.

3.4 Local Approximation for Global Optimization

Generally, Eq. (3.1) is not convex and there may be several local maxima. Conventional response surfaces that use a single quadratic or cubic polynomial to represent the entire domain of the target function are not able to deal with objective functions having multiple local optima [32, 48]. As such, local approximation methods such as Kriging [17, 41] and moving least square [52, 70] should be used for such optimization problem. Kriging is an interpolation method that originated in geostatistics and uses properties of the spatial correlation among the data samples. Moving least square is a weighted least square method such that the weights are functions of the location of approximation. This method has been used for optimization with up to ten variables [48]. Compared with Kriging, moving least square is found to be more accurate and computationally efficient [48]. In this section, we will introduce

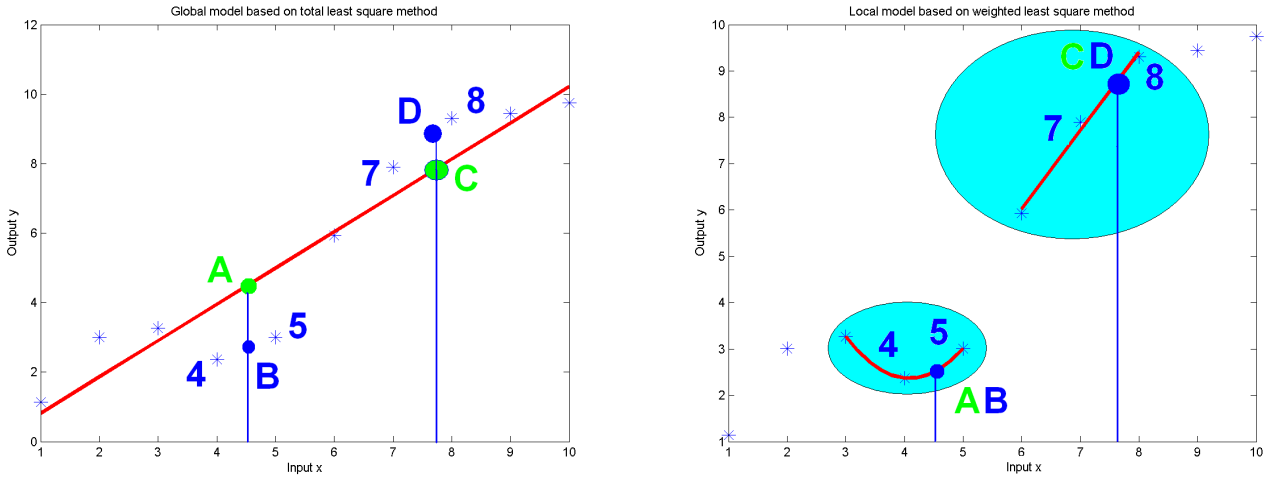
gradient enhanced moving least square for response surface construction, and the principles apply to the Kriging method as well.

3.4.1 Moving Least Square (MLS) method

The moving least squares (MLS) method is a localized surface reconstructing technique which was introduced in [70] by Shepard. Recently, it has become widely used in related engineering areas by virtue of its approximation power [52]. Such as approximation for time-dependent PDEs [27], analysis of metal forming processes by the flow formulation [33], function optimization [37], etc.

MLS intrinsically is a weighted least square method, in which the weighting is a function of the location of the point to be approximated. Figure-3.4 can be used to interpret the concept of moving least square [47]. Figure-3.4(a) shows a line generated by conventional total least square method. In this method, sample points are equally weighted and their contributions to the resulted model are the same. As a consequence, one model with fixed coefficients is used to predict the output no matter what value of the input variables. As shown in Figure-3.4(a), assume we use this model to approximate the output y when input $x = 4.6$, since the original function is highly nonlinear around $x = 4.6$ and the predicted point A is far away from the real point B. At the location where $x = 7.5$, the original function is quite linear, but since the model (solid line) is also influenced by point-4 and point-5, the predicted value is still not good.

Figure-3.4(b) shows that only the samples in the neighborhood of the estimated points are used to construct the response surface. Each point is weighted according to its distance to the estimated point. Consequently, a local approximation of the function will be constructed for each estimated point, with different coefficients that depend on its location. The computing cost of moving least square surface is higher than the total least square but the accuracy is also improved dramatically. The procedure to construct a moving



(a) Global model by total least square method

(b) Local model by Moving least square method

Figure 3.4: The principle of Moving Least Square(MLS)method

least square response surface at a certain estimated point is as follows:

Assume an unknown function $O(\mathbf{x})$ defined in the domain Ω is to be approximated by, $\hat{O}(\mathbf{x}_s)$ as $O(\mathbf{x}_s) = \hat{O}(\mathbf{x}_s) + \varepsilon$, where $\mathbf{x} = [x_1\ x_2\ ...\ x_n]$ and ε is the approximation error. Suppose we have n_s sample points, $\mathbf{x}_s^i \in \mathbf{x}, i = 1, 2, \ldots, n_s$, and the observations, $O(\mathbf{x}_s^i), i = 1, 2, ..., n_s$. First, the local ap-

proximation $\hat{O}(\mathbf{x})$ of $O(\mathbf{x})$ at each position $\mathbf{x}$ in the domain Ω is defined as a linear combination of a set of independent functions,

$$\hat{O}(\mathbf{x}) = \sum_{k=1}^{n_p} p_k(\mathbf{x}) a_k(\mathbf{x}) = \mathbf{p}(\mathbf{x})\mathbf{a}(\mathbf{x}), \tag{3.17}$$

where $p_k(\mathbf{x}), k = 1, 2, ..., n_p$ is the same as defined in Eq. (3.9). Comparing Eq. (3.9) to Eq. (3.17), the only difference is that for the moving least square method the coefficient $\mathbf{a}$ is a function of the location where the original function is to be approximated, which makes it a localized approximation rather than a global approximation.

Similar to the total least square method, the coefficients, $a_k(\mathbf{x})$, can be obtained by minimizing the moving least square error, which is defined as,

$$E_M = \sum_{i=1}^{n_s} w(\mathbf{x} - \mathbf{x}_s^i) \left[\mathbf{p}(\mathbf{x}_s^i)\mathbf{a}(\mathbf{x}) - O(\mathbf{x}_s^i)\right]^2. \tag{3.18}$$

In matrix form, Eq. (3.18) can be written as,

$$E_M = (\mathbf{P}_s\mathbf{a}(\mathbf{x}) - \mathbf{O}_s)^T \mathbf{W}(\mathbf{x})(\mathbf{P}_s\mathbf{a}(\mathbf{x}) - \mathbf{O}_s). \tag{3.19}$$

The definitions of $\mathbf{O}_s$ and $\mathbf{P}_s$ are the same as in total least square method and the weight function matrix is,

$$\mathbf{W}(\mathbf{x}) = \begin{bmatrix} w(\mathbf{x} - \mathbf{x}_s^1) & & & \\ & w(\mathbf{x} - \mathbf{x}_s^2) & & \\ & & \ddots & \\ & & & w(\mathbf{x} - \mathbf{x}_s^{n_s}) \end{bmatrix}_{n_s \times n_s}. \tag{3.20}$$

This equation shows that the moving least square method is intrinsically a weighted square method in which the weight is a function of the location.

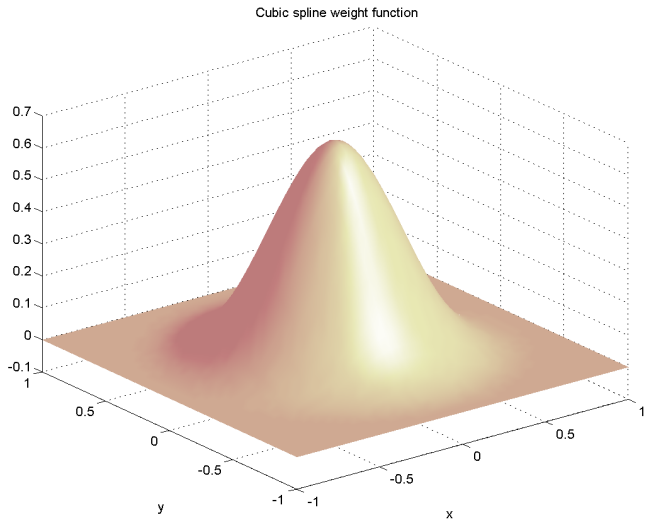


Figure 3.5: The cubic spline weight function

The weight function should be a compactly supported function centered at each sampling point. The cubic spline weight function can be used and it is defined as,

$$w(\mathbf{x}-\mathbf{x}_s^i) = \begin{cases} \frac{2}{3} - 4r^2 + 4r^3 & r < 0.5 \\ \frac{4}{3} - 4r + 4r^2 - \frac{4}{3}r^3 \ 0.5 & < r \leq 1 \\ 0 & r > 1 \end{cases}$$

where $r = \dfrac{\|\mathbf{x}-\mathbf{x}_s^i\|}{r_{\max}}$, and $r_{\max}$ is called the *radius of influence domain*. The value of r_{max} reflects how local the MLS will be. If r_{max} is large enough to cover the whole domain Ω, then MLS becomes a conventional TLS method. The shape of the weight function in 2D is shown in Figure-3.5.

To minimize the least square error E_m, take,

$$\left.\frac{\partial E_M}{\partial \mathbf{a}}\right| = 0 = -2\mathbf{P}_s^T\mathbf{W}(\mathbf{x})\mathbf{O}_s + 2\mathbf{P}_s^T\mathbf{W}(\mathbf{x})\mathbf{P}_s\mathbf{a}(\mathbf{x}) \ . \tag{3.21}$$

Let,

$$\begin{aligned} \mathbb{A}_{[n_p\times n_p]} &= \mathbf{P}_s^T\mathbf{W}(\mathbf{x})\mathbf{P}_s \\ \mathbb{B}_{[n_p\times 1]} &= \mathbf{P}_s^T\mathbf{W}(\mathbf{x})\mathbf{O}_s \end{aligned} \tag{3.22}$$

Then,

$$\mathbf{a}(\mathbf{x}) = \left[\ a_1(\mathbf{x}) \quad a_1(\mathbf{x}) \quad \cdots \quad a_{n_p}(\mathbf{x})\ \right] = \mathbb{A}^{-1}\mathbb{B} \tag{3.23}$$

Comparing Eq. (3.16) and Eq. (3.23), the similarity between MLS and TLS can be found. The condition for the above MLS procedure to work is that matrix $\mathbb{A}$ must be non-singular. This can be guaranteed by adjusting $r_{\max}$ for each estimated point $\mathbf{x}$ so that $\mathbf{rank}(\mathbf{W}(\mathbf{x})) > n_p$. This shows that the MLS approximation has a self-adaptive regulating ability for irregular sample point patterns. At the location where density of the sample points are low, $r_{\max}$ should be larger to includes enough sample points. At the point where density of sample points is high, $r_{\max}$ should be smaller so good local approximation can be acquired. Another attractive property of the MLS approximations is that the continuity of $\hat{O}(\mathbf{x})$ is related to the continuity of the weight function, $w(\mathbf{x} - \mathbf{x}_s^i)$. Hence, one can use a linear basis function to reproduce higher order continuous approximations by choosing a suitable weight function. In addition, the MLS approximation is not necessarily an interpolant, [1] but could be an interpolant by introducing singularity to the weight function and making

[1]Interpolant means the approximated function will go through the sample points

the weight of sample points infinity. For example, use $w'(\mathbf{x}-\mathbf{x}_s^i) = \dfrac{w(\mathbf{x}-\mathbf{x}_s^i)}{\|\mathbf{x}-\mathbf{x}_s^i\|^\alpha}$ as a weight function, where α is a positive even integer [37]. Interpolant MLS is denoted as IMLS in the following discussion.

Example 3.1 Consider a one-dimensional mathematical function approximated by using the proposed MLS and interpolant MLS (IMLS). The analytical function is:

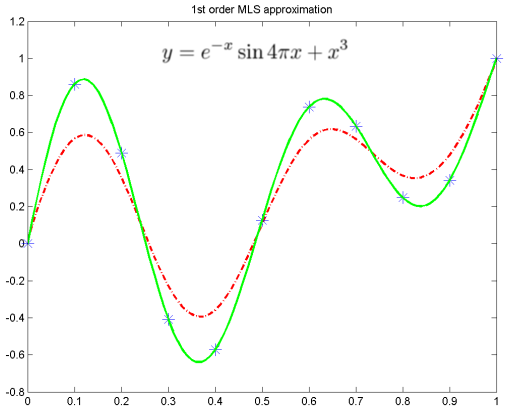
$$y = e^{-x}\sin 4\pi x + x^3$$

In this example, 11 equidistance sample points and their function values (observations) are used for reconstructing this function, as shown in Figure-3.6. Both the linear $\mathbf{p} = [\ 1 \quad x]$, and quadratic $\mathbf{p} = [\ 1 \quad x \quad x^2]$ basis function are used, and the spline cubic function is selected as the weight function for MLS/IMLS, and $\alpha = 2$ for the IMLS. The detailed approximation/interpolation results are given in Figure-3.6. For comparison purposes, results given by conventional least square with a 2nd order polynomial and 7th order polynomial are given in Figure-3.6(e) and 3.6(f).

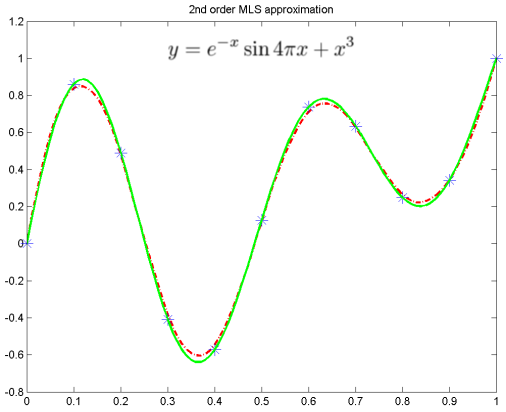
Example 3.2 A multiple variable example is shown in Figure-3.7. The original function is a Matlab function called *peaks*. The expression of this function is

$$z = 3(1-x)^2 e^{-x^2-(y-1)^2} - 10(\frac{x}{5} - x^3 - y^5)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2}$$

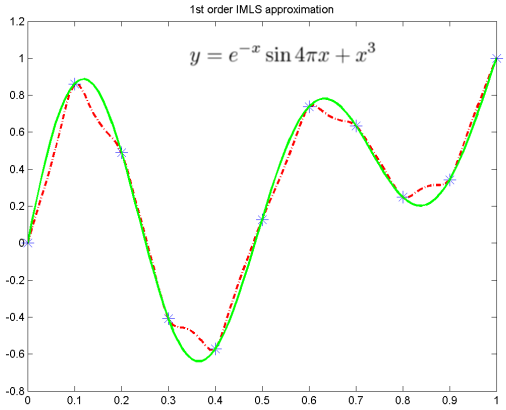
The function is reconstructed with 36 sample points. According to sampling theorem, 36 is very close to the minimum number of samples need to reconstruct the original function. The surface and contours of the original



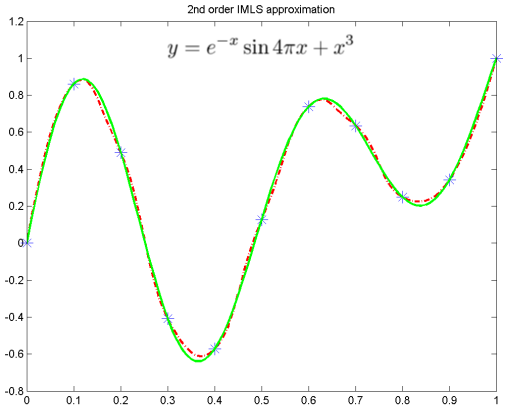
(a) 1st order MLS



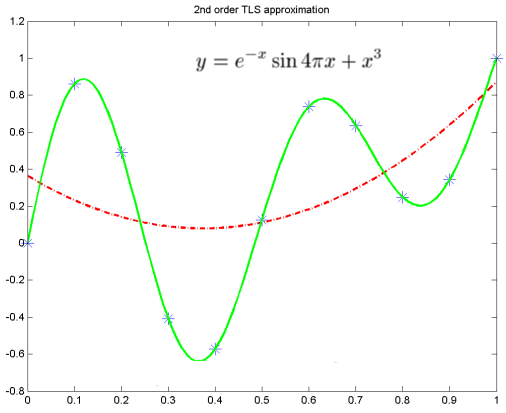
(b) 2nd order MLS



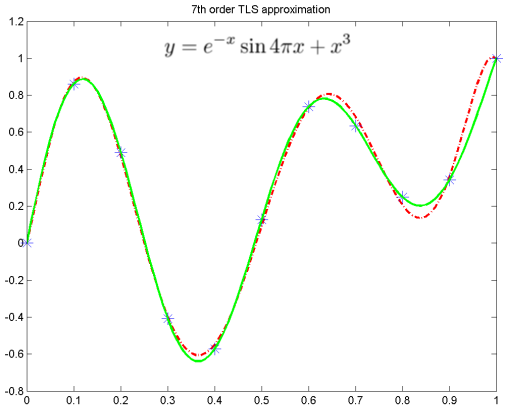
(c) 1st order IMLS



(d) 2nd order IMLS

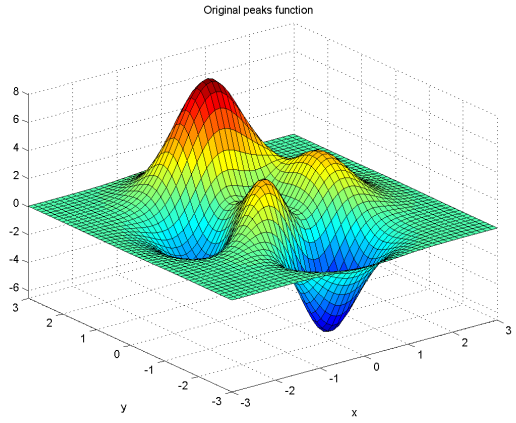


(e) 2nd TLS

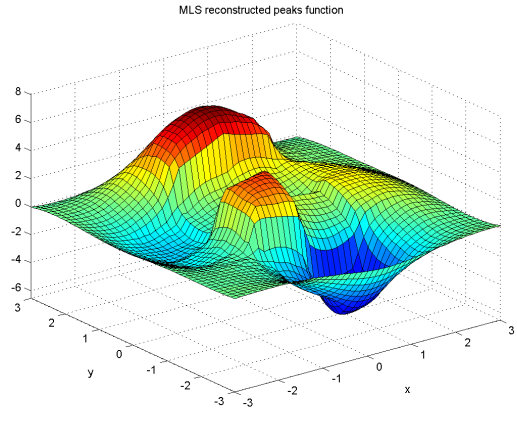


(f) 7th order TLS

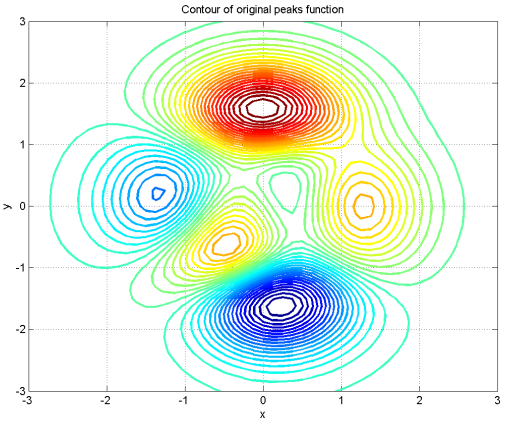
Figure 3.6: Moving least square example



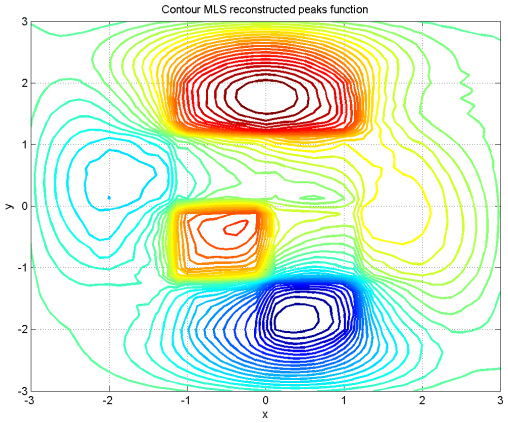
(a) Original Peaks



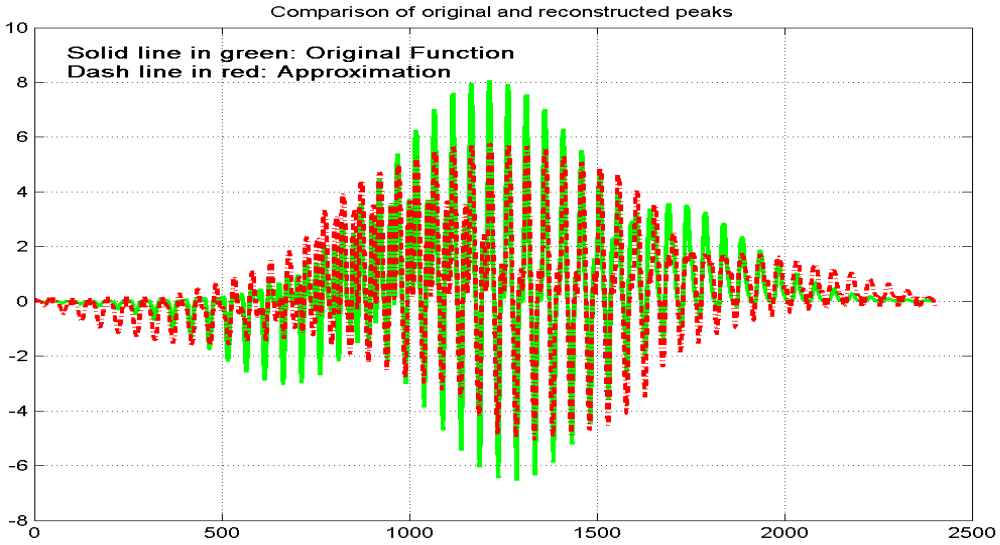
(b) MLS Reconstructed Peaks



(c) Contour of original Peaks



(d) Contour of MLS reconstructed



(e) Comparison of original and reconstructed peaks

Figure 3.7: Moving least square example: Peaks in Matlab

and reconstructed function are shown in Figure-3.7(a). The results show that the stationary points of the two functions are very close.

These two examples illustrate the following features of MLS:

1. MLS is a multi-variable regression method.

2. MLS has the ability of using low order basis functions to approximate highly nonlinear functions with multiple optima, and the continuity of the reconstructed function is related to the continuity of the weight function. Comparing Figure-3.6(a), 3.6(b) and Figure-3.6(e), 3.6(f), it is clear that MLS is good at local approximation. A 2^{nd} order MLS response surface is as good as a 7^{th} order total least square response surface.

3. Although the MLS approximation is not an interpolant, the stationary points (locations where the local optima exist) approximated by MLS are almost the same as their original true values, as shown in Figure-3.6(b), 3.6(d) and Figure-3.7(c), 3.7(d).

4. The IMLS method can force the reconstructed function to pass through the sample points exactly, by introducing singularities into the weight function. However, the shape of the reconstructed function is not as smooth as the one generated by MLS with same basis function. However, this can be improved by increasing the order of the basis function, as can be seen in Figure-3.6(c),3.6(d).

Taking into account the aforementioned observations, one therefore uses the MLS approach in the development of an efficient optimal technique. First the *locations* of the optima are found by using the moving least square response surface and then the corresponding optimal function values can be found by taking these locations as the input to the original function.

3.5 Reducing the Computational Effort

In order to construct the response surface, samples and their corresponding function values must be calculated, a process referred to as *sampling* before. Sampling contains two steps: 1) select the sample points $\mathbf{x}_s$ in the domain Ω, which is called *design of experiment (DOE)* and 2) calculate the corresponding observations O_s. For our problem, the 2^{nd} step means simulation of a determined system, and this can be time consuming. In order to reduce the computational cost to construct a response surface, the amount of sampling should be reduced, while maintaining the quality of the response surface. To reach this target, two approaches may be taken:

1. Find samples that represent the original function better
2. Get more information from one sample point and its corresponding function value.

In this section, both methods will be discussed, with focus on the latter approach.

3.5.1 Design of Experiment

An important aspect of RSM is the design of experiments, usually abbreviated as DOE. These strategies were originally developed for the model fitting of physical experiments, but can also be applied to numerical experiments. The objective of DOE is the selection of the points where the response should be evaluated or where the sample points should be sampled. The choice of the design of experiments can have a large influence on the accuracy of the approximation and the cost of constructing the response surface.

Most existing DOE methods are designed for conventional RSM which use 2^{nd} order polynomial and minimize the total least square error. They are based on a philosophy of sequential experimentation, with the objective of approximating the response with a low-order polynomial in a relatively small region of interest that contains the optimum solution. This implies that there may be only one local maximum. Thus, not all of these DOE methods apply to optimization problems with many local maxima. A detailed description of design of experiments theory can be found in [11, 60].

Some of the existing DOE methods can still be used directly for constructing the moving least square response surface. The can be categorized into so called *space filling design*, and these methods are:

1. *Full Factorial (FF) design or orthogonal lattice design.* A *factorial* experimental is an experiment strategy in which design variables are varied together, instead of one at a time. The lower and upper bounds of each

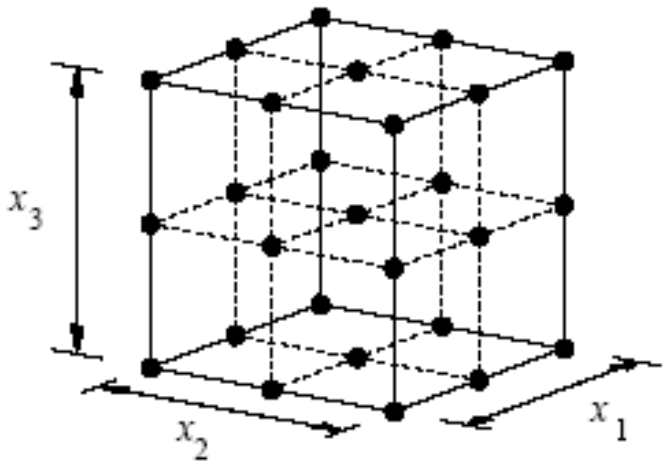


Figure 3.8: A 3 level factorial design

of n design variables in the optimization problem needs to be defined. The allowable range is then discretized at different levels. If each of the variables is defined at only the lower and upper bounds, the experiment is call a 2 level design. Similarly, if the midpoints are included, the design is called 3 level design. This technique can be simply understood as evenly sampling in the search space by dividing the search space into many grids, as shown in Figure-3.8.

2. *Monte Carlo or Random design.* The values of parameters are picked randomly from within their range (usually using a simple uniform probability distribution function–all values are equally likely).

3. *Latin Hypercube (LH) design.*

   Superior alternatives to both the full factorial and Monte Carlo scans ex-

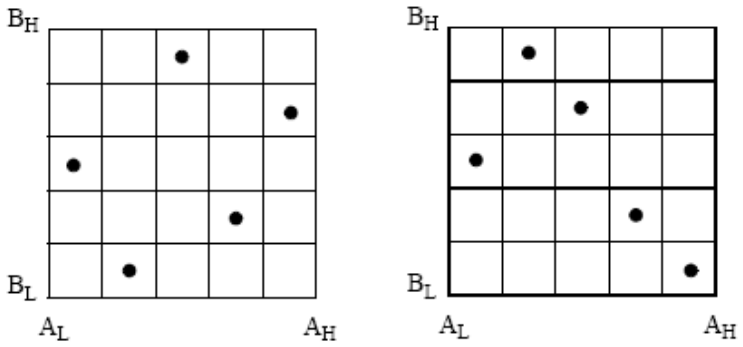


Figure 3.9: Latin hypercube design

ist: descriptive sampling methods such as the latin hypercube method [22, 57]. The basics of the latin hypercube (LH) method are demonstrated visually in Figure-3.9. Instead of selecting values of parameters randomly as is done in a Monte Carlo approach, values are selected descriptively. The resolution of all sampled parameters is the same as the total number of cases evaluated. In Figure-3.9, a problem with two parameters (A and B) has been sampled with a resolution of 5: five values of each parameter are sampled, but the same value of any one parameter is never tested twice. The cells within the hypercube are themselves chosen randomly.

When used for gathering statistical information, such as mean, variance, LH descriptive sampling is approximately 5 to 10 times more efficient than Monte Carlo sampling. In other words, the same level of accuracy can be gained in only 10% to 20% of the evaluations needed by the Monte Carlo (and presumably the FF) search methods.

3.5.2 Local Sensitivity Analysis of ODE systems

Sensitivity analysis (SA) is the study of how the variation in the output of a model can be apportioned, qualitatively or quantitatively to different sources of variation, and of how the given model depends upon the information input into it [67]. Sensitivity analysis can be divided into two large categories: local and global sensitivity analysis. Local sensitivity analysis focuses on estimation of model sensitivity to input and parameter variation in the vicinity of a sample point. This sensitivity is often characterized through gradients or partial derivatives at the sample point [42]. In other words, for a multivariable system, local sensitivity analysis methods refer to small changes of one parameter while other parameters are fixed. Global sensitivity analysis is a domain-wide sensitivity analysis that involves the study of the system behavior over the entire range of parameter variation, often taking the uncertainty in the parameter estimates into account. It refers to the effect of simultaneous parameter changes in a much larger amplitude.

In this section, we focus on the local sensitivity analysis of ODE systems. We concentrate on the numerical computing of the parametric sensitivity coefficients and explore the relationship between the ODE system and its adjacent sensitivity equations. We will show how sensitivity analysis will help to further reduce the computational cost by reducing the number of samples to construct a response surface.

Considering the state equations for a parametric uncertain system:

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{u}, t) \\ \mathbf{x}(0) = \mathbf{x_0} \end{cases} \tag{3.24}$$

Differentiating Eq. (3.24) with respect to $\boldsymbol{\lambda}$ yields,

$$\begin{cases} \dfrac{\partial \dot{\mathbf{x}}}{\partial \boldsymbol{\lambda}} = \dfrac{\partial \mathbf{f}}{\partial \mathbf{x}} \cdot \dfrac{\partial \mathbf{x}}{\partial \boldsymbol{\lambda}} + \dfrac{\partial \mathbf{f}}{\partial \boldsymbol{\lambda}} \\ \left.\dfrac{\partial \mathbf{x}}{\partial \boldsymbol{\lambda}}\right|_{t=0} = \mathbf{0} \end{cases} \tag{3.25}$$

Define, $\mathbf{S_\lambda} = \dfrac{\partial \mathbf{x}}{\partial \boldsymbol{\lambda}}$, and $\mathbf{J_x} = \dfrac{\partial \mathbf{f}}{\partial \mathbf{x}}$, $\mathbf{J_\lambda} = \dfrac{\partial \mathbf{f}}{\partial \boldsymbol{\lambda}}$, so Eq. (3.25) can be written as,

$$\begin{cases} \dot{\mathbf{S}}_\lambda = \mathbf{J_x} \cdot \mathbf{S}_\lambda + \mathbf{J}_\lambda \\ \mathbf{S}_\lambda(0) = \mathbf{0} \end{cases} \tag{3.26}$$

Differentiating Eq. (3.24) with respect to $\mathbf{x_0}$ yields,

$$\begin{cases} \dfrac{\partial \dot{\mathbf{x}}}{\partial \mathbf{x}_0} = \dfrac{\partial \mathbf{f}}{\partial \mathbf{x}} \cdot \dfrac{\partial \mathbf{x}}{\partial \mathbf{x}_0} \\ \left.\dfrac{\partial \mathbf{x}}{\partial \mathbf{x}_0}\right|_{t=0} = \mathbf{I} \end{cases} \tag{3.27}$$

Define $\mathbf{S_0} = \dfrac{\partial \mathbf{x}}{\partial \mathbf{x_0}}$, Eq. (3.27) can be written as:

$$\begin{cases} \dot{\mathbf{S}}_\mathbf{0} = \mathbf{J_x} \cdot \mathbf{S_0} \\ \mathbf{S_0}(0) = \mathbf{0} \end{cases} \tag{3.28}$$

Eq. (3.26) and Eq. (3.28) are called sensitivity equations. Here $\mathbf{J_x}$ is a $n \times n$ matrix that contains derivatives of the right-hand side of the differential equation with respect to the system variables and is called the Jacobian matrix. $\mathbf{J_\lambda}$ is a $n \times m$ matrix that contains derivatives of the right-hand side of the differential equation with respect to the system parameters and is called the

parametric Jacobian matrix. $\mathbf{S_\lambda}$ is a $n \times m$ matrix that contains all the parametric sensitivity coefficients and is called the parametric sensitivity matrix. $\mathbf{S_0}$ $n \times n$ matrix that contains all the initial sensitivity coefficients and is called initial sensitivity matrix. $\mathbf{I}$ is the $n \times n$ identity matrix. The following example shows how these matrixes are defined numerically.

Example 3.3a Consider the following linear system,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & \lambda_2 \\ \lambda_3 & \lambda_4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \lambda_1 x_1 + \lambda_2 x_2 \\ \lambda_3 x_1 + \lambda_4 x_2 \end{bmatrix}$$

The number of states $n = 2$, the number of parameters $m = 4$. We have,

$$\mathbf{J_x} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix} \lambda_1 & \lambda_2 \\ \lambda_3 & \lambda_4 \end{bmatrix}_{2\times 2} \qquad \mathbf{J_\lambda} = \frac{\partial \mathbf{f}}{\partial \lambda} = \begin{bmatrix} x_1 & x_2 & 0 & 0 \\ 0 & 0 & x_1 & x_2 \end{bmatrix}_{2\times 4}$$

$$\mathbf{S_\lambda} = \begin{bmatrix} \frac{dx_1}{d\lambda_1} & \frac{dx_1}{d\lambda_2} & \frac{dx_1}{d\lambda_3} & \frac{dx_1}{d\lambda_4} \\ \frac{dx_2}{d\lambda_1} & \frac{dx_2}{d\lambda_2} & \frac{dx_2}{d\lambda_3} & \frac{dx_2}{d\lambda_4} \end{bmatrix}_{2\times 4} \qquad \mathbf{S_0} = \begin{bmatrix} \frac{dx_1}{dx_0^1} & \frac{dx_1}{dx_0^2} \\ \frac{dx_2}{dx_0^1} & \frac{dx_2}{dx_0^2} \end{bmatrix}_{2\times 2}$$

The sensitivity equations of λ_1 and x_0^1 are as follows:

$$\mathbf{S}_{\lambda_1} = \begin{bmatrix} S_1^1 \\ S_1^2 \end{bmatrix} = \begin{bmatrix} \frac{dx_1}{d\lambda_1} \\ \frac{dx_2}{d\lambda_1} \end{bmatrix} = \begin{bmatrix} \lambda_1 x_1 + \lambda_2 x_2 + x_1 \\ \lambda_3 x_1 + \lambda_4 x_2 + 0 \end{bmatrix}, \mathbf{S}_{\lambda_1}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathbf{S}_{x_0^1} = \begin{bmatrix} S_1^{1-0} \\ S_2^{1-0} \end{bmatrix} = \begin{bmatrix} \frac{dx_1}{dx_0^1} \\ \frac{dx_1}{dx_0^1} \end{bmatrix} = \begin{bmatrix} \lambda_1 x_1 + \lambda_2 x_2 \\ \lambda_3 x_1 + \lambda_4 x_2 \end{bmatrix}, \mathbf{S}_{x_0^1}(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

From this example, we can find that the form of the sensitivity equations is very close to the original system ODE. It implies that there may

be some relationship between the original ODE and the adjacent sensitivity equations. The sensitivity equations show that the solutions of the sensitivity equation requires the knowledge of the solution of the original system equation at all points where the ODE solver calculates the right-hand side of the sensitivity equation. Connections between these two sets of equations can be made in several ways and lead to different methods to solve the sensitivity equations.

There are several methods that can be used to solve Eq. (3.26) and Eq. (3.28) [61, 67]. The simplest one is called *brute force method* or *indirect method* by using the finite-difference approximation,

$$\frac{d\mathbf{x}}{d\lambda_i} \approx \frac{\mathbf{x}(\lambda_i + \Delta\lambda_i) - \mathbf{x}(\lambda_i)}{\Delta\lambda_i}, \qquad i = 1, ..., m \tag{3.29}$$

Solving sensitivity equation in this way requires $m + 1$ simulations of the original model. Obviously, the accuracies of this method depend on the parameter change, $\Delta\lambda_i$, and the linearity of the system. The advantage of this method is its robustness. No modification to the system ODE is needed. The system can also be treated as a black box.

A set of more accurate methods is called *Direct methods* (DM). These solve the Eq. (3.26) and Eq. (3.28) directly. Numerical solutions of Eq. (3.26) and Eq. (3.28) requires knowledge of $\mathbf{J_x}$ and $\mathbf{J_\lambda}$ at each step of the ODE solver and thus the values of the system state variable $\mathbf{x}$ have to be known. Therefore, the system ODE Eq. (3.24) must be solved in advance, or simultaneously, and thus the computational cost of *DM* is quite high.

It was Dunker [24] who first showed that a special relation existed between the sensitivity equation Eq. (3.26), Eq. (3.28) and the system ODE Eq. (3.24). This relationship can be used to solve Eq. (3.26) and Eq. (3.28) with little extra computational cost when solving system ODE Eq. (3.24). Algorithms based on his idea are called *decoupled direct method* or *DDM*. It has been proved to be the best general method to calculate sensitivities [67].

The basic idea of Dunker begins by writing the *implicit* finite-time difference form of Eq. (3.24) in the time interval $[t_k, t_{k+1}]$ can be written as,

$$\dot{\mathbf{x}}(t_{k+1}) = \frac{\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k)}{\Delta t} = \mathbf{f}(\mathbf{x}(t_{k+1}), \boldsymbol{\lambda}, \mathbf{u}, t), \tag{3.30}$$

where $\Delta t = t_{k+1} - t_k$. Using Taylor expansion to expand $\mathbf{f}(\mathbf{x}(t_{k+1}), \boldsymbol{\lambda}, \boldsymbol{u}, t)$ near $x(t_k, \boldsymbol{\lambda}, \mathbf{u}, t)$ yields,

$$\mathbf{f}(\mathbf{x}(t_{k+1}), \boldsymbol{\lambda}, \mathbf{u}, t) = \mathbf{f}(\mathbf{x}(t_k), \boldsymbol{\lambda}, \mathbf{u}, t) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}(t_k)} \Delta\mathbf{x} + O(\mathbf{x}^2), \tag{3.31}$$

where $\Delta\mathbf{x} = \mathbf{x}(t_{k+1}) - \mathbf{x}(t_k)$. Ignoring the higher terms and substituting Eq. (3.30) into Eq. (3.31),

$$\frac{\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k)}{\Delta t} = \mathbf{f}(\mathbf{x}(t_k), \boldsymbol{\lambda}, \mathbf{u}, t) + \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}(t_k)} (\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k)) \tag{3.32}$$

Let $\mathbf{J}_\mathbf{x} = \dfrac{\partial \mathbf{f}}{\partial \mathbf{x}}$, and $\mathbf{J}_{\mathbf{x},k} = \left.\dfrac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}(t_k)}$, and rearrange Eq. (3.32) to get the following equations that solve the system ODE Eq. (3.24) iteratively,

$$\begin{aligned} \mathbf{x}(t_{k+1}) &= \mathbf{x}(t_k) + \Delta t \left(\mathbf{J} - \Delta t \mathbf{J}_{\mathbf{x},k}\right)^{-1} \mathbf{f}(\mathbf{x}(t_k), \boldsymbol{\lambda}, \mathbf{u}, t) \\ \mathbf{x}(t_{k+2}) &= \mathbf{x}(t_{k+1}) + \Delta t \left(\mathbf{J} - \Delta t \mathbf{J}_{\mathbf{x},k+1}\right)^{-1} \mathbf{f}(\mathbf{x}(t_{k+1}), \boldsymbol{\lambda}, \mathbf{u}, t) \end{aligned} \tag{3.33}$$

where $\mathbf{J}$ is a unitary matrix (all elements are 1).

From Eq. (3.33) we can see the major computing task to solve system ODE Eq. (3.24) is to calculate the matrix $(\mathbf{J} - \Delta t \mathbf{J}_{\mathbf{x},k})^{-1}$.

Similarly, let $\mathbf{S}_{\boldsymbol{\lambda}}(t_k) = \mathbf{S}_{\boldsymbol{\lambda},k}$, the finite-difference form of parametric sensitivity equation Eq. (3.28) is,

$$\dot{\mathbf{S}}_{\boldsymbol{\lambda},k+1} = \frac{\mathbf{S}_{\boldsymbol{\lambda},k+1} - \mathbf{S}_{\boldsymbol{\lambda},k}}{\Delta t} = \mathbf{J}_{\mathbf{x},k+1}\mathbf{S}_{\boldsymbol{\lambda},k+1} + \mathbf{J}_{\boldsymbol{\lambda},k+1} \tag{3.34}$$

Reorganizing this equation,

$$\mathbf{S}_{\boldsymbol{\lambda},k+1} = \left(\mathbf{J}_{\mathbf{x},k+1}\mathbf{S}_{\lambda,k+1} + \mathbf{J}_{\boldsymbol{\lambda},k+1}\right)\Delta t + \mathbf{S}_{\boldsymbol{\lambda},k},$$

Rewriting above equation yields an equation that solves the parametric sensitivity equation Eq. (3.26) iteratively,

$$\mathbf{S}_{\boldsymbol{\lambda},k+1} = \left(\mathbf{J} - \Delta t \mathbf{J}_{\mathbf{x},k+1}\right)^{-1}\left(\mathbf{J}_{\lambda,k+1}\Delta t + \mathbf{S}_{\boldsymbol{\lambda},k}\right) \tag{3.35}$$

Let $\mathbf{S}_{\mathbf{0}}(t_k) = \mathbf{S}_{\mathbf{0},k}$ and applying the same procedure to the initial sensitivity equation Eq. (3.24) yields an equation that solves the initial sensitivity equation Eq. (3.28) iteratively::

$$\mathbf{S}_{\mathbf{0}}(t_{k+1}) = \left(\mathbf{J} - \Delta t \mathbf{J}_{\mathbf{x},k+1}\right)^{-1}\mathbf{S}_{\mathbf{0}}\left(\mathbf{x}\left(t_k\right)\right) \tag{3.36}$$

Compare Eq. (3.33), Eq. (3.35) and Eq. (3.36), to find that the matrix $\left(\mathbf{J} - \Delta t \mathbf{J}_{\mathbf{x},k}\right)^{-1}$ used to solve Eq. (3.33) can be used to solve Eq. (3.35) and Eq. (3.36) as well. Thus only little extra computational cost is needed for solving Eq. (3.35) and Eq. (3.36) if Eq. (3.33) is to be solved.

This analysis shows that the sensitivity equation can be solved with little extra cost when solving the original system ODEs.

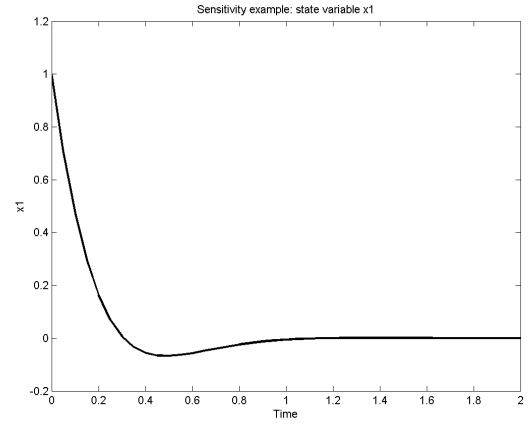
Example 3.3b For the system ODE used in **Example 3.3a**, assume we have $\boldsymbol{\lambda} = \begin{bmatrix} -2 & -3 & 5 & -6 \end{bmatrix}$ and $\mathbf{x}_0 = \begin{bmatrix} 1 & 1 \end{bmatrix}$, so we can calculate $\mathbf{S}_\lambda$ and $\mathbf{S_0}$ by using *DDM*. The results are shown in Figure-3.10.

In this section, we show that the sensitivity equation Eq. (3.26) and Eq. (3.28) can be solved as a byproduct of solving the original system Eq. (3.24) with little extra cost. This is a very important conclusion and it means we can get the function value and its derivative as a secondary function in one sampling process. With both primary and secondary function values, the quality of the response surface can be improved, or in other words, less samples are needed if the quality of the response surfaces remains the same.

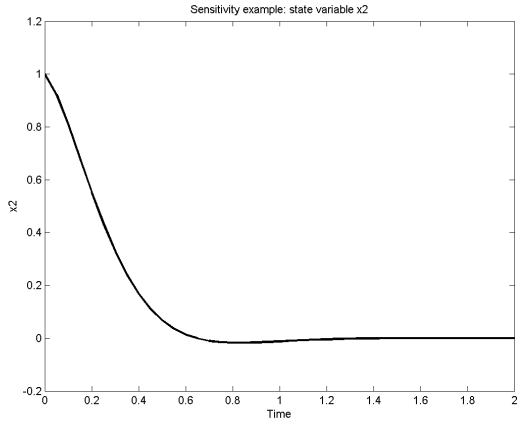
3.5.3 Gradient (Sensitivity) Enhanced MLS

The gradient enhanced response surface method (GERSM) uses both the primary function values and the gradient (sensitivity) information as the secondary function for construction of response surface. It has provided very attractive results in many applications [16, 17, 51, 56]. Generally, there are two ways to use sensitivity information to form a gradient enhanced moving least square response surface (GEMLS). The first method, which is called GEMLS1 in this dissertation, is to use sensitivity information to generate some 'pseudo samples' around a real sample. Assume we have a sample $O(\mathbf{x}_s)$, by using Taylor expansion,

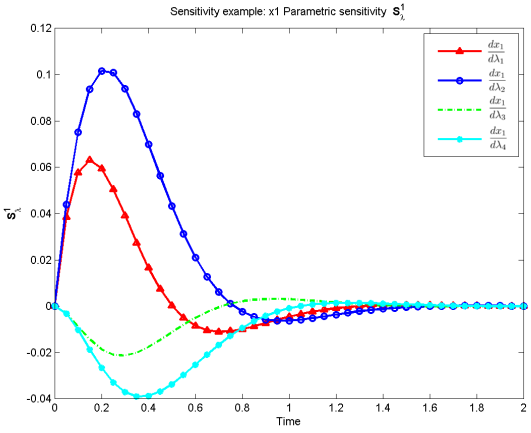
$$O(\mathbf{x}_s + \boldsymbol{\delta}) \approx O(\mathbf{x}_s) + \frac{\partial O}{\partial \mathbf{x}_s} d\mathbf{x}_s = O(\mathbf{x}_s) + \mathbf{S}_{\mathbf{x}_s}\boldsymbol{\delta} \tag{3.37}$$



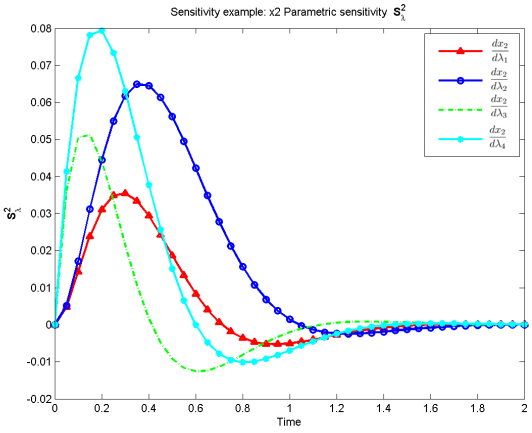
(a) x_1



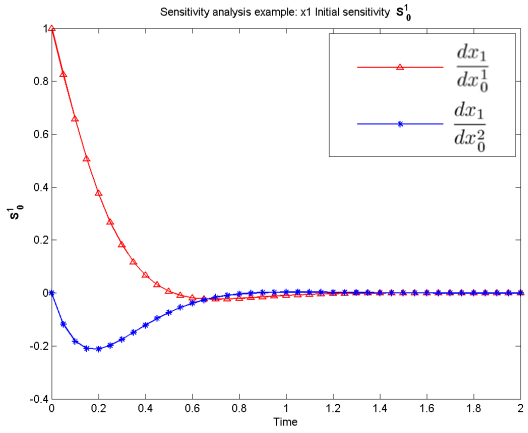
(b) x_2



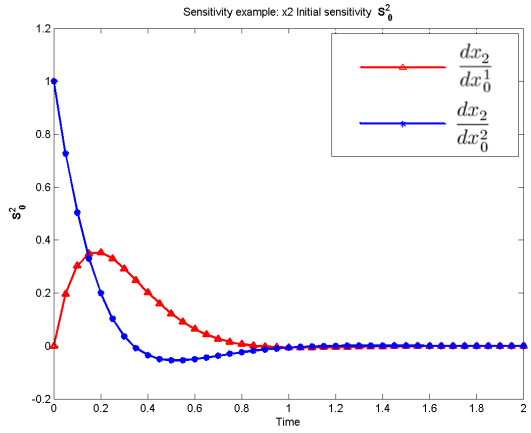
(c) $S_\lambda^1 = (\frac{dx_1}{d\lambda_1}, \frac{dx_1}{d\lambda_2}, \frac{dx_1}{d\lambda_3}, \frac{dx_1}{d\lambda_4})$



(d) $S_\lambda^2 = (\frac{dx_2}{d\lambda_1}, \frac{dx_2}{d\lambda_2}, \frac{dx_2}{d\lambda_3}, \frac{dx_2}{d\lambda_4})$



(e) $\mathbf{S_0^1} = (\frac{dx_1}{dx_0^1}, \frac{dx_1}{dx_0^2})$



(f) $\mathbf{S_0^2} = (\frac{dx_2}{dx_0^1}, \frac{dx_2}{dx_0^2})$

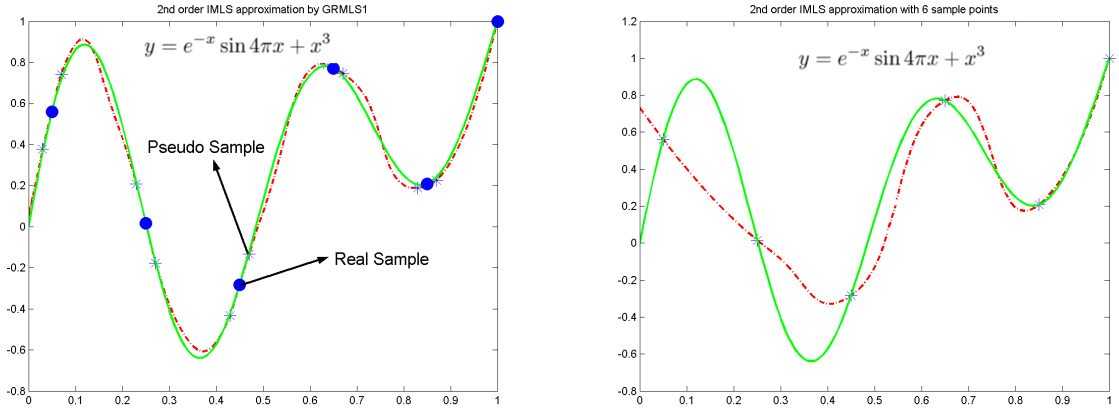
Figure 3.10: Sensitivity analysis example

where $\boldsymbol{\delta}$ should be small enough so Eq. (3.37) holds. From Eq. (3.37) it is relatively easy to get the observation at the point $\mathbf{x}_s + \boldsymbol{\delta}$ given the observation $O(\mathbf{x}_s)$ and the sensitivity information $\mathbf{S}_{\mathbf{x}_s}$. Define $O(\mathbf{x}_s + \boldsymbol{\delta})$ as pseudo samples around the real sample $O(\mathbf{x}_s)$. As pointed out in the previous section, when $O(\mathbf{x}_s)$ is determined, $\mathbf{S}_{\mathbf{x}_s}$ is obtained with less cost. Thus, the pseudo samples can be acquired along with the real sample $O(\mathbf{x}_s)$ with little cost. This method has also been called database argumentation in [54], or indirect method in [17]. The advantage of GEMLS1 is that no additional coding is required for the MLS method. The disadvantage is that the value of step size, $\boldsymbol{\delta}$, is hard to select. Generally $\delta < 5\%$ of the total range of $\mathbf{x}$ is the rule. In [54], a method to selecting a better step size by including the step size as one of parameters in the response surface model is given.

Example 3.4 *Pseudo Samples.* In Example 3.1, there are 11 samples used to reconstruct the original function. Figure-3.11(a) shows the result of using GEMLS1 with 6 samples and 10 pseudo samples to reconstruct the same function used in Example 3.1, which is much better than the result shown in Figure-3.11(b), which is generated by using MLS with 6 real samples.

Another method (GEMLS2 in this dissertation) is to treat sensitivity information as a secondary function. The response surface should fit both the original and secondary functions. A similar method is referred to direct method in [16, 17].

Since $O(\mathbf{x}_s) = \hat{O}(\mathbf{x}_s) + \varepsilon$, by taking derivative with respect to $x_j, j =$



(a) GEMLS1: 6 real plus 10 Pseudo samples (b) MLS with 6 real samples

Figure 3.11: GEMLS1 by using pseudo samples

$1, 2, ..., n$, we have,

$$\frac{\partial O(\mathbf{x})}{\partial x_j} = \frac{\partial \mathbf{p}(\mathbf{x})}{\partial x_j}\mathbf{a}(x) + \mathbf{p}(\mathbf{x})\frac{\partial \mathbf{a}(\mathbf{x})}{\partial x_j} + \frac{\partial \varepsilon}{\partial x_j} \tag{3.38}$$

Let $O_j^d(\mathbf{x}) = \dfrac{\partial O(\mathbf{x})}{\partial x_j}$, $\mathbf{p}_j^d = \dfrac{\partial \mathbf{p}(\mathbf{x})}{\partial x_j}$ and $\varepsilon_j^d = \mathbf{p}(\mathbf{x})\dfrac{\partial \mathbf{a}(\mathbf{x})}{\partial x_j} + \dfrac{\partial \varepsilon}{\partial x_j}$, then Eq. (3.38) can be rewritten as,

$$O_j^d(\mathbf{x}) = \hat{O}_j^d(\mathbf{x}) + \varepsilon_j^d = \mathbf{p}_j^d(\mathbf{x})\mathbf{a}(\mathbf{x}) + \varepsilon_j^d \tag{3.39}$$

Let $\mathbf{\Theta} = \begin{bmatrix} O \\ O_1^d \\ O_2^d \\ \vdots \\ O_n^d \end{bmatrix}, \hat{\mathbf{\Theta}} = \begin{bmatrix} \hat{O} \\ \hat{O}_1^d \\ \hat{O}_2^d \\ \vdots \\ \hat{O}_n^d \end{bmatrix}, \mathbf{\Pi} = \begin{bmatrix} \mathbf{p} \\ \mathbf{p}_1^d \\ \mathbf{p}_2^d \\ \vdots \\ \mathbf{p}_n^d \end{bmatrix}, \mathbf{e} = \begin{bmatrix} \varepsilon \\ \varepsilon_1^d \\ \varepsilon_2^d \\ \vdots \\ \varepsilon_n^d \end{bmatrix}$, and write Eq. (3.39) in matrix form as,

$$\begin{aligned} \mathbf{\Theta}(\mathbf{x}) &= \hat{\mathbf{\Theta}}(\mathbf{x}) + \mathbf{e} \\ \hat{\mathbf{\Theta}}(\mathbf{x}) &= \mathbf{\Pi}(\mathbf{x})\mathbf{a}(\mathbf{x}) \end{aligned} \tag{3.40}$$

Apply the same moving least square procedure to Eq. (3.40) as to Eq. (3.9), and form a set of similar equations,

$$\begin{aligned} \mathbf{a}(\mathbf{x}) &= \mathcal{A}^{-\mathbf{1}}\mathcal{B} \\ \mathcal{A}_{n_p \times n_p} &= \mathbf{\Pi}_s^T \mathbf{\Psi}(\mathbf{x}) \mathbf{\Pi}_s \\ \mathcal{B}_{n_p \times 1} &= \mathbf{\Pi}_s^T \mathbf{\Psi}(\mathbf{x}) \mathbf{\Theta}_s \end{aligned} \tag{3.41}$$

$$\text{where } \mathbf{\Theta}_s = \begin{bmatrix} O(\mathbf{x}_s^1) \\ O_1^d(\mathbf{x}_s^1) \\ \vdots \\ \underline{O_n^d(\mathbf{x}_s^1)} \\ \vdots \\ \overline{O(\mathbf{x}_s^{n_s})} \\ O_1^d(\mathbf{x}_s^{n_s}) \\ \vdots \\ O_n^d(\mathbf{x}_s^{n_s}) \end{bmatrix}_{[(n+1)\cdot n_s]\times 1} \qquad \mathbf{\Pi}_s = \begin{bmatrix} \mathbf{p}(\mathbf{x}_s^1) \\ \mathbf{p}_1^d(\mathbf{x}_s^1) \\ \vdots \\ \underline{\mathbf{p}_n^d(\mathbf{x}_s^1)} \\ \vdots \\ \overline{\mathbf{p}(\mathbf{x}_s^{n_s})} \\ \mathbf{p}_1^d(\mathbf{x}_s^{n_s}) \\ \vdots \\ \mathbf{p}_n^d(\mathbf{x}_s^{n_s}) \end{bmatrix}_{[(n+1)\cdot n_s]\times n_p}$$

$$\mathbf{\Psi} = \begin{bmatrix} \begin{bmatrix} w(\mathbf{x}-\mathbf{x}_s^1) & & \\ & \ddots & \\ & & w(\mathbf{x}-\mathbf{x}_s^1) \end{bmatrix} & & \mathbf{0} \\ & \begin{bmatrix} w(\mathbf{x}-\mathbf{x}_s^2) & & \\ & \ddots & \\ & & w(\mathbf{x}-\mathbf{x}_s^2) \end{bmatrix} & \\ \mathbf{0} & & \ddots \\ & & \begin{bmatrix} w(\mathbf{x}-\mathbf{x}_s^{n_s}) & & \\ & \ddots & \\ & & w(\mathbf{x}-\mathbf{x}_s^{n_s}) \end{bmatrix}_{(n+1)\times(n+1)} \end{bmatrix}$$

$\mathbf{\Theta}_s, \mathbf{\Pi}_s$ and $\mathbf{\Psi}$ are called extended observation matrices, basis function matrix and weight matrix, respectively.

It is worth noting that matrix $\mathcal{A}$ is still a $n_p \times n_p$ matrix so the computational cost to construct a response surface with sensitivity information

doesn't increase very much.

Example 3.5 Reconstruct the *Peaks*. The Matlab function called *peaks* is used to illustrate GERSM. This function has six local optima and the expression of this function is,

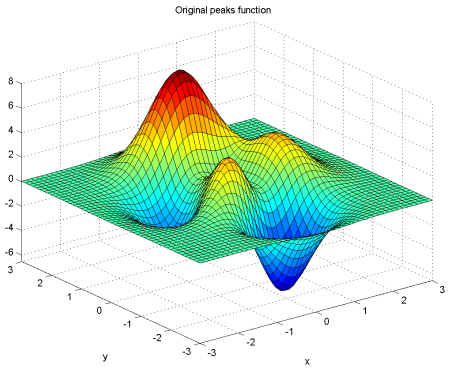
$$z = 3(1-x)^2 e^{-x^2-(y-1)^2} - 10(\frac{x}{5} - x^3 - y^5)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2}$$

Figure-3.12 shows the surface and contours of the original and reconstructed function by using MLS (49 real samples) and GEMLS1 (25 samples and 100 pseudo samples), GEMLS2 (25 samples and derivative as secondary function). 2304 test points (48×48 grids) are generated to test these three methods.

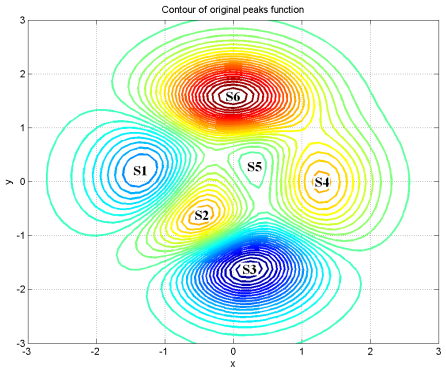
Table 3.1: Comparison of MLS and GEMLS

2304 test pts	RMS error	Distance between real and reconstructed stationary points					
		S1	S2	S3	S4	S5	S6*(global)
Original	0	0	0	0	0	0	0
MLS	0.8452	0.0339	0.7858	0.3631	0.5730	0.5334	0.2422
GEMLS1	0.8469	0.1328	0.2793	0.1155	0.2443	0.0894	0.0330
GEMLS2	0.6906	0.0233	0.4242	0.1526	0.0632	0.1232	0.0881

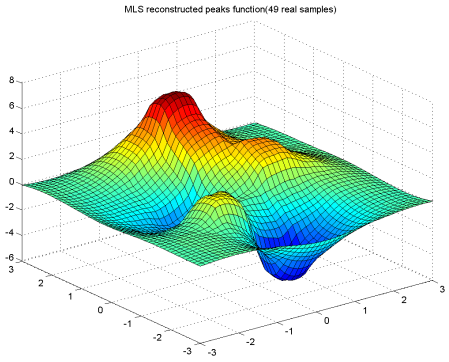
Table 3.1 summarizes the difference between the real and reconstructed *peaks* function. From Figure-3.11 and Table 3.1 it is found that even though GERSM uses only half of the samples, it outperforms or matches the results from the MLS. For this method, the stationary points reproduced are very close to the real ones, especially at the global point. These results showed that by using GEMLS, the computational cost to construct the response surface can be greatly reduced.



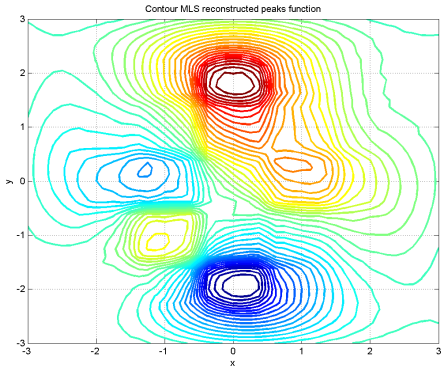
(a) Original Peaks



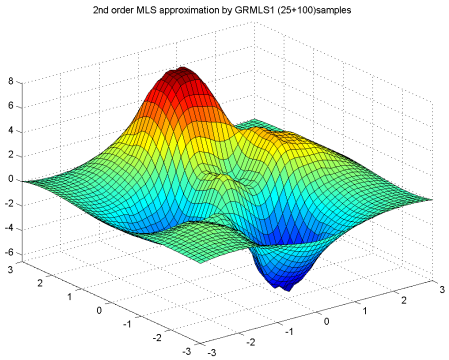
(b) Contour of original Peaks



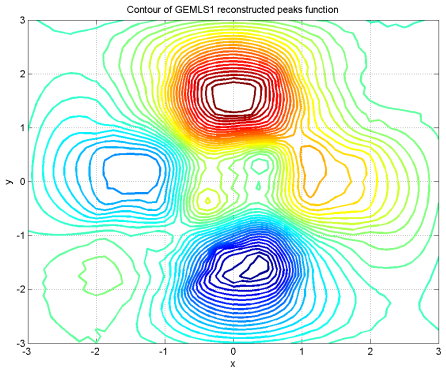
(c) MLS Reconstructed Peaks (49 samples)



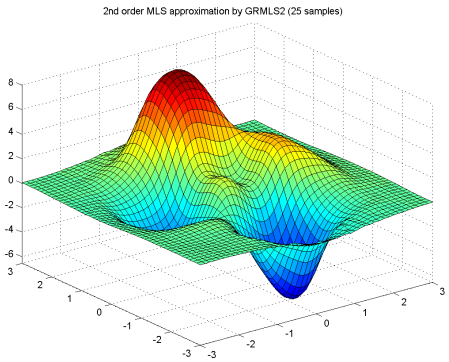
(d) Contour of MLS reconstructed peaks



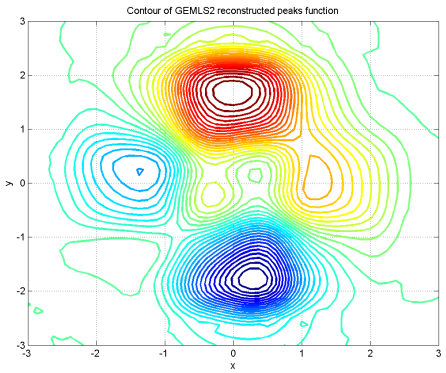
(e) GEMLS1 (25+100) reconstructed Peaks



(f) Contour of GEMLS1 reconstructed Peaks



(g) GEMLS2 (25) reconstructed Peaks



(h) Contour of GEMLS2 reconstructed Peaks

Figure 3.12: Gradient Enhanced Moving Least Square: Peaks in Matlab

3.6 Further Reducing the Computational Effort

As mentioned before, there are two types of sensitivity analysis: local and global. Local sensitivity analysis has been used to construct the gradient enhanced moving least square to reduce the computational effort in a previous section. In this section, global sensitivity analysis is used to further reduce the computational effort.

3.6.1 Monotonicity and Global Sensitivity Analysis

A multi-variable function $x(\lambda_1, \lambda_2, ...\lambda_n)$ is said to be increasing with regard to λ_i, one of its independent variables, if and only if $\dfrac{\partial x}{\partial \lambda_i} > 0$ in the whole parameter space, or decreasing w.r.t λ_i if $\dfrac{\partial x}{\partial \lambda_i} < 0$. Notice that the same function can increase w.r.t. one variable while decreasing w.r.t. another.

It is well known that the monotonicity of an objective function can often be used to obtain a simplified optimization problem. This principle has been successfully used in many design optimization problems [63]. Consider the following simple optimization problem:

$$\begin{array}{ll} \max\limits_{x} & x = 0.5\lambda_1 + \frac{\lambda_2^2}{\lambda_1} \\ \min\limits_{x} & x = 0.5\lambda_1 + \frac{\lambda_2^2}{\lambda_1} \\ s.t. & \lambda_1 \in [10, 20],\ \lambda_2 \in [10, 20] \end{array} \tag{3.42}$$

Assume evaluation of x is very time consuming, so the response surface method described in the previous section is used to solve this problem. The first step of RSM is to take samples. For this simple problem, by testing the monotonic character of the function, it can be found that only two samples

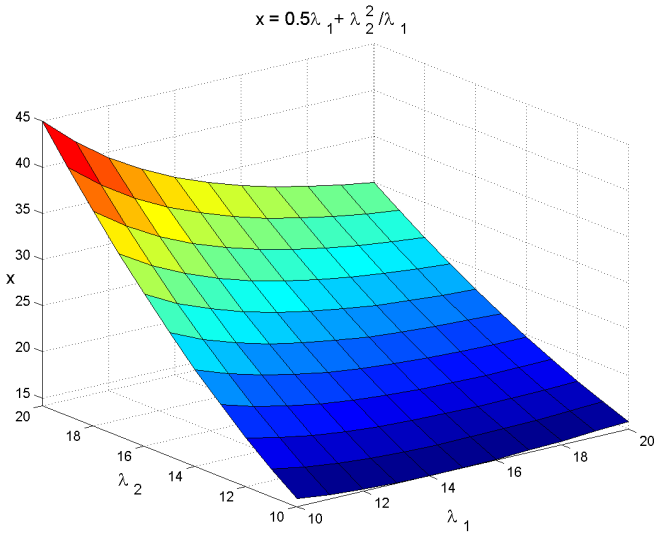


Figure 3.13: Monotonicity analysis of multi-variable function

are needed. This is because by studying the sensitivity coefficient, $S = \dfrac{dx}{d\lambda_2} = 2\lambda_2 \in [20, 40] > 0$, which means x is a monotonically increasing function with λ_2. Then the minimum of x will be x_{min}=$\lambda_{2min}^2 = 10^2 = 100$ and the maximum of x will be x_{max}=$\lambda_{2max}^2 = 20^2 = 400$.

Now consider the following problem:

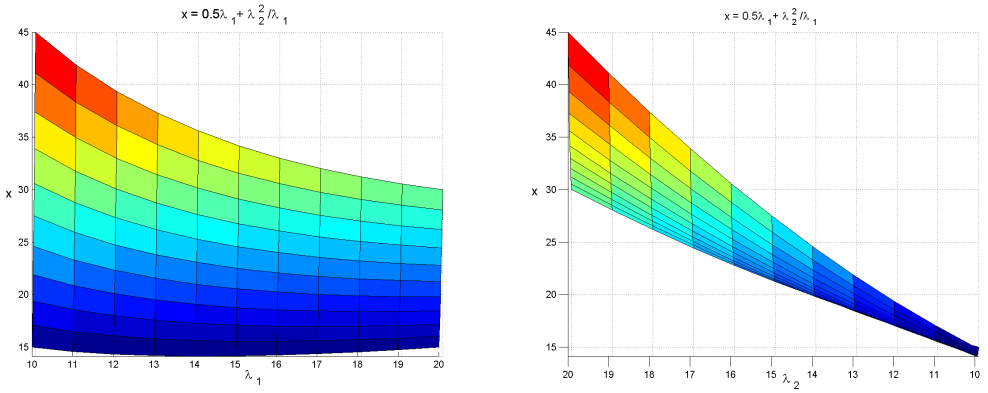
$$\begin{aligned} \max \quad & x = 0.5\lambda_1 + \tfrac{\lambda_2^2}{\lambda_1} \\ \min \quad & x = 0.5\lambda_1 + \tfrac{\lambda_2^2}{\lambda_1} \\ s.t. \quad & \lambda_1 \in [10, 20], \lambda_2 \in [10, 20] \end{aligned} \tag{3.43}$$

The shape of this function is shown in Figure-3.13.

We still try to use monotonicity analysis first by taking derivatives,

$$\begin{aligned} \frac{dx}{d\lambda_1} &= 0.5 - \left(\frac{\lambda_2}{\lambda_1}\right)^2 \\ \frac{dx}{d\lambda_2} &= 2\frac{\lambda_2}{\lambda_1} \end{aligned} \tag{3.44}$$

We can easily find that x is still monotonically increasing with λ_2, since $\frac{dx}{d\lambda_2} = 2\frac{\lambda_2}{\lambda_1} > 0$ is guaranteed. But the sign of $\frac{dx}{d\lambda_1}$ is not known. Its value is between -3.5 to 0.25, depending on the value of λ_1 and λ_2. Thus the output x is not a monotonic function w.r.t λ_1. This also can be seen from Figure-3.14(a), which shows that the minimum x is not located at corner value of λ_1. However, since we do know x is monotonically increasing to λ_2, as shown in Figure-3.14(b), then only the corner values of λ_2 and several samples along λ_1 is needed to be taken to construct the response surface. Because of the monotonicity of the objective function, the number of samples is greatly reduced. From the



(a) x value along λ_1 direction, not monotonic (b) x value along λ_2 direction, monotonic

Figure 3.14: Monotonicity analysis example

above example, it can be found that the role of sensitivity analysis here is not

to get the value of sensitivity coefficient at certain point in the domain, but to find its sign when the variables λ_1, λ_2 change in the whole range. Thus we call this process global sensitivity analysis, in comparison to local sensitivity analysis discussed in previous sections. Global sensitivity analysis is based on the local analysis but its main purpose is to find the monotonic information of the function. We use $S_{ij\square}$ to denote global sensitivity coefficient. Clearly, the value of $S_{ij\square}$ is not a number but an interval.

3.6.2 Global Sensitivity Analysis of ODE Systems

Global sensitivity has been used in many optimization problems, such as linear circuit worst case simulation [77], design optimization [63], etc. However, testing the sign of $S_{ij\square}$ is a very difficult task, especially for dynamic system represented as ODEs where time t plays an important role.

The purpose of global sensitivity analysis of ODE system is to explore the monotonicity characteristics of the system output with respect to the uncertain parameters.

Consider a dynamic system with uncertain parameters:

$$x = x(t, [\underline{\lambda_1}, \overline{\lambda_1}], [\underline{\lambda_2}, \overline{\lambda_2}], [\underline{\lambda_3}, \overline{\lambda_3}], ..., [\underline{\lambda_n}, \overline{\lambda_n}]), \tag{3.45}$$

The local sensitivity coefficient provides the information of a single point λ_s in the parameter space, which can be denoted as,

$$S_{i,\lambda_s} = \left.\frac{\partial x}{\partial \lambda_i}\right|_{\lambda=\lambda_s} = \left.\frac{\partial x}{\partial \lambda_i}\right|_{\lambda_1=\lambda_s^1, \lambda_2=\lambda_s^2, ..., \lambda_i=\lambda_s^i, ..., \lambda_n=\lambda_s^n} \tag{3.46}$$

As mentioned before, we can solve S_{i,λ_s} with *DDM* and S_{i,λ_s} is a curve that changes with time. However, since we are dealing with an uncertain system, the parameter $\boldsymbol{\lambda}$ can vary within a range. We want to study the effect of changing all these parameters together. In a mathematic form, we want to study the global sensitivity coefficient:

$$S_{ij\square} = \left.\frac{\partial x_i}{\partial \lambda_j}\right|_{\lambda_1 \in [\underline{\lambda_1}, \overline{\lambda_1}], \lambda_2 \in [\underline{\lambda_2}, \overline{\lambda_2}], \ldots, \lambda_n \in [\underline{\lambda_n}, \overline{\lambda_n}]} \tag{3.47}$$

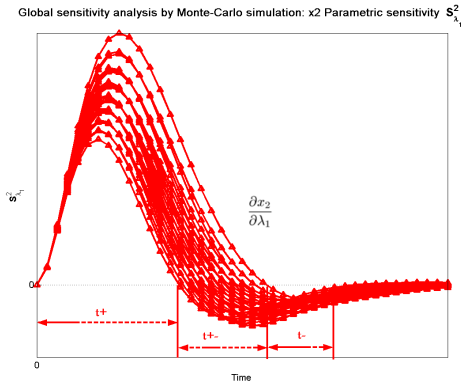
For monotonicity analysis, we only need to determine the sign of $S_{ij\square}$ but not the exact range. Consider the following example.

Example 3.6 *Global sensitivity analysis:* For the system ODE used in **Example 3.3**, assume we have the following uncertain parameters and a fixed initial condition: $\lambda = [[-2.5, -1.5], [-3.5, -2.5], 5, -6]$ and $\mathbf{x_0} = \begin{bmatrix} 1 & 1 \end{bmatrix}$. We can use Monte-Carlo Method and *DDM* to calculate $\mathbf{S}_\lambda$ and $\mathbf{S_0}$ with the uncertain parameters vary in their range. The global coefficients $S_{21\square}, S_{22\square}, S_{23\square}, S_{24\square}$ calculated by Monte-Carlo method are shown in Figure-3.15a-d. It can be found that when the uncertain parameters varying in their range, the sensitivity curves form a boundary similar to the system states. Each global sensitivity coefficient $S_{ij\square}$ is a collection of sensitivity curves and we call the collection of curves a *sensitivity band*. Each sensitivity band divides the t axis into three types of *monotone intervals*, namely $t+, t-$ and $t\pm$ intervals. In $t+$ interval, the value of the global sensitivity coefficient is always positive and in $t-$ intervals the global sensitivity coefficient is always negative. In $t\pm$ interval, the sign of the global sensitivity coefficient changes between positive

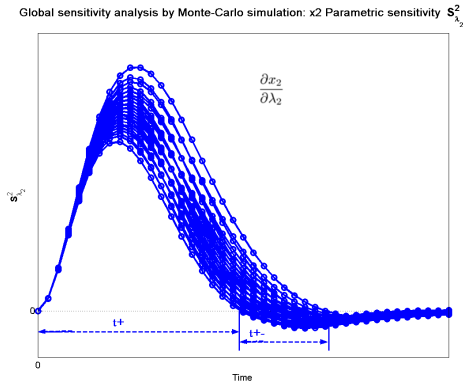
and negative. In the $t+$ or $t-$ interval, the objective function is monotonically increasing or decreasing with corresponding uncertain parameters. Only the corner value of these uncertain parameters needs to be sampled for response surface construction. In the $t\pm$ interval, more samples are needed.

There is no easy way to find these monotone intervals because in order to find the sensitivity band the boundary of the state variable must be found first, while the purpose of the global sensitivity analysis is to find the boundary of the state variables more effectively. Thus, unless the monotonic intervals can be found with less computational cost, global sensitivity analysis will not help to reduce the computational effort. There are two methods that can be used to estimate the monotonic intervals effectively:

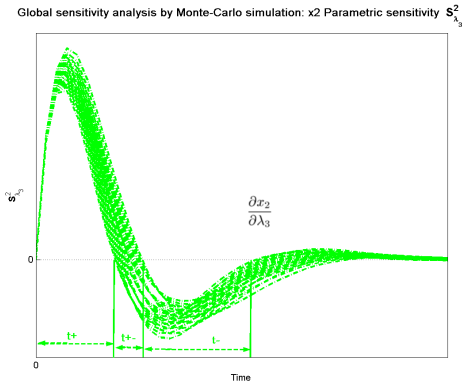
1. Less accurate method. Estimate the monotone intervals by using local sensitivity coefficient with uncertain parameters at their nominal values. In this case, global sensitivity analysis becomes local sensitivity analysis. There is only one curve that divides the time axis into only two types of intervals: $t+$ or $t-$, as shown in Figure-3.15(e),(f), where $S_{23\square}$ and $S_{24\square}$ are approximated by this method. The monotonic intervals approximated by this method are not very accurate. However, it still gives some monotonic information qualitatively. Around the time where the sensitivity curve crosses zero, more samples, both in parametric and time space, should be taken since it is close to the $t\pm$ intervals. At the time span where the curve is far away from zero, it is quite safe to treat this time span as $t+$ or $t-$ intervals.



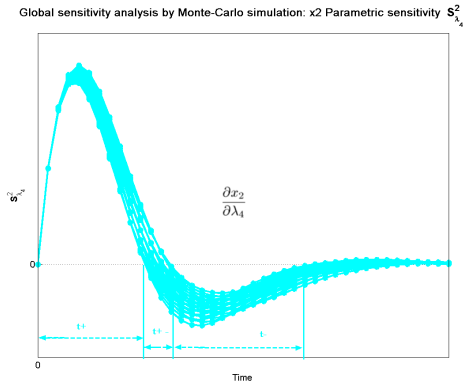
(a) Global sensitivity analysis by MC: S_{21}□



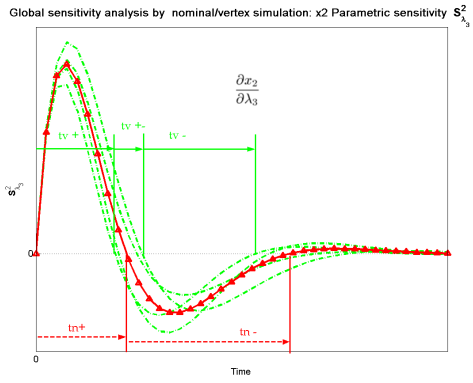
(b) Global sensitivity analysis by MC: S_{22}□



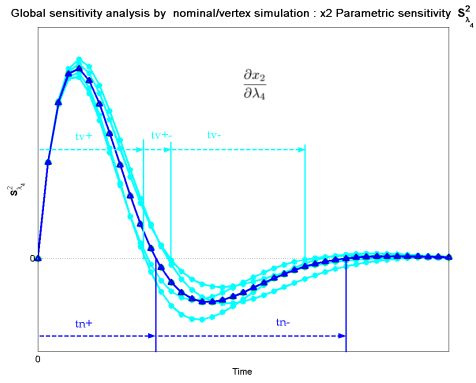
(c) Global sensitivity analysis by MC: S_{23}□



(d) Global sensitivity analysis by MC: S_{24}□



(e) Global sensitivity analysis by nominal/vertex simulation: S_{23}□



(f) Global sensitivity analysis by nominal/vertex simulation: S_{24}□

Figure 3.15: Global sensitivity and monotonicity analysis

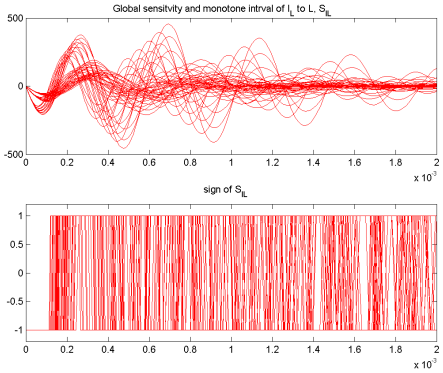
2. More accurate method. Use vertex simulation to form the sensitivity band to approximate the monotone intervals. More accurate estimations can be given by vertex simulation, as shown in Figure-3.15(e),(f).

For certain systems, the time intervals are quite short and monotone analysis will not help too much. For example, the step response of a simple RLC circuit in Figure-3.16(d) shows no strong sign of monotonicity to the value of L and C, as shown in Figure-3.16(a),(b),(c), among three parameters, only the R has pretty long $t+$ and $t-$ monotonic intervals, while monotonic intervals of R and C basically are all $t\pm$ intervals. However, for certain systems, they show strong monotonicity. The extreme case is that the $t+$ or $t-$ monotonic intervals of one or more global sensitivity coefficients are so long that it cover all the time span $[0, T]$ to be simulated. The following theorem will show the how to find such kind of system effectively.

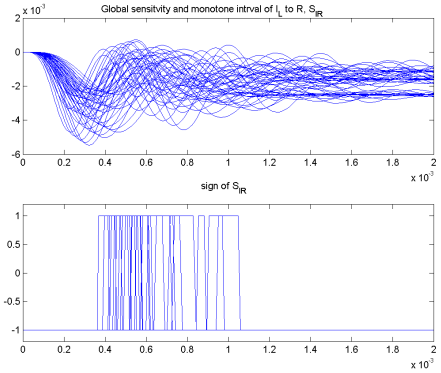
3.6.3 Monotone Theorem

Consider the dynamic system,

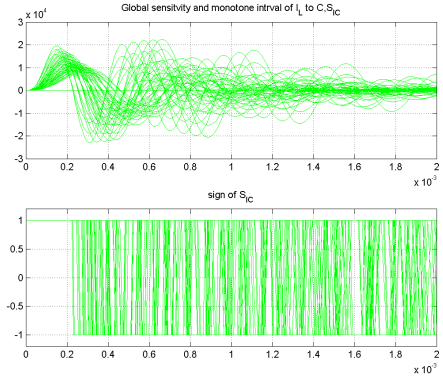
$$\begin{cases} \dot{x}_1 = f_1(x_1, x_2, \cdots, x_n, \lambda_1, \lambda_2, \cdots, \lambda_m, t) \\ \dot{x}_2 = f_2(x_1, x_2, \cdots, x_n, \lambda_1, \lambda_2, \cdots, \lambda_m, t) \\ \qquad\qquad \vdots \\ \dot{x}_n = f_n(x_1, x_2, \cdots, x_n, \lambda_1, \lambda_2, \cdots, \lambda_m, t) \end{cases}$$



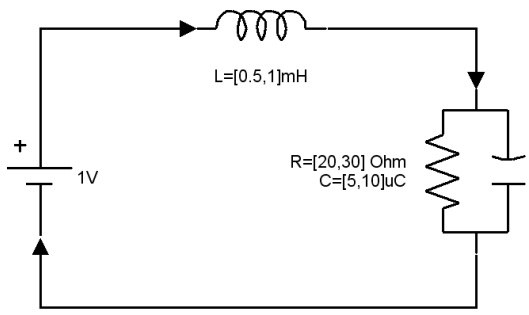
(a) Global sensitivity analysis of L



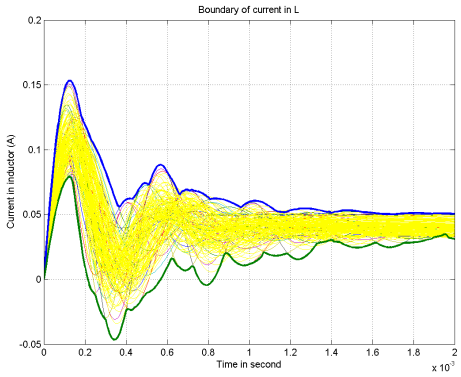
(b) Global sensitivity analysis of R



(c) Global sensitivity analysis of C



(d) LRC circuit with uncertain parameters



(e) Boundary by GEMLS and monte carlo results

Figure 3.16: System without strong monotonicity

The sensitivity coefficient of parameter λ_j is $S_{ij} = \dfrac{\partial x_i}{\partial \lambda_j}$,

$$\begin{aligned}
\dot{S}_{ij} &= \frac{\partial f_1}{\partial x_1} S_{1j} + \frac{\partial f_2}{\partial x_2} S_{2j} + ... + \frac{\partial f_n}{\partial x_n} S_{nj} + \frac{\partial f_i}{\partial \lambda_j} \\
&= \frac{\partial f_i}{\partial x_i} S_{ij} + \left(\sum_{l=1, l \neq i}^{n} \frac{\partial f_i}{\partial x_i} S_{lj} + \frac{\partial f_i}{\partial \lambda_j} \right) \\
&= N(t) S_{ij} + M(t)
\end{aligned} \tag{3.48}$$

$$S_{ij}(0) = 0$$

Equation (3.48) is called a *monotonicity equation* and the following theorem holds:

Monotone theorem. For the monotonicity equation

$$\begin{aligned}
&\dot{S} = N(t)S(t) + M(t) \\
&S(0) = 0
\end{aligned}$$

1. If $\forall t \in (0, T], M(t) > 0$, then $S(t) > 0$.

2. If $\forall t \in (0, T], M(t) < 0$, then $S(t) < 0$.

3. If $\forall t \in (0, T], M(t) = 0$, then $S(t) = 0$.

Proof of conclusion (1): Refer to Figure-3.17.

$$\because \qquad S(0) = 0, M(0) > 0$$

$$\therefore \qquad \dot{S}(0) = N(t)S(0) + M(0) > 0$$

Also, by the definition of derivatives, we will have $\delta_t > 0$ so that:

$$\left.\begin{array}{c} S(0) = 0 \\ \dot{S}(0_+) = \underset{\delta_t}{Lim} \dfrac{S(0 + \delta_t) - S(0)}{\delta_t} \end{array}\right\} \Rightarrow S(\delta_t) > 0$$

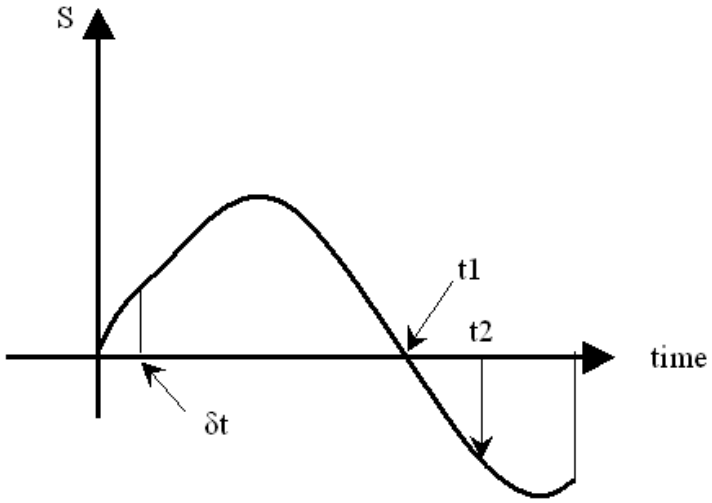


Figure 3.17: Prove of monotone theorem

Assume $t_1 > 0$ is the first point that $S(t_1) = 0$, in other words,

$$\exists(\delta_t < t_1 < T), S(t_1) = 0$$
$$\delta < t < t_1, \forall S(t) > 0$$

Then we have:

$$\dot{S}(t_{1-}) = N(t_{1-})S(t_{1-}) + M(t_{1-}) = M(t_{1-}) > 0 \tag{3.49}$$

However, by definition we have:

$$\dot{S}(t_{1-}) = \lim_{\delta_t \to 0} \frac{S(t_{1-}) - S(t_{1-} - \delta_t)}{\delta_t} = \frac{0 - S(t_{1-} - \delta_t)}{\delta_t} < 0 \tag{3.50}$$

Eq. 3.49 conflicts with Eq. 3.50 so the assumption is not correct. Thus, for $\forall t \in (0, T], S(t) \neq 0$.

Further, assume $\exists t_2 \in (\delta_t, T], S(t_2) < 0$. Since $S(\delta_t) > 0$, by intermediate value theorem, $\exists t_1$ and $S(t_1) = 0$. This is conflict with the above

conclusion and thus the assumption $\exists t_2 \in (\delta_t, T], S(t_2) < 0$ is not correct. Combined with above conclusion, $S(t)$ can only be great than zero. Conclusion (1) is proved. Using the same procedure, conclusion (2) and (3) can also be proved.

The power of this theorem is that if the sign of $M(t)$ is fixed in the time range $[t_0, T]$ and $S(t_0) = 0$, then the sign of $S(t)$ depends only on the sign of $M(t)$. No need to calculate the sign of $N(t)$.

Notice that from Eq. 3.48, since $M(t)$ is the summation of all the sensitivity coefficients and the parametric Jacobian terms, it is very hard to know the value of $M(t)$ and thus the usage of the monotone theorem is limited. However, for certain decoupled systems, using monotone theorem can greatly reduced the computational cost. The following Two-Tank example will show how this theorem can be used.

Example 3.7: Two-tank, use of monotone theorem:

Figure-3.18 shows the two-tank system that has been widely used as a benchmark problem for uncertain analysis. The system equations are:

$$\begin{cases} \dot{V}_1 = Au(t) - k_1\sqrt{V_1} \\ \dot{V}_2 = k_1\sqrt{V_1} - Bk_2\sqrt{V_2} \end{cases}$$

where

$$A = step(V_{1\max} - V_1) = \begin{cases} 1 & V_1 \le V_{1\max} \\ 0 & V_1 > V_{1\max} \end{cases}$$
$$B = step(V_{2\max} - V_2) = \begin{cases} 1 & V_2 \le V_{2\max} \\ 0 & V_1 \le V_{2\max} \end{cases}$$

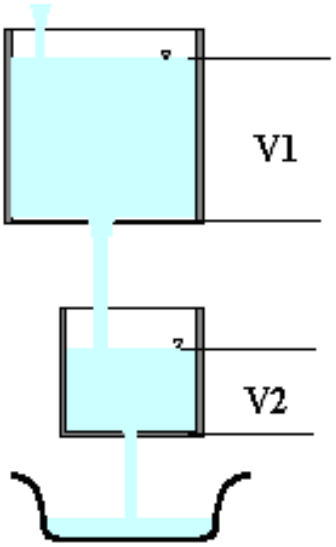


Figure 3.18: The two-tank system

$V_{1,2}$ stand for the volume of the two tanks and $k_{1,2}$ stands for the flow rate coefficient. $u(t) \geq 0$ is the input flow rate, which is independent of $V_{1,2}$. By introducing two step function A, B, above model accounts for the saturation of the two tanks. Now assume we have two uncertain parameters, $k_1 \in [0.8, 1], k_2 \in [0.5, 0.7]$ and we need to find the boundary of V_1 and V_2. If the monotonicity is not studied, we need to search in both k_1 and k_2 directions.

Calculate the global sensitivity coefficient of k_1 and k_2, w

$$\begin{aligned}
\dot{S}_{11} &= \frac{\partial \dot{V}_1}{\partial k_1} = \frac{dA}{dV_1} u - \sqrt{V_1} - \frac{k_1}{2\sqrt{V_1}} \frac{\partial V_1}{\partial k_1} \\
&= -\delta(V_{1\max} - V_1) \frac{\partial V_1}{\partial k_1} u - \sqrt{V_1} - \frac{k_1}{2\sqrt{V_1}} \frac{\partial V_1}{\partial k_1} \\
&= -\sqrt{V_1} + \left(-\delta(V_{1\max} - V_1) u - \frac{k_1}{2\sqrt{V_1}} \right) \frac{\partial V_1}{\partial k_1}
\end{aligned} \tag{3.51}$$

$$\dot{S}_{12} = \frac{\partial \dot{V}_1}{\partial k_2} = 0 \tag{3.52}$$

$$\begin{aligned}
\dot{S}_{21} &= \frac{\partial \dot{V}_2}{\partial k_1} = \sqrt{V_1} + \frac{k_1}{2\sqrt{V_1}}\frac{\partial V_1}{\partial k_1} - (\left(\frac{dB}{dk_1}\right) k_2\sqrt{V_2} + Bk_2\frac{\partial V_2}{\partial k_1}) \\
&= \sqrt{V_1} + \frac{k_1}{2\sqrt{V_1}}\frac{\partial V_1}{\partial k_1} - (-\delta(V_{2\max} - V_2)\frac{\partial V_2}{\partial k_1}k_2\sqrt{V_2} + Bk_2\frac{\partial V_2}{\partial k_1}) \\
&= \sqrt{V_1} + \frac{k_1}{2\sqrt{V_1}}\frac{\partial V_1}{\partial k_1} + [\delta(V_{2\max} - V_2)k_2\sqrt{V_2} - Bk_2]\frac{\partial V_2}{\partial k_1}
\end{aligned} \tag{3.53}$$

$$\begin{aligned}
\dot{S}_{22} &= \frac{\partial \dot{V}_2}{\partial k_2} = \frac{k_1}{2\sqrt{V_1}}\frac{\partial V_1}{\partial k_2} - \left[\left(\frac{dB}{dk_2}\right) k_2\sqrt{V_2} + B\left(\sqrt{V_2} + \frac{k_2}{2\sqrt{V_2}}\frac{\partial V_2}{\partial k_2}\right)\right] \\
&= 0 - \left[-\delta(V_{2\max} - V_2)\frac{\partial V_2}{\partial k_2}k_2\sqrt{V_2} + B\sqrt{V_2} + \frac{Bk_2}{2\sqrt{V_2}}\frac{\partial V_2}{\partial k_2}\right] \\
&= -B\sqrt{V_2} + \frac{\partial V_2}{\partial k_2}\left[\delta(V_{2\max} - V_2)k_2\sqrt{V_2} - \frac{Bk_2}{2\sqrt{V_2}}\right]
\end{aligned} \tag{3.54}$$

From above equations, we can find that:

$$\begin{aligned}
M_{11} &= -\sqrt{V_1} \\
M_{12} &= 0 \\
M_{21} &= \sqrt{V_1} + \frac{k_1}{2\sqrt{V_1}}\frac{\partial V_1}{\partial k_1} = \sqrt{V_1} + \frac{k_1}{2\sqrt{V_1}}S_{11} \\
M_{22} &= -B\sqrt{V_2}
\end{aligned}$$

Applying the monotone theorem, $S_{11} \leq 0$, $S_{22} \leq 0$ since $M_{11} \leq 0, M_{22} \leq 0$. For S_{21}, the sign of $M_{21} = \sqrt{V_1} + \frac{k_1}{2\sqrt{V_1}}\frac{\partial V_1}{\partial k_1} = \sqrt{V_1} + \frac{k_1}{2\sqrt{V_1}}S_{11}$ could not be determined. However, qualitative monotonicity information can still be acquired by studying M_{21}. Assume the initial condition is $V_1 = 1000$

and $V_2 = 0$. Because at time zero, $S_{11} = 0$ and thus M_{21} must be larger than zero. We can imagine that M_{21} will be larger than zero for a while and thus S_{21} will also be larger than zero. With the decreasing of V_1 and S_{11} (became negative), M_{21} will decrease to zero and move to negative region and thus S_{21} will also became negative.

Nevertheless, we now know V_1 will monotonically decrease with k_1 and V_2 will monotonically decrease with k_2. Thus, upper bound of V_2 will be the envelope of all the possible trajectory generated by the parameter $[\forall k_1, \underline{k_2}]$, the lower bound will be the envelope of all the possible trajectory generated by the parameter $[\forall k_1, \overline{k}_2]$. Thus, we only need to search $[k_1, k_2]$ space in k_1 direction in order to generate the boundary of V_2.

For V_1, it is even simpler since V_1 is a monotonic decreasing function of k_1 and thus the upper bound is given by $\underline{k_1}$ and the lower bound is given by $\overline{k_1}$.

Figure-3.19 shows monte-carlo simulation results of the above two-tank system with input $u = 0$. Figure-3.19(a) shows the result of $V1$, S_{11} and S_{12}. Since the lower tank will not influence the upper tank, the value of k_2 will not influence the status of the upper tank. Because of $S_{11} < 0$, the upper boundary of V_1 can be acquired by simulating the system with $k_1 = 0.7$. The lower boundary of V_1 can be acquired by simulating the system with $k_1 = 1$. This result can be explained physically: if the orifice of the upper tank is larger, at any time the volume in the upper tank will be lower and vise versa.

Figure-3.19(b) shows the result of V_2. From Figure-3.19, we can find that for all possible combination of $k1 \in [0.7, 1], k2 \in [0.3, 0.5]$, $S_{22} = \dfrac{\partial V_2}{\partial k_2}$ is always less than zero. Also, the sign of S_{21} changes from positive to negative, which proved the qualitative monotonicity analysis by studying S_{21}. For the boundary of V_2, the upper boundary will be given by $k_2 = 0.3$. The lower boundary will be given by $k_2 = 0.5$. The physical meaning of the monotonicity is that if the orifice of the lower tank is smaller, the higher the volume of the lower tank and vise versa. To find the boundary of V_2, only the value of k_1 will be searched.

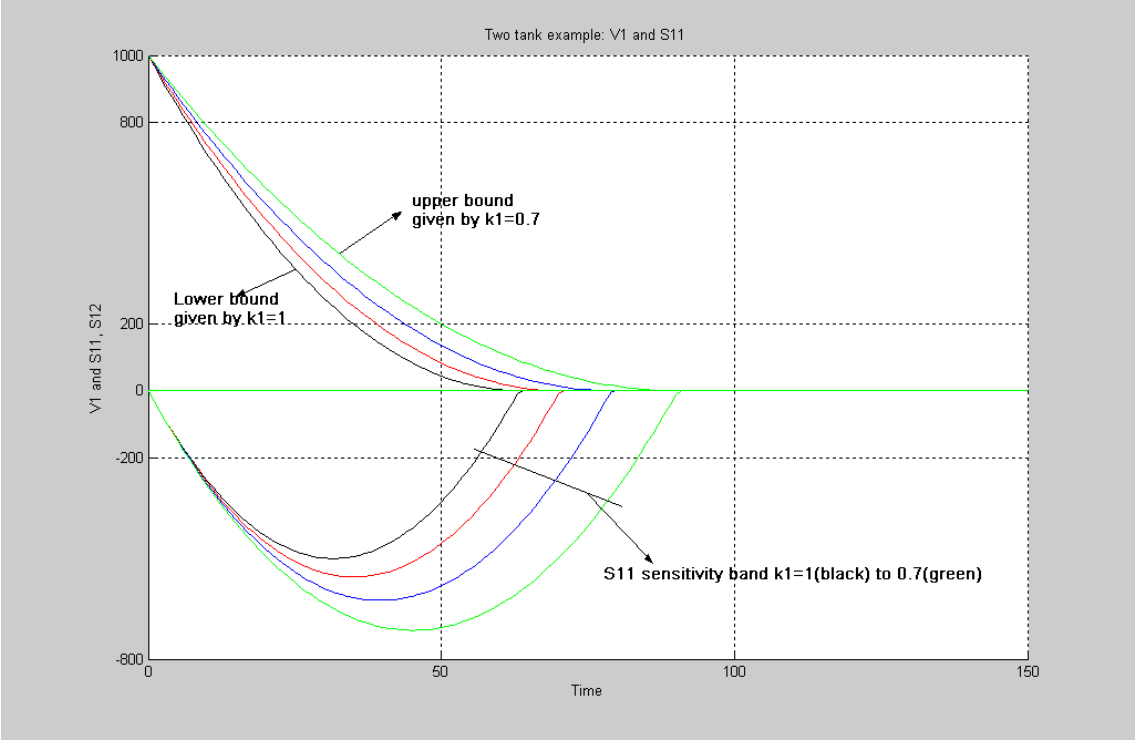
From the above example, it can be found that the monotonicity theorem can be applied to systems, for which the sign of $M(t)$ in Eq. (3.48) can be easily determined. Such systems are usually highly decoupled and systems whose state variables are physically bounded, such as in the two-tank example that $V_{12} > 0$ is known physically.

3.7 Numerical Example

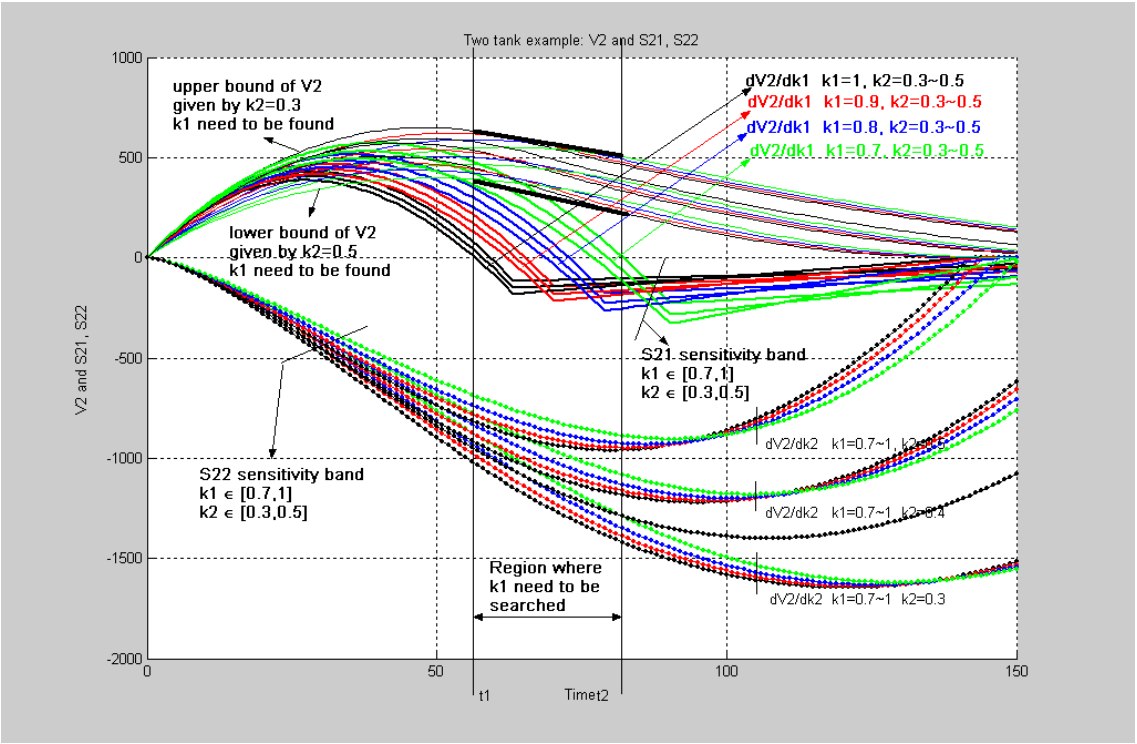
In this section, an example which illustrates our simulation approach is given. The example is a linear system with two uncertain parameters $k_1 \in [0.4, 0.8], k_2 \in [0.4, 0.8]$ and two certain parameters $k_3 = k_4 = 0.6$. The initial condition is a certain value $\mathbf{x_0} = [1, 1.5]$. The system ODE is:

$$\begin{cases} \dot{x}_1 = (k_1 - 1)x_1 + k_2 x_2 \\ \dot{x}_2 = -k_3 x_1 + (k_4 - 1)x_2 \end{cases}$$

The above system can be proven to be robustly stable by using Khari-



(a) V1 and S11,S12



(b) V2 and S21,S22

Figure 3.19: Two Tank example: use of monotone theorem

tonv's theorem in frequency domain. Note that one advantage of our approach is that since it is not limited to linear systems, it can handle general nonlinear dynamic systems. *DDE* method is used to solve the adjunct sensitivity equation. Monotonicity theorem can not be applied easily so monotonicity test is not used. Use 16 samples and the corresponding sensitivity information (GEMLS2) to construct the moving least square surface in the [0,10] second time range with time step $\Delta t = 1s$.

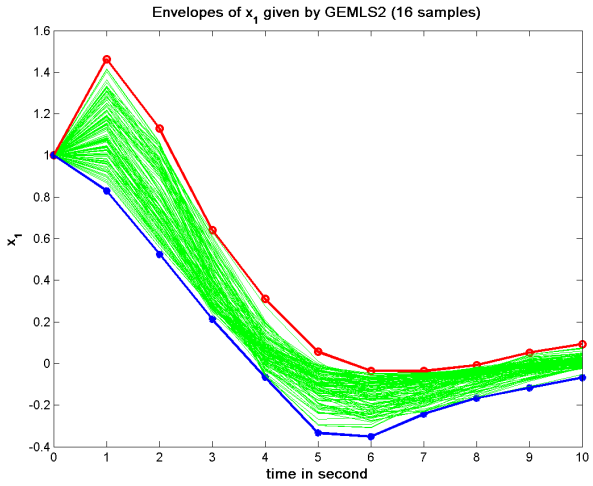
In Figure-3.20(a),(b) are the time response of x_1 and x_2, given by Monte-Carlo of 125 simulations and GEMLS2 of only 16 simulation. It shows that GEMLS2 gives a very good approximation of the upper and lower envelopes of both variables, with much less computational cost. Both of the envelopes show that the system is robustly stable as proved by Kharitonv's theorem.

Figure-3.20(c) shows the hyper-rectangle in phase plane given by the bound of x_1 and x_2. It can be seen that the box covers all the Monte-Carlo results. However, the rectangle is not a good approximation of the set composed by all possible states at a given time instance. This set can be imagined as a cross section of the system performance tube (also called flow pipe), which is the collection of all the possible trajectories, as shown in Figure-3.20(d). As can be seen, there is some 'empty space' in the rectangle that the system will not reach. In other words, although the envelope of each state variable is accurately estimated, the *boundary (edge of the set)* of the system performance tube is not accurately approximated. This set is also called *reachable set* at

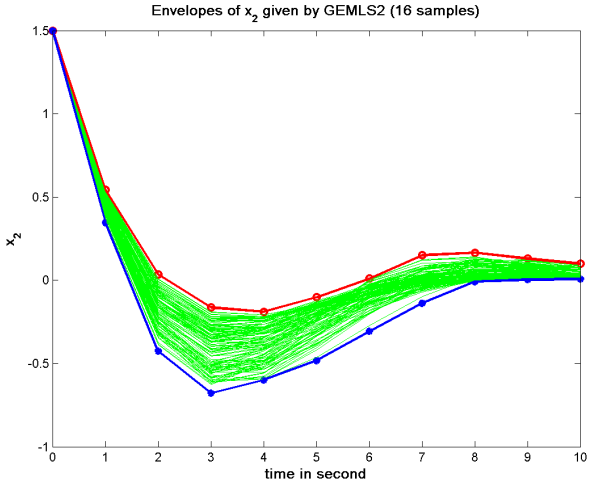
time t of the system. In next chapter, we will show how the reachable set can be accurately acquired by our methods.

3.8 Summary

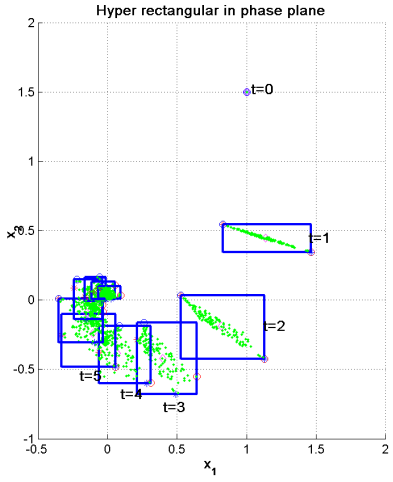
In this chapter, the computational cost of simulation of uncertain system is first analyzed and a gradient response surface response method is proposed to reduce the computational cost, based on local sensitivity analysis. Further, the concept of global sensitivity and sensitivity band are introduced for monotonicity analysis to further reduce the computational cost. A theorem to test the monotonicity of certain ODE systems effectively is proved. By integrating all these methods together, the computational cost of simulation of uncertain system can be greatly reduced. Figure-3.21 shows the structure of the proposed method.



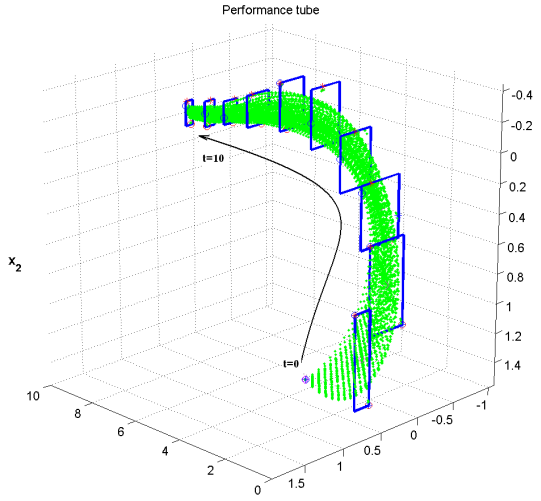
(a) Envelops of x_1 given by GEMSL2(16 samples)



(b) Envelops of x_2 given by GEMSL2(16 samples)



(c) Hyper rectangle in phase plane



(d) Hyper rectangle in state space

Figure 3.20: Numerical example: Performance tube

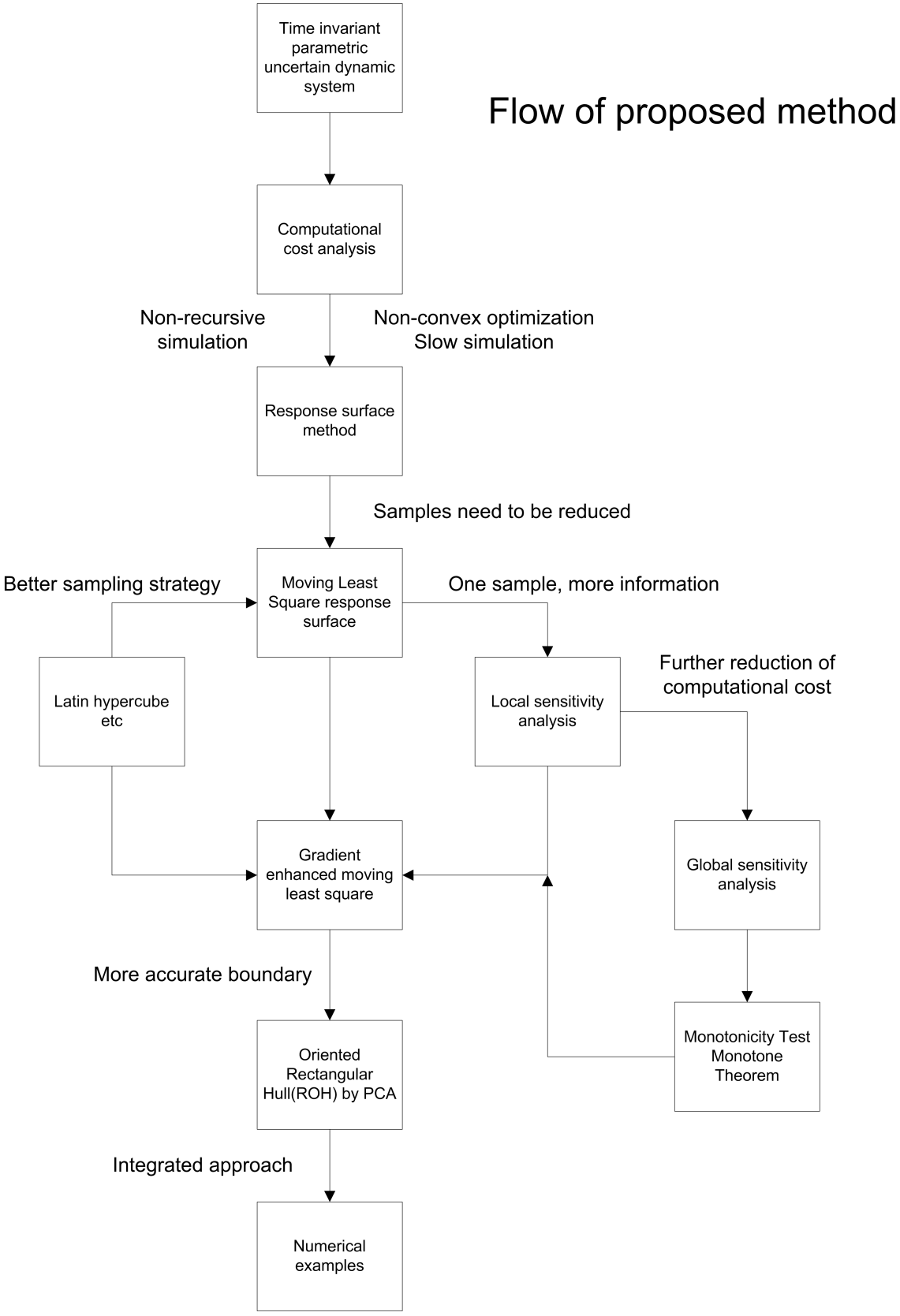


Figure 3.21: Flow of proposed method

Chapter 4

Reachable Set Approximation for Hybrid System Verification

Hybrid systems, which contain both discrete and continuous dynamics, have attracted a lot of attention recently [3, 12]. Reachable set computing is a basic problem in hybrid system analysis concerned with representing and computing all possible states that the continuous dynamics of a hybrid system can reach from a given set of initial conditions. This analysis has become a key method in verifying the correctness of a hybrid system [3, 23].

In this chapter, the concept of hybrid system and hybrid system verification is first introduced and then the reachable set computing/approximation problem is formulated as simulation of dynamic system with uncertain initial conditions. The computational burden of current methods for reachable set approximation, such as polyhedral approximation is studied. It shows that these methods involve global optimization techniques that embedded numerical simulation of the dynamic system response into the routine for evaluating the objective function. The search space is the entire uncertain initial state in $\mathbb{R}^n$. It is general but computationally expensive, and thus not applicable if simulation of the system is already computationally burdensome. The

applicability of some existing approaches that avoid the global optimization problem by employing the fundamental inequality theorem are shown to be very limited.

To reduce the computational burden, we first prove that the boundary of the reachable set is formed only by the trajectories from the boundary of the initial state region. This result reduce the search space from $\mathbb{R}^n$ to $\mathbb{R}^{n-1}$. For more complicated system, the method proposed in previous chapter is used to solve the global optimization problem more efficiently. Finally, it is shown that it will be more efficient and accurate if GEMLS is integrated with Principal Component Analysis (PCA) to find an oriented rectangular hull for reachable set representation and approximation.

4.1 State-dependent (threshold) Events Driven Hybrid Systems

Dynamical systems that are described by an interaction between continuous and discrete dynamics are usually called *hybrid systems*. Continuous dynamics usually may be represented by ODEs and discrete dynamics can be represented as a finite-state automata, with state $\boldsymbol{q}$ taking values in some finite set $\boldsymbol{Q}$, where transitions between different discrete states are triggered by suitable values of an input variable, $\boldsymbol{v}$. When the input $\boldsymbol{u}$ to the continuous dynamics is some function of the discrete state $\boldsymbol{q}$ and, similarly, the value of the input $\boldsymbol{v}$ to the discrete dynamics is determined by the value of the continuous state $\boldsymbol{x}$, a hybrid system arises [53]. A simple hybrid system can be

shown in the following example [53].

Example 4.1 A very simple manual car model can be expressed as:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(a, q) \end{aligned}$$

where x_1 is the position and x_2 is the velocity, a is the acceleration input which is a function of $q \in \{1, 2, 3, 4, 5, -1, 0\}$, the gear shift position. In this system, x_1 and x_2 are the continuous states and q is the discrete state. Clearly, the discrete transition (shift position) affects the continuous states and the continuous states (speed) will determine the transition of the discrete states.

One way to study the hybrid systems is to treat them as continuous systems with discrete switching events. Basically, switching events in such systems can be *state-dependent* or *time-dependent* events.

In this chapter, we focus on how the approach developed Chapter 3 can be used for the verification of state-dependent hybrid systems. For this kind of system, the continuous state space is partitioned into a number of operating regions by means of a family of switching surfaces. In each of these regions, a continuous dynamical system represented by a set of ODEs is given. Whenever the system trajectory hits a switching surface, the continuous system switches to a set of new ODEs. This type of hybrid system is called *threshold-event-driven hybrid systems (TEDHS)* [18] or *state-dependent events driven hybrid system systems* (SEDHS) [53]. Such systems can be illustrated by Figure-4.1 [18]. This system consists three types of interconnected sub-

systems: 1) switched continuous systems with discrete piecewise constant inputs that select the continuous dynamics and continuous outputs, 2) threshold event generators that take the continuous outputs of the switched continuous systems and generate events when they cross certain thresholds (or switching surfaces), and 3) finite state machines that are purely discrete transition systems with a finite number of states. The state transitions are triggered by the event outputs from the threshold event generators. The discrete outputs of the finite state machines, in turn, determine the dynamics of the switched continuous systems. TEDHS are attractive from the modeling perspective as they directly support block diagram modeling in which a system can be easily constructed by interconnecting the inputs and outputs of various subsystems [18].

4.2 Verification of Hybrid System

Verification of a hybrid system refers to methods for determining whether or not given properties (specifications) are true for a given model of a dynamic system. In general, there are two approaches to verification: *Theorem proving* and *model checking*. Theorem proving aims at inferring/contradicting a specification for a model using the methods of logical proof systems, where model checking approach uses the state-transition relation in iterative computations to arrive at the set of states for which the specification is true [72]. Model checking is an algorithmic technique and has a close relationship to the simulation of uncertain systems. The following batch reactor system example used

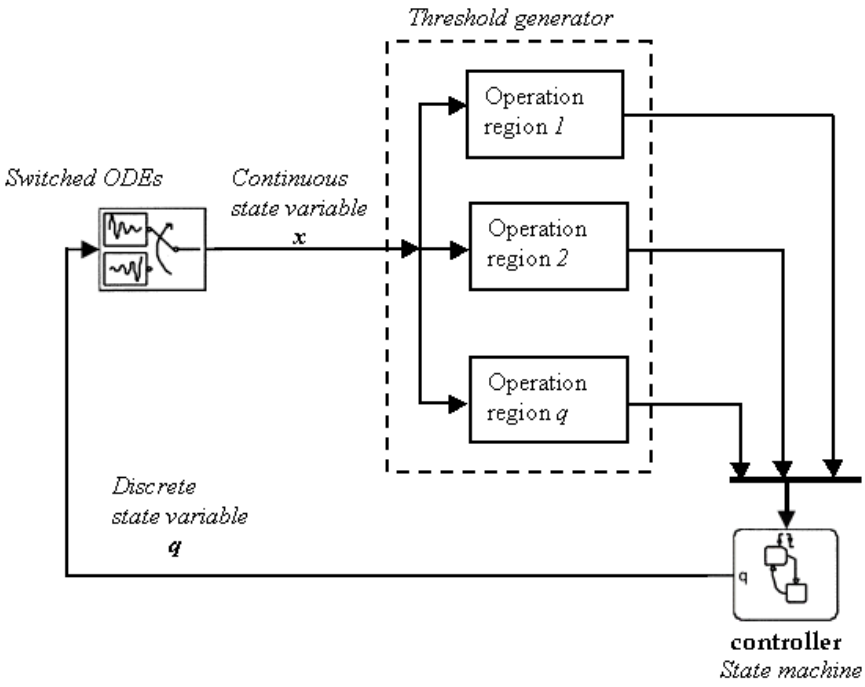


Figure 4.1: State-dependent events driven hybrid systems, adapted from [18].

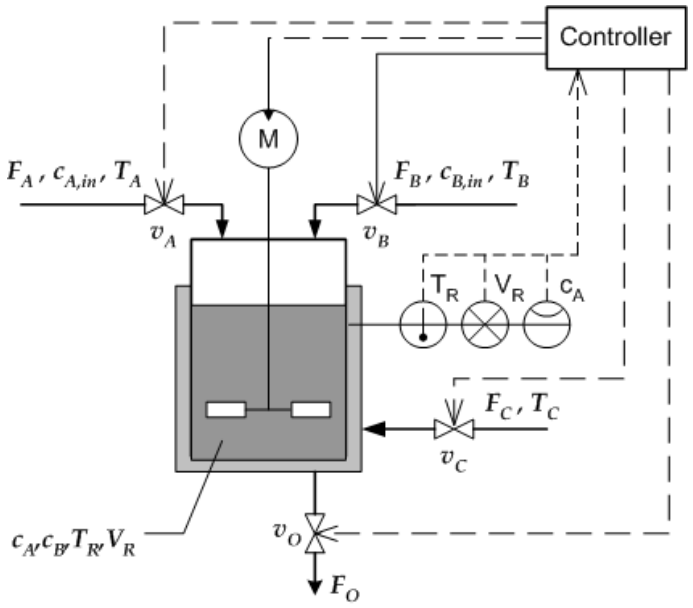


Figure 4.2: The batch reactor system, adapted from [72].

in [72] is borrowed to illustrate the model checking of hybrid systems.

Example 4.2: Verification of a hybrid system: the batch reactor system: As shown in Figure-4.2, the reactor is filled by two liquid streams F_A and F_B with temperature T_A and T_B and concentration $c_{A,in}, c_{B,in}$ of two dissolved substances A and B. The streams can be controlled through the valves v_A and v_B in the inlet pipes. The stirred content of the reactor is cooled by a cooling jacket. The supply of cooling water is switched on by opening valve v_C. Cooling is necessary since an exothermic chemical reaction $2A + B \rightarrow D$ leads to an increase of the reactor temperature T_R. The reaction product can be discharged through the valve v_O which is controlled by a discrete controller. Measurements of the temperature T_R, the liquid volume V_R, and the concen-

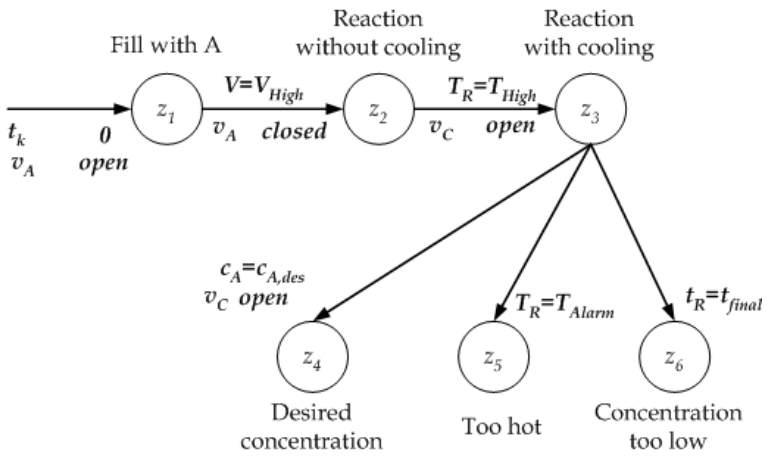


Figure 4.3: Operation procedure of the batch reactor, adapted from [72].

tration c_A indicate whether these variables exceed specific thresholds or not. The system equations and variables are,

$$\frac{dc_A}{dt} = \frac{s_1 \cdot F_A}{V_R}(c_A^{in} - c_A) - 2 \cdot r$$
$$\frac{dT_R}{dt} = \frac{s_1 \cdot F_A}{V_R} \cdot (T_A - T_R) - \frac{s_2 \cdot k_C \cdot A_C}{\rho \cdot c_P \cdot V_R} \cdot (T_R - T_C) - \frac{H_R \cdot \tau}{\rho \cdot c_P}$$
$$\frac{dV_R}{dt} = s_1 \cdot F_A, \frac{dt_R}{dt} = 1$$
$$r = c_A^2 \cdot k_0 \cdot e^{d\frac{-E_A}{R_m \cdot T_R}}, A_C = \frac{\pi}{4} \cdot D_R^2 + \frac{4}{D_R} \cdot V_R$$

$s_1 \in [0, 1]$- switch the valve v_A to close or open. $s_2 \in [0, 1]$- switch the valve v_C to close or open.

The production procedure can be shown as a transition model (state machine) as in Figure-4.3. Initially, assume one half of the reactor volume is already filled with solution B (and v_B is closed). In the first step (denoted by z_1), valve v_A is opened to supply the solution A until the volume V_R reaches

an upper limit V_{High}. The chemical reaction leads to an increase of the temperature (state z_2) such that T_R eventually reaches a threshold $T_R = T_{High}$. From state z_3 (reaction with cooling) three different states can be reached: the 'normal' operation is that the concentration c_A dropped down to $c_{A,des}$ corresponding to a sufficiently high concentration of the product D (c_D is high enough), and the reactor is emptied through valve v_O. If, alternatively, the temperature increases further to an upper threshold T_{Alarm}, the state z_5 is reached. (Note that T_R can show an over-shooting behavior when v_C is opened.) As a third possibility, a specified reaction time t_{final} can elapse before the desired concentrations are reached and the procedure terminates in state z_6. The reaction time is measured by a clock t_R, which is reset when valve v_A is opened.

The states z_5 and z_6 should be excluded from the course of operation, because these two states means failures of the processes and are not desired states. A discrete controller has to be designed such that is switches the valves v_A, v_C, and v_O in order to ensure that the operation always ends in state z_4, the desired state. The objective of verification for this system is: 1) to determine if the temperature threshold $T_R = T_{High}$ is chosen appropriately to guarantee T_{Alarm} is never exceed, 2) to ensure that the desired product concentration c_D (*or* $c_{A,des}$) is reached (for which T_{High} must not be chosen too low) within the specified reaction time t_{final}, and **given certain region of initial states because of uncertainties in the initial operating conditions**. For example, we want to verify that the system will

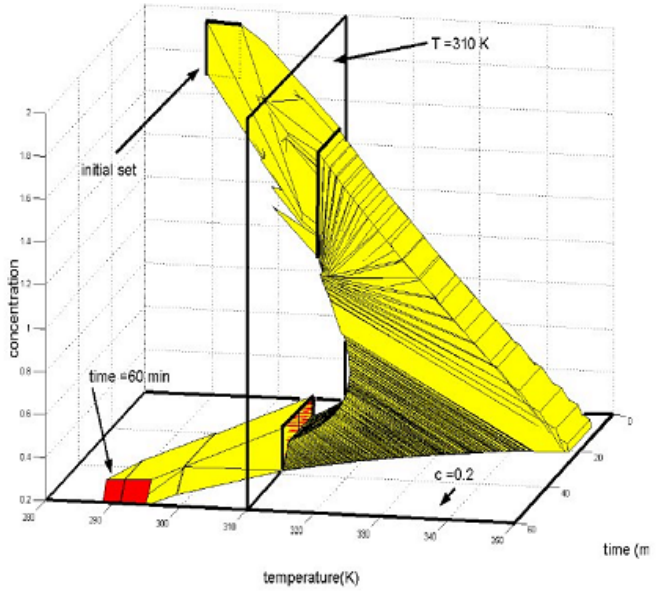


Figure 4.4: Verification of the batch reactor system, adapted from [20].

work properly (within 60 minutes, $c_A < 0.2$ with $T_{High} = 310K$) if the initial concentration of liquid A is in the interval $[1.7, 1.9]unit$ and the initial tank temperature is in the interval $[288, 295]K$. Thus, the verification of the system can be converted to verify that if the system will go from the an uncertain initial region $R_0(c_{A0} \in [1.7, 1.9], T_{R0} \in [288, 295])$ to a final region $R_f(c_A \in [0, 0.2], t_f = 60min)$. A verification result given by using the Checkmate tool [20] is shown in Figure-4.4. It showed that with $T_{High} = 310K$, at $t_R = 60min$, $0.2 < c_A < 0.3$. This example clearly shows that verification of hybrid system is equivalent to simulation of parametric uncertain dynamic system with only uncertain initial conditions. Thus, the techniques developed

in the previous chapter can be applied for hybrid system verification.

4.3 Reachable Set Approximation

As shown in Figure-4.1 and Figure-4.4, for hybrid systems, the continuous state space $\mathbb{R}^n$ is divided into several operating regions controlled by different continuous systems. Switching between different continuous systems is determined by whether the state of the system enters into these regions. Verification of a hybrid system establishes whether a continuous system enters a certain region with a given uncertain initial state region. This problem is called *reachable set computing* and is defined by the following.[1]:

For a continuous dynamic system,

$$\begin{aligned} &\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{t}) \\ &\mathbf{x}_0 \in \mathbf{X}_0 \subseteq \mathbf{R}^n, \end{aligned} \tag{4.1}$$

where $\mathbf{X}_0$ is a set of initial conditions. The reachable set of the above dynamic system at time t is defined as,

$$R_t(\mathbf{X}_0) = \{\, \mathbf{x}_f |\, \mathbf{x}_f = \mathbf{x}(\mathbf{x}_0, t), \forall \mathbf{x}_0 \in \mathbf{X}_0 \}\,. \tag{4.2}$$

The reachable set from initial time t_0 to final time T is the union of all the reachable sets from t_0 to T, which is defined as,

$$R_{[t_0,T]} = \cup_{t\in[t_0,T]} R_t(\mathbf{X}_0) \tag{4.3}$$

[1]Most literatures consider only uncertain initial condition when discussing reachable set computing. Unless otherwise state, we follow these literatures.

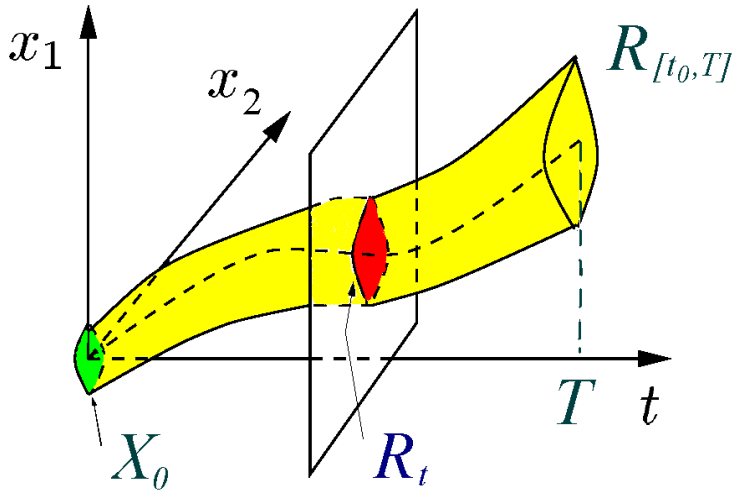


Figure 4.5: Reachable set computing problem

The reachable set in the time interval $R_{[t_1,t_2]}$ is also called the *flow pipe* from $\mathbf{X}_0$ in the time interval $R_{[t_1,t_2]}$ [20]. Figure-4.5 illustrates the reachable set of a dynamic system. It can be found that reachable set $R_{[t_0,T]}$ equals to the performance tube described in the previous chapter. R_t is a cross section of the performance tube at time t. Clearly, reachable set computing problem is a special case of simulation of parametric uncertain system with only uncertain initial conditions. Thus the techniques developed in previous chapters can be applied directly to solve this problem. However, due to its special character, some new techniques will be introduced in this chapter for reachable set computing.

Because of its critical role in the verification of hybrid systems, many methods to compute the reachable set have been developed. Meanwhile, as stated before, finding the exact reachable set of a general nonlinear system

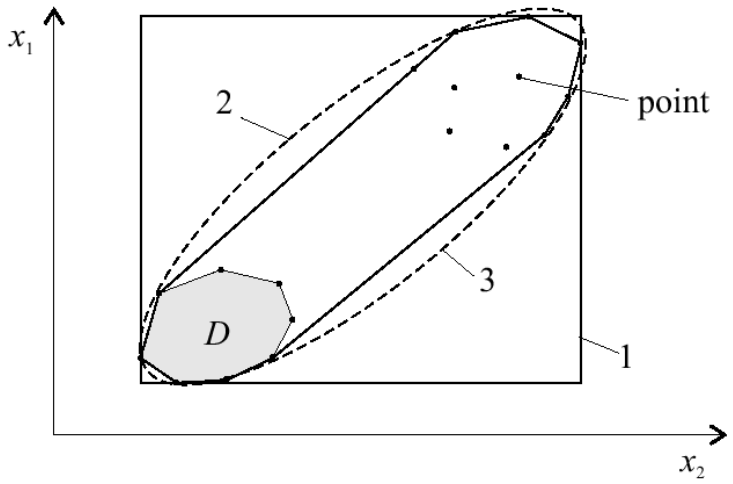


Figure 4.6: Geometry to represent a set:1-hyper rectangular 2-hyperellipsoid, 3-convex hull, adapted from [76]

is extremely difficult. As a result, all available tools focus on finding various types of approximations of R_t, which is denoted as $\tilde{R}_t$.

Generally, most of the existing methods to get $\tilde{R}_t$ adopt a two step strategy [58]. The first step is to choose a simplified parametric geometry to represent the reachable set. Unlike simulation of parametric uncertain system, in which case the outputs are often represented by a hyper-rectangle, reachable sets need to be represented more precisely. Thus, besides hyper-rectangle, these parametric geometries include hyperellipsoid [29], or polyhedra (convex hull) [20]. Commonly used geometry is shown in Figure-4.6 [76]. Selection of the geometry to represent the reachable set is a trade-off between accuracy and computational cost. A comparison of different geometries used can be found in [76]. Once the shape of the geometry is selected, the next step is

to find the parameters that define the geometry so it will over-approximate the real reachable set, subject to minimizing the difference between the two. This is analogous to the optimization step in simulation of uncertain system. For reachable set computing problem, at this step, different assumptions are made on the right hand side of the ODE, $\mathbf{f}(\mathbf{x})$, to simplify the problem and lead to different types of hybrid models [58, 72]. For example, [9] studied the reachability of *timed automata* by letting $\mathbf{f}(\mathbf{x}) = \mathbf{c}, \mathbf{c} \in \mathbb{R}^n$ is constant. The verification tool HyTech [35] provided a method to verify *linear hybrid automata* in which the continuous dynamics $\mathbf{f}(\mathbf{x})$ can be specified by differential inclusion as $\mathbf{f}(\mathbf{x}) \in [C_{min}, C_{max}]$. Another tool called d/dt [23] can deal with hybrid systems with linear continuous dynamics in the form $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{u}$. These assumptions simplified the reachable set problem but also limited the applicability of these tools.

CheckMate [19, 20, 73] is a verification tool that can handle general nonlinear dynamics. An algorithm called flow pipe algorithm is used in this tool. It uses a sequence of convex polyhedra with n_f faces, as shown in Figure-4.7, to approximate the reachable set. Such polyhedra can be defined by a set of linear inequalities as,

$$\begin{array}{l} POLY(\mathbf{C}, \mathbf{d}) = \{\mathbf{x} | \, \mathbf{C}\mathbf{x} < \mathbf{d}\} \\ (\mathbf{C}, \mathbf{d}) \in R^{n_f \times n} \times R^{n_f} \end{array} \tag{4.4}$$

A minimum convex polyhedra is found by integrating the global optimization algorithms and numerical simulation of the dynamic system response together. Details of the flow pipe algorithm are in next section. If the operational region

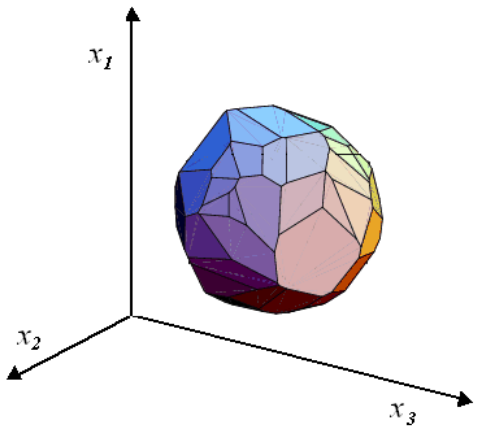


Figure 4.7: A polyhedra, from www.mathworld.com

of a hybrid system are represented by polyhedra, this class of hybrid system is called *Polyhedral-Invariant Hybrid Automata* (PIHA) [20]. For example, the above batch tank system can be represented as a PIHA system, as shown in in Figure-4.8 [20].

Nevertheless, like simulation of uncertain system, for existing methods the complexity of the computation restricts applicability to fairly low-order systems. These methods are computationally expensive due to the reasons stated in the last chapter. The global optimization problem in the flow pipe method [20], for example, involves many iterations of numerical simulation (solving system ODEs) to compute the objective function, which is computationally expensive. The verification of systems with five continuous variables with nonlinear dynamics usually requires hours of computation [72]. It will be more difficult to apply these methods to higher order systems where the number of faces in the polyhedral n_f will be high. As such, some methods

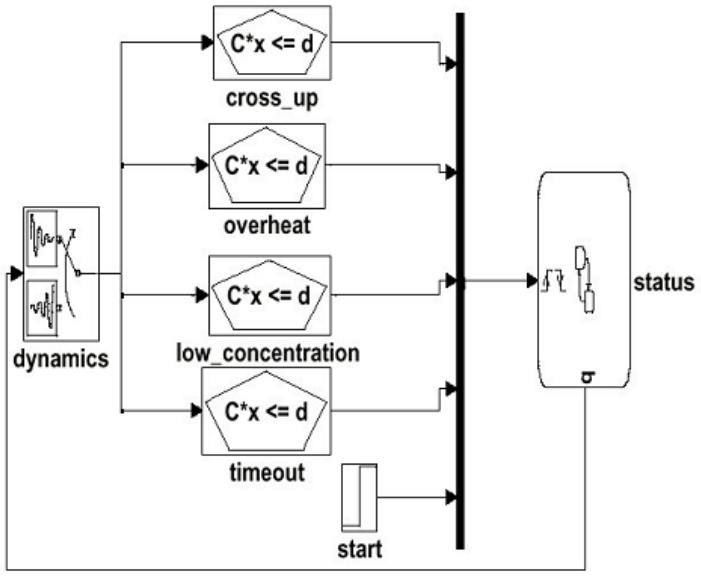


Figure 4.8: A polyhedra invariant hybrid system, adapted from [20]

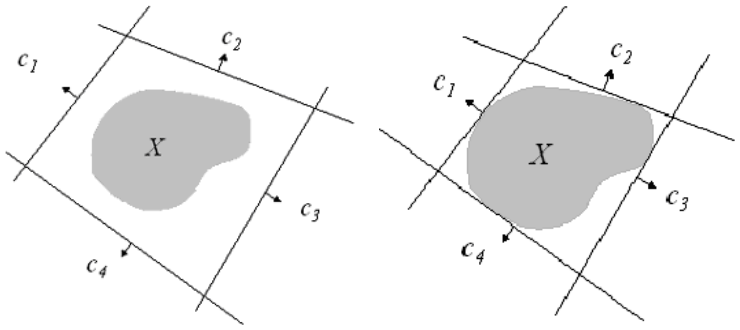
to overcome the computational burden based on the fundamental inequality theorem are reported, but as will be seen in the following, their practical applications may be very limited.

4.4 Computational Burden of Reachable Set Approximation of Nonlinear Systems

4.4.1 Flow Pipe Algorithm and its Computational Cost

In this section, the flow-pipe method is described and the sources of computational burden that limit the application of this kind of method are identified. Flow-pipe method and the software package CheckMate [19, 20] can be used for reachable set approximation of general nonlinear systems.

The original Flow-pipe method is used to approximate a reachable set segment $R_{[t_0,T]} = \cup_{t\in[t_0,T]} R_t(\mathbf{X}_0)$. By very little modification, it can be used to approximate R_t, the reachable set at a given time t. The algorithm to approximate R_t is illustrated in Figure-4.9. Assume the real reachable set is set $\mathbf{X}$



(a) Using polyhedra to represent the reachable set at time t

(b) Minimize the difference by optimization

Figure 4.9: Flow pipe algorithms (at time t)

and a convex polyhedra with n_f faces is used to over-approximate set $\mathbf{X}$. This polyhedra can be defined as,

$$\begin{array}{l} POLY(\mathbf{C},\mathbf{d}) = \{\mathbf{x} |\, \mathbf{Cx} < \mathbf{d}\} \\ (\mathbf{C},\mathbf{d}) \in \mathbb{R}^{n_f \times n} \times \mathbb{R}^{n_f} \end{array} \tag{4.5}$$

where $\mathbf{x}$ is any possible state variable vector at time t. Each row $\mathbf{c}_j^T, j = 1, ..., n_f$ of the matrix $\mathbf{C}$ is an unit normal vector to the j^{th} face of the polyhedra, as shown in Figure-4.9(a). $\mathbf{C}$ can be called the direction matrix. The elements of vector $\mathbf{d}$ are constants. A very close approximation of $\mathbf{X}$ can be given, with the number of faces of the polyhedra growing to a large number which will be hard to handle. Thus determination of matrix $\mathbf{C}$ is a trade-off

between computational complexity and accuracy. A effective method based on singular value decomposition (SVD) to find $\mathbf{C}$ is given in [76]. The polyhedra found by this method is called *oriented rectangular hull* and will be discussed later.

Once the direction matrix is selected, the value of the element of vector $\mathbf{d}$ determines the volume of the polyhedral. To get an accurate approximation of $\mathbf{X}$, a polyhedra with minimum volume that cover $\mathbf{X}$ should be found, and the following optimization problem will arise,

$$\begin{array}{ll} \min\limits_{\mathbf{d}} & \text{volume}[POLY(\mathbf{C}, \mathbf{d})] \\ \text{s.t.} & R_t(\mathbf{X}_0) \subseteq POLY(\mathbf{C}, \mathbf{d}) \end{array} \tag{4.6}$$

Solution of this equation is denoted as $\mathbf{d}^*$. Eq. (4.6) is equal to the optimization problem,

$$\begin{array}{ll} \max\limits_{\mathbf{x}} & \mathbf{c}_j^T \mathbf{x} \\ \text{s.t.} & \mathbf{x} \in R_t(\mathbf{X}_0) \end{array} \tag{4.7}$$

Using the definition of $R_t(\mathbf{X_0})$, Eq. (4.7) can be rewritten,

$$\begin{array}{ll} \max\limits_{\mathbf{x}} & \mathbf{c}_j^T \mathbf{x} \\ \text{s.t.} & \mathbf{x}(t) = \mathbf{x}_0 + \int_0^t \mathbf{f}(\mathbf{x}, \tau) d\tau, \;\; \mathbf{x}_0 \in \mathbf{X}_0 \end{array} \tag{4.8}$$

The minimized polyhedra given by above equation is the approximation $\tilde{R}_t$.

Comparing Eq. (3.1) and Eq. (4.8), it is found that the flow pipe algorithm is actually a direct global optimization method for parametric uncertain system simulation. Thus, it inherits those problems that lead to high computational cost stated in Chapter 3, such as,

1. Many numerical simulations ($\mathbf{x}(t) = \mathbf{x}_0 + \int_0^t \mathbf{f}(\mathbf{x}, \tau) d\tau$) must be embedded into the optimization routine.

2. Non-recursive simulation. Simulations embedded in the optimization routine must always start from $t = 0$.

3. Increasing the number of the faces (n_f) of the polyhedra will increase the accuracy of the approximation as well as the computational cost, since in each direction an optimization problem is to be solved.

To overcome the above problems, [20] proposed a bounding method based on the theorem of fundamental inequality to avoid applying global optimization algorithms to solve Eq. (4.8). Other studies [6] have relied on this theorem in a different manner to approximate the reachable set of non-autonomous systems. This method in [20] is referred to as non-recursive and the method in [6] is a recursive method. However, it will be shown that both methods are not very applicable to practical problems.

4.4.2 Limitation of the Theorem of Fundamental Inequality

The fundamental theorem applies to the general theory of differential equations and can be stated as follows [39]:

Definition 4.1: A number L is a *Lipschitz constant* with respect to x for a function $f(t, x)$ defined on a region A of R^2 (the t,x-plane) if,

$$|f(t, x_1) - f(t, x_2)| \leq L\,|x_1 - x_2|, \text{ for all } (t, x_1), (t, x_2) \text{ in } A.$$

The Fundamental Inequality Theorem: *If, on a rectangle* $R = [a, b] \times [c, d]$*, the differential equation* $x' = f(t, x)$ *satisfies a Lipschitz condition with*

respect to x, with Lipschitz constant $L \neq 0$*, and if* $u_1(t)$*and* $u_2(t)$ *are two approximate solutions, piecewise differentiable, satisfying,*

$$|u_1'(t) - f(t, u_1(t))| \leq \varepsilon_1$$
$$|u_2'(t) - f(t, u_2(t))| \leq \varepsilon_2$$

for all $t \in [a, b]$ *at which* $u_1(t)$ *and* $u_2(t)$ *are differentiable; and if for some* $t_0 \in [a, b], \quad |u_1(t_0) - u_2(t_0)| \leq \delta$*; then for all* $t \in [a, b]$*,*

$$|u_1(t) - u_2(t)| \leq \delta e^{L|t-t_0|} + \left(\frac{\varepsilon_1 + \varepsilon_2}{L}\right)\left(e^{L|t-t_0|} - 1\right) \tag{4.9}$$

Based on the fundamental inequality theorem, the following lemma can be derived [20]:

Lemma 1: Let $\mathbf{f}(\mathbf{x})$ be Lipschitz in $\mathbf{x}$ on $\mathbf{M}$ with a Lipschitz constant L, where $\mathbf{M} \subset R^n$ is a open connect set. Let $\mathbf{x}_0$ and $\mathbf{x}_0^*$ be initial conditions such that $\mathbf{x}(t, \mathbf{x}_0), \mathbf{x}(t, \mathbf{x}_0^*) \in \mathbf{M}$. Based on the fundamental theorem, the following inequality holds:

$$\|\mathbf{x}(t, \mathbf{x}_0) - \mathbf{x}(t, \mathbf{x}_0^*)\| \leq e^{Lt} \|\mathbf{x} - \mathbf{x}_0^*\| \tag{4.10}$$

This lemma shows that if $\mathbf{x}(t, \mathbf{x}_0^*)$ is a trajectory from initial time t_0 to time t of the system with initial condition $\mathbf{x}_0^*$, then at time t the trajectory from any arbitrary initial condition $\mathbf{x}_0$ with $\|\mathbf{x}(t, \mathbf{x}_0) - \mathbf{x}(t, \mathbf{x}_0^*)\| \leq \delta_{\mathbf{x}_0}$ must be contained in the γ-ball centered at $\mathbf{x}(t, \mathbf{x}_0^*)$, with,

$$\gamma = e^{Lt}\delta_{\mathbf{x}_0} \tag{4.11}$$

In other words, the system need be simulated only once with a nominal initial condition $\mathbf{x}_0^*$ to get $\mathbf{x}(t, \mathbf{x}_0^*)$. Assuming the face normal vector $\mathbf{c}_j$ is of

unit length, then the objective function $\mathbf{c}_j^T\mathbf{x}$ is bounded by $\mathbf{c}_j^T\mathbf{x}(t,\mathbf{x}_0^*) \pm \gamma$. i.e. $\mathbf{c}_j^T\mathbf{x}(t,\mathbf{x}_0^*) - \gamma \leq \mathbf{c}_j^T\mathbf{x}(t,\forall\mathbf{x}_0 \in \mathbf{X}_0) \leq \mathbf{c}_i^T\mathbf{x}(t,\mathbf{x}_0^*) + \gamma$. This result can be used to approximate the solutions of Eq. (4.8), and the time consuming procedures to find the real optimal results can be avoided.

This approach seems quite promising since a guaranteed over estimation of the reachable set is found without solving the optimization problem. However, further examination of the equation, $\gamma = e^{Lt}\delta_{\mathbf{x}_0}$, is discouraging because γ grows exponentially with ta. The reachable set given by this method will be too large.

The above method is a non-recursive method because the center of the γ ball $\mathbf{x}(t,\mathbf{x}_0^*)$ at time t is calculated from the initial time t_0, not from the previous time step $t-\Delta t$. This raises a question: can t be changed in Eq. (4.11) to Δt by iteration? If Δt can be made very small so that $\gamma = e^{L\Delta t}\delta_{\mathbf{x}_0}$ is less than the predefined tolerance, then the right side of Eq. (4.11) will not be a problem.

Reference [6] describes use of theorem of fundamental inequality in such a recursive manner to approximate the reachable set of a non-autonomous system. This method can be briefly described as follows [6]:

Lemma 2: Let $\Phi_f(t,\mathbf{x})$ be a trajectory of autonomous system $\dot{\mathbf{x}} = \mathbf{f}(t,\mathbf{x})$ and $\Phi_s(t,x,\mathbf{u})$ be a trajectory of system $\dot{\mathbf{x}} = \mathbf{f}(t,\mathbf{x}) + \mathbf{u}$, where $\mathbf{u}$ is bounded by mu. Based on the fundamental theorem, the following inequality holds:

$$\|\Phi_f(t,\mathbf{x}) - \Phi_s(t,x,\mathbf{u})\| \leq \frac{\mu}{L}(e^{Lt} - 1) \tag{4.12}$$

This theorem states that to approximate the reachable set of the non-autonomous system with input $\mathbf{u}$, we can appropriately expand the reachable set of the autonomous system by the amount given on the right hand side of Eq. (4.12). However, as shown this grows exponentially. However, Asarin, et al [6] proposed an iterative algorithm described as follows:

Let $P_t(\mathbf{X}_0)$ be the reachable set of an autonomous system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}), \mathbf{x}_0 \in \mathbf{X}_0$. Let $\tilde{Q}_t$ be the approximated reachable set of a non-autonomous system in the form, $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{u}(t), \|\mathbf{u}\| \leq \mu$, at time t. The set $\tilde{P}_{t+\Delta t}(\mathbf{X}_0)$ is first computed iteratively by taking Q_t as the initial region, $\tilde{P}_{t+\Delta t}(\mathbf{X}_0) \approx \tilde{P}_{\Delta t}(\tilde{Q}_t)$, rather than using $P_{t+\Delta t}(\mathbf{X}_0) = P_{\Delta t}(P_t)$. Note that in this step, error will be generated, as shown in Figure-4.10. Then $\tilde{Q}_{t+\Delta t}(\mathbf{X}_0)$ is expanded from $\tilde{Q}_{t+\Delta t}(\mathbf{X}_0)$ by a γ ball, as $\tilde{Q}_{t+\Delta t}(\mathbf{X}_0) = \tilde{P}_{t+\Delta t}(\mathbf{X}_0) + \gamma \approx \tilde{P}_{\Delta t}(\tilde{Q}_t) + \gamma$. In this case, we have,

$$\gamma = \frac{\mu}{L}(e^{L\Delta t} - 1). \tag{4.13}$$

Comparing this equation with Eq. (4.12), it appears Eq. (4.13) is quite promising since the right side is exponentially increasing with Δt other than t. If Δt is small enough, then γ could be a tight bound. However, this method is also very limited. In Figure-4.10, it can be found that the error due to the approximation, $\tilde{P}_{t+\Delta t}(\mathbf{X}_0) \approx \tilde{P}_{\Delta t}(\tilde{Q}_t)$, will propagate to the next time step and accumulates, even if γ is a constant. This is similar to the wrapping effect mentioned in Chapter 3. In fact, the *Hausdorff distance*[2] between the real

[2]Hausforff distance is a measure of the resemblance of two (fixed) sets of geometric

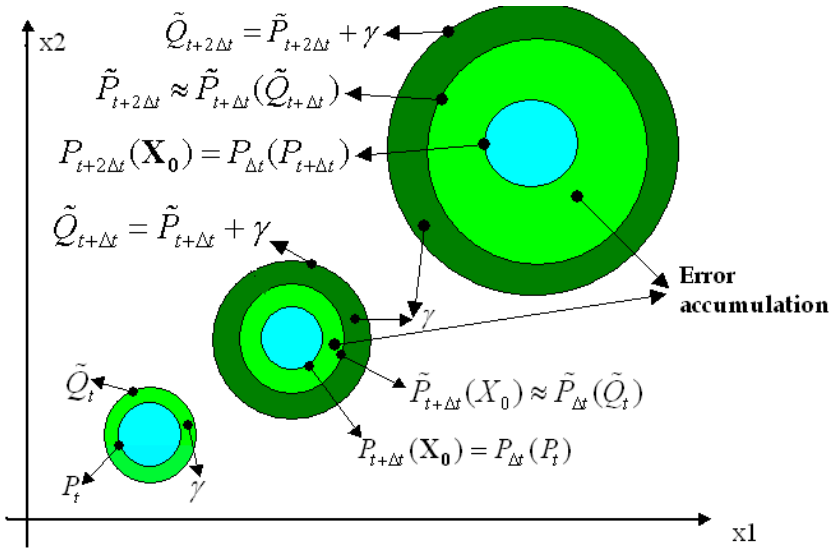


Figure 4.10: Accumulating of errors

reachable set $Q_{[0,T]}$ and the reachable set approximated by this method $\tilde{Q}_{[0,T]}$ is given as, $2\mu\Delta t e^{Lt}$ [6], which still grows exponentially.

The proceeding analysis shows that methods based on the fundamental inequality to reduce the computational burden can only be applied within a very short of period of time. This shortcoming seriously weakens the applicability of this method. Other methods must be found that solve the optimization problem effectively.

---

points $\boldsymbol{P}$ and $\boldsymbol{Q}$, defined as $H(\mathbf{P},\mathbf{Q}) = \max\{\max_{\mathbf{a}\in\mathbf{P}} \min_{\mathbf{b}\in\mathbf{P}} d(\mathbf{a},\mathbf{b}), \max_{\mathbf{a}\in\mathbf{Q}} \min_{\mathbf{b}\in\mathbf{P}} d(\mathbf{a},\mathbf{b})\}$, $d(.,.)$ is the distance metric, usually the Euclidean distance [2]

4.4.3 Boundary Theorem

Reachable set computing is a special case of simulation of parametric uncertain systems since only the initial conditions are uncertain. Thus, a theorem that can reduce its computational cost can be derived.

Boundary Theorem: *For a locally Lipschitz ODE system* $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$ *with initial conditions set* $\mathbf{X}_0$ *and* $\mathbf{f}(\mathbf{x}, t)$ *is not an explicit function of* $\mathbf{X}_0$*, the boundary of the reachable set* $R_T(\mathbf{X}_0)$ *of this system, denoted as* $\boxed{R_T(\mathbf{X}_0)}$ *can be acquired by simulating the system from each point on the boundary of the initial condition set, denoted as* $\boxed{\mathbf{X}_0}$*;i.e.*

$$\boxed{R_t(\mathbf{X}_0)} = \left\{ \mathbf{x}_f | \, \mathbf{x}_f = \mathbf{x}(\mathbf{x}_0, t), \forall \mathbf{x}_0 \in \boxed{\mathbf{X}_0} \right\}.$$

Proof: First, claim that for a locally Lipschitz ODE system, two trajectories from two different initial conditions will never intersect. Refer to Figure-4.11 and assume that two trajectories from different initial conditions intersects at point (X_m, T_m), then chose X_m as the initial condition and there will be two solutions to an IVP problem with X_m as the initial condition. This is contradicted to the unique existence theorem of ODE, thus the claim is true. Because of this theorem, it is obvious that the trajectories from the boundary of the initial condition region will form the boundary of all the trajectories from the initial condition region. If one trajectory originating from a point inside the initial condition exists outside the boundary, it must intersect with one of the trajectories, as shown in Figure-4.12. However, this is contradictory to the claim proven above and thus is impossible. The theorem is proven. ■

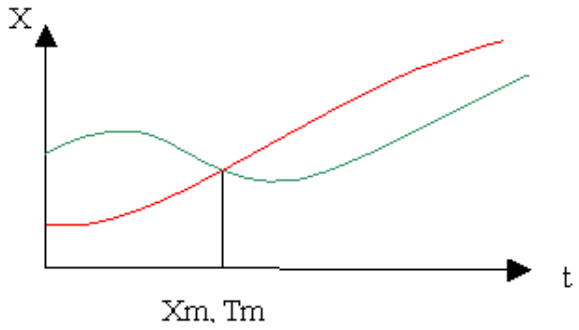


Figure 4.11: Any two trajectories will not intersect

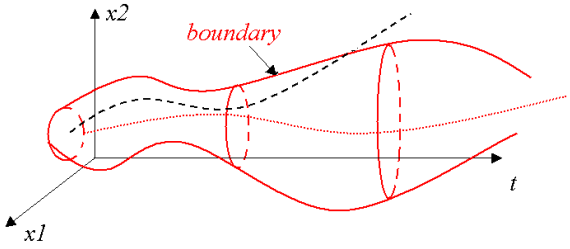


Figure 4.12: Boundary theorem

This theorem shows that the search space to find the polyhedra that approximate the reachable set can be reduced from $\mathcal{R}^n$ to $\mathcal{R}^{n-1}$ since the boundary of R_t originates from the boundary of $\mathbf{X}_0$; i.e., Eq. (4.8) can be rewritten to,

$$\begin{array}{ll} \max\limits_{\mathbf{x}} & \mathbf{c}_j^T\mathbf{x} \\ \text{s.t.} & \mathbf{x}(t) = \mathbf{x}_0 + \int_0^t \mathbf{f}(\mathbf{x},\tau)d\tau,\ \ \mathbf{x}_0 \in \boxed{\mathbf{X}_0} \end{array} \tag{4.14}$$

If a Monte Carlo method is used for reachable set computing, this theorem shows that in order to find the reachable set, sample points only need to be taken from the boundary of the initial condition set.

For linear systems, computational costs can be further reduced. This theorem shows that if the initial condition set of a linear system is a polyhedra, then the reachable set of this system can be acquired by simulating the system only from the corner of the initial polyhedra.

Boundary theorem: Linear system. If a linear time invariant system, $\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}$, with an uncertain initial condition set as a polyhedra, $\{\mathbf{x}^0 | \mathbf{x}^0 = \mathbf{c}^T\mathbf{x} \leq \mathbf{d}^0\}$, then the reachable set of this system can acquired by simulating from the corner of the initial polyhedral.

Proof: Points at the boundary of the initial condition satisfy, $\left\{\mathbf{x}_b^0 | \mathbf{c}^T\mathbf{x}_b^0 = \mathbf{d}_b^0\right\}$, thus we have, $\mathbf{x}_b^0 = (\mathbf{c}^T)^{-1}\mathbf{d}_b^0$. From boundary theorem, it is known that the boundary of the reachable set is,

$$\boxed{R_t} = e^{\mathbf{A}t}\mathbf{x}_b^0 = e^{\mathbf{A}t}(\mathbf{c}^T)^{-1}\mathbf{d}_b^0 = \xi^T\mathbf{d}_b^0 \tag{4.15}$$

If ξ^T is a direction matrix, then Eq. (4.15) shows that R_t of linear system is still a polyhedra. The corner of the initial set will evolve to the corner of R_t.

Thus the corner of R_t is found by simulating from the corner of the initial polyhedra. Once the corner of polyhedra R_t is known, $R - t$ is determined. ■

4.4.4 Oriented Rectangular Hull and RSM

When a hyper-rectangle is used to approximate an unknown set ,**S**, the rectangle can be of any direction. One generally used case is an axes-parallel rectangle. However, there exits an 'optimal' direction so that the size of the rectangle can be minimized if the set **S** is directionally strong, as shown in Figure-4.13. It is very hard to determine the real optimal direction since the

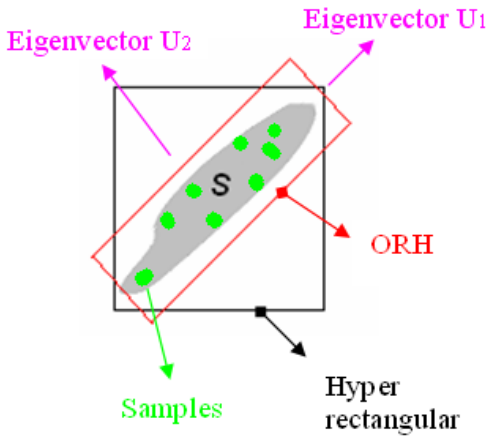


Figure 4.13: Oriented Rectangular Hull representation

shape of set **S** is not known and there are infinite elements in **S**. However, a better direction based on a subset of **S**, which is composed of some samples in **S** can be found. The preferred directions can be found by using the Principal Component Analysis (PCA) [44], as proposed in [76]. The idea is to take some

samples in $\mathbf{S}$ by simulation and use PCA to find the principle axes of $\mathbf{S}$. The direction vectors representing the principle axes found by PCA will form the direction matrix $\mathbf{C}$. Once the direction matrix $\mathbf{C}$ is determined by PCA, then Eq. (4.8) needs to be solved to find the ORH with minimum size. In short, PCA decomposes a matrix in the form,

$$\mathbf{X} = \mathbf{USC}'. \tag{4.16}$$

This step can be done by singular value decomposition, where $\mathbf{X}$ is a $n \times n_s$ matrix, a column vector of $\mathbf{X}$ is a mean centered sampled state vector. $\mathbf{C}$ is the matrix that we are interested in. This matrix contains eigenvectors that are the optimal directions, as shown in Figure-4.13. For details of PCA, please see [44].

The step for finding the ORH to approximate a reachable set R_t by using PCA are:

1. Generate n_s sample points at the boundary of the initial condition set, i.e. $\mathbf{x}_{s0}^i, i = 1, 2, ..., n_s, \mathbf{x}_{s0}^i \in \boxed{\mathbf{X_0}}$.

2. Simulate the system from t_0 to t with $\mathbf{x}_{s0}^i, i = 1, 2, ..., n_s$. The set that contains all the corresponding system response $\mathbf{x}_s^i$ is denoted as $\mathbf{X}_s$.

3. Transformation: find the geometric center $\mathbf{X}_s^C$ of $\mathbf{X}_s$ and move the origin of the original state space to $\mathbf{X}_s^C$ so $\bar{\mathbf{X}}_s = \mathbf{X}_s - \mathbf{X}_s^C$.

4. Use singular value decomposition to find the direction matrix $\mathbf{C}$ as, $\bar{\mathbf{X}}_s = \mathbf{U} \times \mathbf{S} \times \mathbf{C}$, where $\mathbf{S}$ is a $n_s \times n_s$ diagonal eigenvalue matrix, $\mathbf{C}$ is the

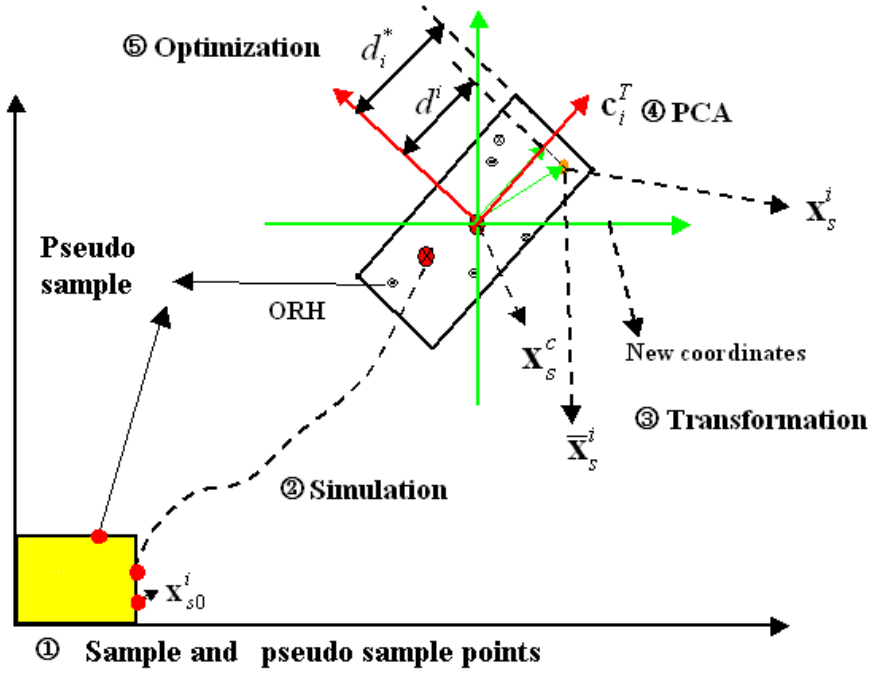


Figure 4.14: Integration of GEMLS1 and Oriented Rectangular Hull

matrix containing eigenvectors (or direction of the principle axes in the new coordinate system),

5. Optimization: solve $d_j^* = \max_{\mathbf{x}} \mathbf{c}_j^T \ \bar{\mathbf{x}}_i^*, \qquad i = 1, 2, \cdots, n_f$ $\bar{\mathbf{x}} \in R_t\left[\mathbf{X}_0\right] - \mathbf{X}_s^C$ and find the oriented rectangular hull determined by $\mathbf{C}$ and d_i^*.

The above procedure can be illustrated by Figure-4.14. This diagram shows that finding the direction of the ORH is a sample-based method. Thus, it will be more efficient if it is integrated with boundary theorem and GEMLS1. This is because: 1) Samples can be taken only from the points on the boundary of the initial set because of boundary theorem; 2) In step 1, in order to find

an accurate ORH, enough samples should be taken and this can be done by generating pseudo samples from sensitivity analysis; 3) In step 5, once $\mathbf{C}$ is found, the same samples and pseudo samples used to find ORH can then be used to calculate the observations of the objective function as, $d_j = \mathbf{c}_j^T \bar{\mathbf{x}}_s = {}^jO_s(\mathbf{x}_{s0})$, and then the response surface can be constructed to solve the optimization problem to find the ORH with minimal size. Notice that the geometric meaning of $d_j = \mathbf{c}_j^T \bar{\mathbf{x}}_s^j$ is the length of $\bar{\mathbf{x}}_s^i$'s projection on the j^{th} principle axes.

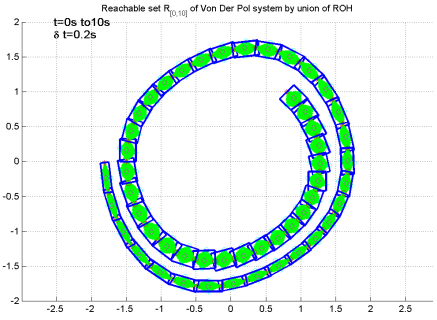
4.4.5 Numerical Example

Example 4.3 Benchmark: The reachable set of a 2-D Van der Pol system[3].

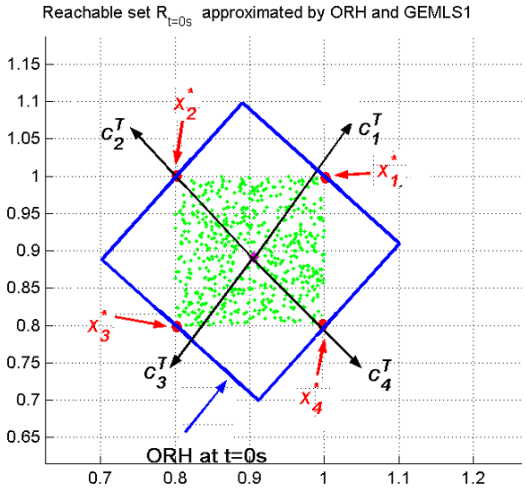
$$\begin{aligned}
&\dot{x}_1 = x_2 \\
&\dot{x}_2 = -.02(x_1^2 - 1)x_2 - x_1 \\
&\mathbf{X}_0 = \{0.8 \le x_1 \le 1, 0.8 \le x_2 \le 1\} \quad t \in [0s, 10s], \Delta t = 0.2s
\end{aligned}$$

is approximated by the approach described early. Random samples along the border of the initial condition set (a square) are taken (8 real samples+ 4 pseudo samples). The domain $\mathbf{X}_0$ is divided into 100 grids to evaluate the response surface. For comparison purposes, Monte Carlo simulation results based on 625 samples, taken in the square are also given, as shown in Figure-4.15. Figure-4.15(a) shows the reachable set from $t = 0$ to $t = 10s$ as a union of series ORHs. The time step is $\Delta t = 0.2s$. Figure-4.15(b) shows the reachable set at $t = 0s$. Notice that at $t = 0$, since the real reachable set (a rectangle) does not have a strong preferred orientation, the ORH found is not a very good

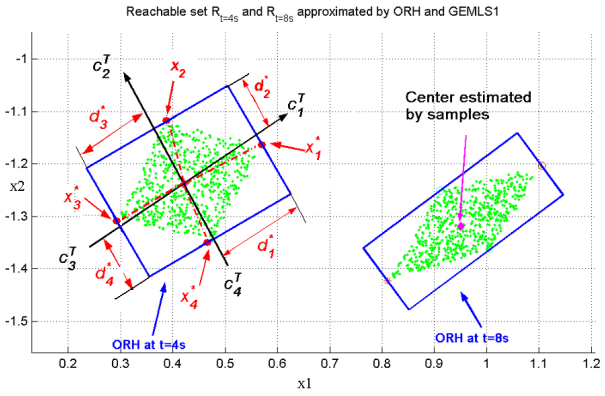
[3]This is a widely used benchmark problem, see [6, 19, 76]



(a) Reachable set given by union of ROH



(b) Reachable set at t=0s



(c) Reachable set at t=4,8 s

Figure 4.15: Reachable set of Van del Pol system given by GEMLS1 and ORH

approximation due the singularities in the covariance matrix used for PCA [76]. Due to the dynamics of the system, the shape of the reachable set became more oriented. Figure-4.15(c) shows the reachable set at $t = 4s, t = 8s$. It can be found that even with just 8 real samples, GEMLS1 can find a very accurate optimal location $\mathbf{x}_i^*$ as well as the corresponding optimal d_i^* that determines the size of the ORH. It happens that the reachable sets of this problem at different times are all polygons, so only four simulations originated from four corner values of the initial condition set are needed to determine the reachable set.

Example 4.4 Using ORH to approximate reachable set of parametric uncertain system. A more general definition of a reachable set is the performance tube of a parametric uncertain system that contains both uncertain parameters and uncertain initial conditions. In this case, the boundary theorem can not be applied. However, the integrated steps using ORH and GEMLS1 mentioned above still hold, taking samples inside the initial condition set and in the parameter space. Figure-4.16 shows the reachable set of the system used in Example 3.8, generated by ORH. Contrasting with the reachable set generated by hyper rectangle shown in Figure-3.20(c), it is found that ROH gives a more accurate approximation.

4.5 Summary

This chapter formulates the verification of a hybrid system as a reachable set approximation problem, which is a special case of simulation of para-

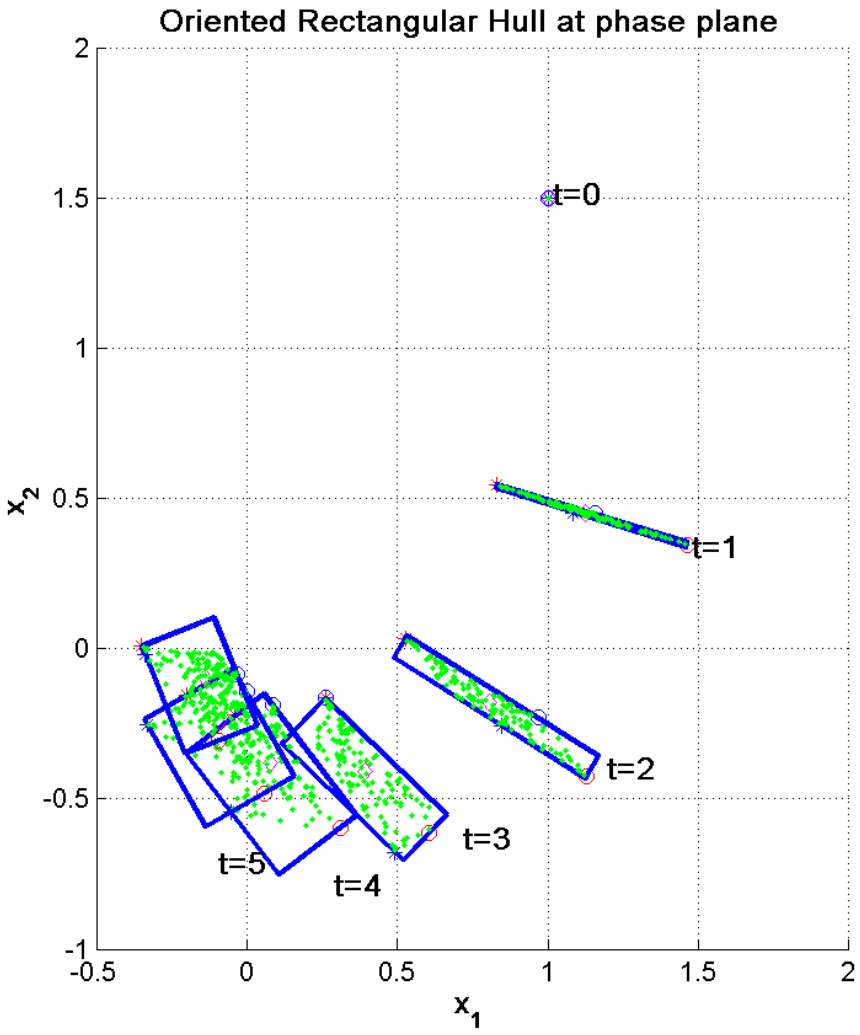


Figure 4.16: Oriented Rectangular Hull for reachable set approximation

metric uncertain system. The flow pipe algorithm is analyzed and shortcomings in using the fundamental inequality theorem to reduce the computational cost are pointed out. A boundary theorem that can reduce the search space from $\mathbb{R}^n$ to $\mathbb{R}^{n-1}$ is then derived. Finally, ORH and GEMLS1 are integrated together to approximate reachable sets with better efficiency and better accuracy.

Chapter 5

Fault Detection as Simulation of Parametric Uncertain System

This chapter treats the model-based fault detection problem as simulation of parametric uncertain systems. First, a fault free system is modeled as a parametric uncertain system whose parameters belong to a given bounded parameter set, which is called normal set. The performance of a fault free system is bounded by the boundary that can be acquired by using the approach described in previous chapters. A fault is defined when system parameters do not belong to the normal set due to malfunction or degradation. Once such fault(s) occur, the monitored system performance will extend beyond the system boundary predicted by the parametric uncertain model. A fault is reported whenever the monitored system performance interacts with the system boundary. Compared to conventional model-based fault detection methods that use fixed threshold, method used in this chapter use boundary as an adaptive threshold that can give better miss alarm and false alarm ratio.

5.1 Fault and Failure

Any complex system is susceptible to faults or failures. In most general terms a fault is any change in a system that prevents it from operating in the proper manner [79]. It is recognized that *fault detection* is more properly called *change detection* and that a fault can be either a failure in a physical component, or a change in system performance. These definitions may be formalized as follows [15]:

- Fault: Undesired changes in system parameters that degrades performance. A fault may not represent a component failure.
- Failure: Catastrophic or complete breakdown of a component or function. To be contrasted with a fault which may be a tolerable malfunction.
- Fault detection: A binary decision making process. Either the system is functioning properly, or there is a fault present.
- Fault isolation: Determination of source of a fault.

In this research, *fault is defined as a change or degrading of system parameter values away from a normal value set.* For example, when the internal resistance of a power supply is between 0.01Ω to 0.02Ω , the system is treated as working properly or *normal.* Whenever the internal resistance is out of this range, the system is treated as abnormal.

5.2 Fault Detection and Isolation (FDI)

For a complex system, with a number of potential fault states (system parameters), FDI is most often considered to be a multi-stage process:

1. Fault detection: firstly the fact that a fault has occurred (something wrong?) must be recognized.

2. Fault diagnosis

   - Fault isolation: Secondly the nature of the fault should be determined (what is wrong?) such that appropriate remedial action may be initiated.

   - Fault identification: Further information on a fault (How bad?) is required after isolation, such as magnitude, cause, type of the fault.

3. Fault accommodation: actions are to be taken to account for the fault (what to do with it?). This may vary from triggering an alarm or replacing a part.

This reach focuses on the fault detection problem and a simple discussion on fault isolation is given at the end of this chapter.

Fault detection systems have several major performance criteria. These criteria are [79]:

1. Missed alarms ratio. It is most important that the system does actually

detect all the faults that occur. The missed alarm ratio reflects how good the FD system at catching the faults.

2. False alarm ratio. A false alarm refers to when the FD system indicates a fault and the system is actually not faulty. It reflects how reliable the FD system is.

3. Detection delay. It is the delay between the appearance and the detection of a fault It measures the speed at which the detection and handling system operates when a fault happened.

An ideal FD system detects faults as soon as they appear to keep the monitored system safe and with neither missed alarms nor false alarms. A more realistic goal is to reduce these three indices to the minimum.

5.3 Fault Detection Methods

Basically, fault detection is done by comparing the performance of that system against some ideal references system, as shown in Figure-5.1 [4]. This ideal may be either another redundant, identical physical system, or a computer model. The difference of the two systems is called residual. When the residual is larger than certain threshold value, a fault occurs.

If the reference is another redundant physical system, the the fault detection is a hardware based fault detection system. In many cases this is not practical because of economical reasons or other kinds of constraints [15].

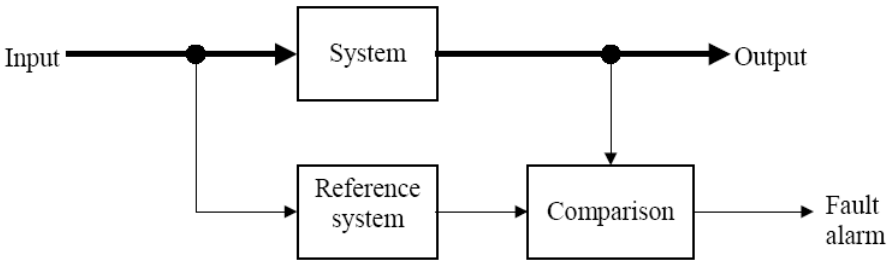


Figure 5.1: Redundancy based fault detection, adapted from [4]

An alternative to overcome these problems is the use of a model to generate the reference behavior, i.e. to perform analytical redundancy. This approach is called model-based fault detection.

The major advantage of the model-based approach is that no additional hardware components are needed in order to realize an FDI algorithm. A model-based FDI algorithm can be implemented in software on the process control computer. Furthermore, the measurement necessary to control the process are, in many cases, also sufficient for the FDI algorithm so that no additional sensors have to be installed [15].

A comparison of hardware vs model based redundancy approach is shown in Figure-5.2 [15].

5.4 FD as Simulation of Parametric Uncertain Systems

As mentioned before, a fault is defined as undesired change in system parameters that degrade performance. Thus, we can define fault detection as

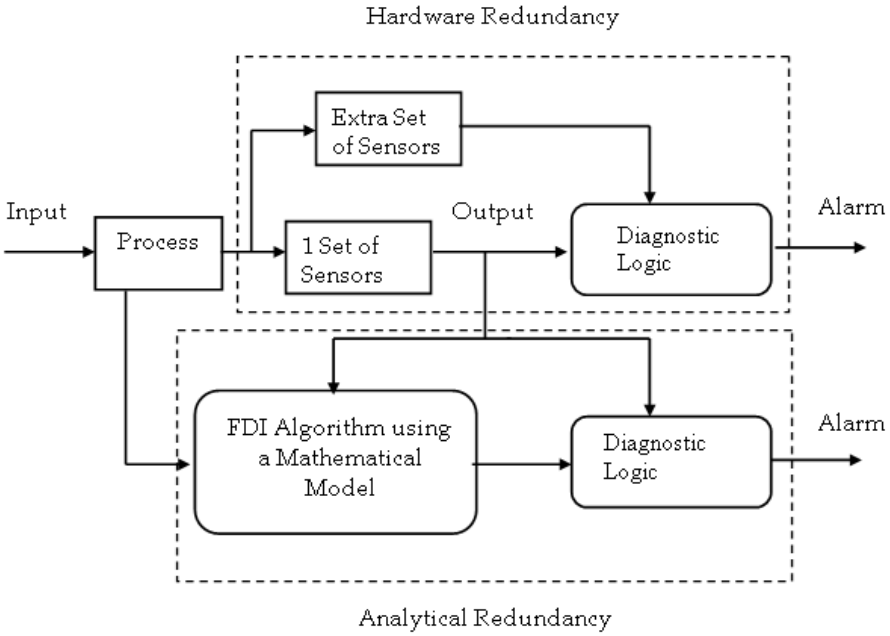


Figure 5.2: Hardware vs analytical redundancy, adapted from [15]

finding the change of parameters and transform FD to simulation of parametric uncertain systems. First, a fault free system is modeled as a parametric uncertain system whose parameters belong to a given bounded parameter set, which is called a normal set. The performance of a fault free system is bounded by the boundary that can be acquired by using the approach described in Chapter 3 and Chapter 4. A fault is defined when system parameters are not in the normal set, due to malfunction or degradation. Once such fault(s) occur, the monitored system performance will go beyond the system boundary predicted from the parametric uncertain model. A fault is reported whenever the monitored system performance interacts with the system boundary. This approach is similar to the approach reported in [4], in which the boundary is found by using interval analysis.

Such FD system is illustrated in Figure-5.3.

The advantage of modeling FD as simulation of uncertain systems are as follows:

1. Practically, the value of a component with acceptable quality is defined in a bounded set other than a value. For example, a resistor value is often given as $1K \pm 1\%$, which means the resistor is in good condition if its real value is between 990Ω and 1100Ω . The uncertain model of the system naturally reflects this fact.

2. Conventional model based fault detection compare a fixed thresholds with the residual generated to make decisions. This fixed threshold usu-

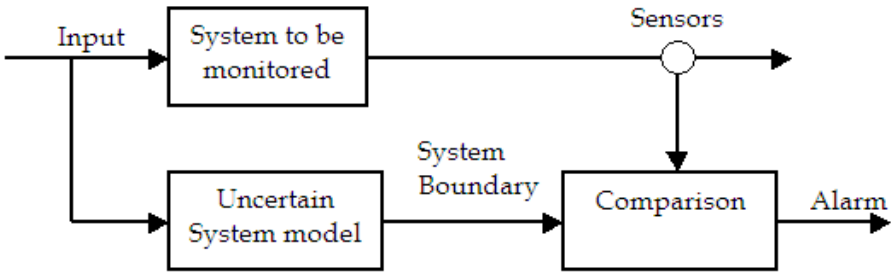


Figure 5.3: FD as simulation of uncertain system, adapted and modified from [4]

ally is not very useful for dynamic systems because of the changing operating point. In this case, it is better to compute a new threshold every time step. This is an adaptive threshold [4].

Boundary of the system can be treated as an adaptive threshold with changing values. The way it changes reflects the dynamics of the system and thus the miss alarm and false alarm ratio can be reduced. This can be shown in Figure-5.4 [4].

5.5 Numerical Example

5.5.1 The Motor-Pump-Pipe System

The following system is to be studied: a sub-unit in a process plant that maintains the liquid level of a reactant tank to be used by a Continuous Stirred Tank Reactor (CSTR). The system schematics are shown in Figure-5.5[1] as a

[1]This example is taken from the final exam of ME-397 Fall, Fault detection, given by Dr. Fernandez in Mechanical Engineering Department, UT Austin

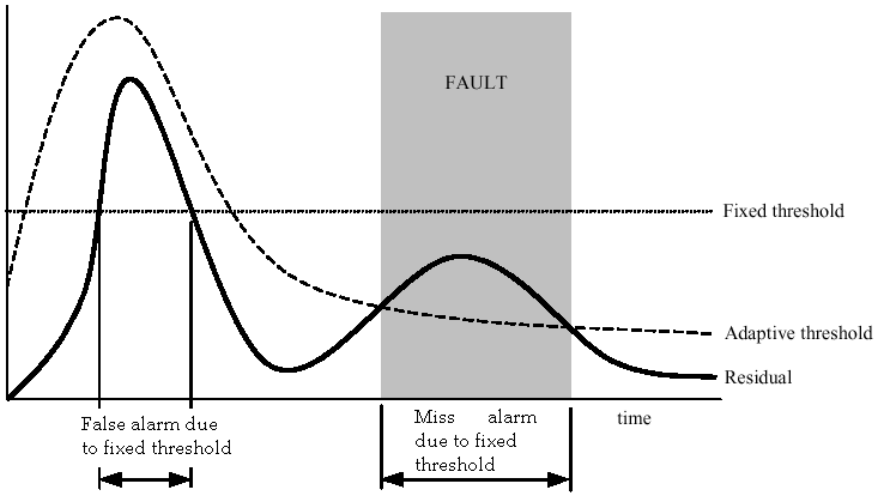


Figure 5.4: Boundary as adaptive threshold, adapted from [4]

simplified Motor-Pump-Hydraulic System Circuit. A DC motor controlled via voltage regulation is coupled to a centrifugal pump. The pump is associated with a hydraulic circuit. To regulate the speed of the the pump, a proportional (P) type controller is added to the system. To regulate the reactant level a Proportional-Integral (PI) type controller is used.

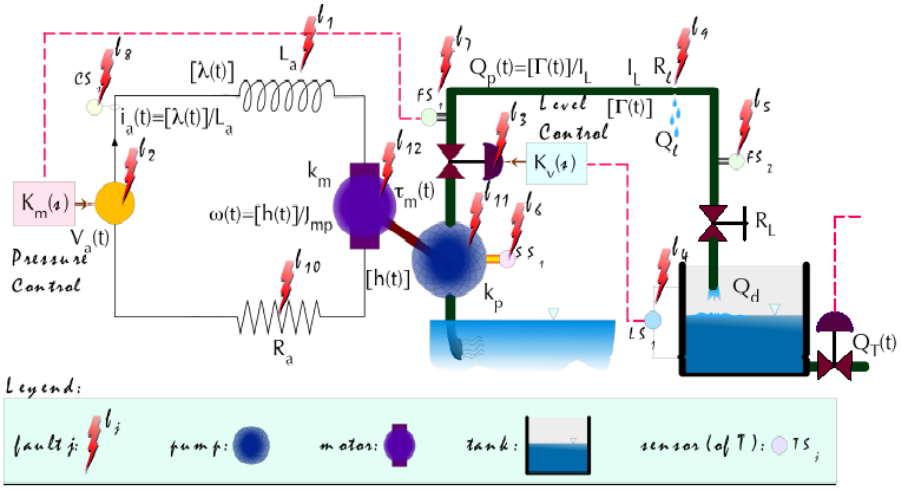


Figure 5.5: Motor-Pump-Pipe System.

5.5.2 System Model

The system equations of the Motor-Pump-Pipe system given in state space form are:

$$
\begin{aligned}
\dot{\lambda}(t) = \dot{x}_1(t) &= K_{p_1} \cdot (\omega_{ref}(t) - \frac{h(t)}{J_{mp}}) - \frac{R_a}{L_a}\lambda(t) - \frac{k_m}{J_{mp}}h(t) \\
\dot{h}(t) = \dot{x}_2(t) &= \frac{k_m}{L_a}\lambda(t) - \frac{R_{mp}}{J_{mp}}h(t) - \frac{k_p}{I_L}\Gamma(t) \\
\dot{\Gamma}(t) = \dot{x}_3(t) &= \frac{k_p}{J_{mp}}h(t) - \left\{ K_{p_2} \cdot (H_{ref}(t) - \frac{V(t)}{A_T}) + K_i \cdot e_I(t) + R_{dl}(t) \right\} \frac{\Gamma(t)}{I_L} \\
\dot{V}(t) = \dot{x}_4(t) &= \frac{1}{1 + R_d/R_l(t)} \frac{\Gamma(t)}{I_L} - \frac{g\rho}{A_T} R_T(t) V(t) \\
\dot{e}_I(t) = \dot{x}_5(t) &= H_{ref}(t) - \frac{V(t)}{A_T}
\end{aligned}
\tag{5.1}
$$

The measurable outputs are $y(t) = \begin{bmatrix} i\,(t) & \omega(t) & Q_i(t) & Q_d(t) & H(t) \end{bmatrix}^T$, where:

$$\begin{aligned} i(t) = y_1(t) &= g_i \frac{\lambda(t)}{L_a} \\ \omega(t) = y_2(t) &= g_\omega \frac{h(t)}{J_{mp}} \\ Q_i(t) = y_3(t) &= g_{Q_i} \frac{\Gamma(t)}{I_L} \\ Q_d(t) = y_4(t) &= g_{Q_d} \frac{R_l \Gamma(t)}{R_{dl} I_L} \\ H(t) = y_5(t) &= g_H \frac{V(t)}{A_T} \end{aligned} \tag{5.2}$$

Physical meanings of the state and output variables are listed in Table 5.1. System parameters and their nominal values are shown in Table-5.2.

Table 5.1: Motor-pump-pipe state and output variables

State and output variables	Physical Meaning	Unit
λ	Flux linkage in armature	Wb
h	Combined-pump angular momentum	$N-m/s$
Γ	The fluid momentum	$\mathrm{kg}/m-s$
V	The tank's volume	m^3
$\dot{e}_I$	Tank liquid level error	m
i	Current in the armature	A
ω	Pump angular speed	Rad/s
Q_i	Inlet flow rate through the pump	m^3/s
Q_d	The leakage flow rate	m^3/s
H	The liquid level in the tank	m
g_j	The gain of sensor j	

5.5.3 Modeling of Fault Modes

It can be found by examining Figure-5.5 that faults may occur anywhere in the system. A complete model can be built to cover all the possible faults.

Table 5.2: Motor-pump-pipe system parameters

Parameter Symbol	Physical Meaning	Nominal value	Unit
R_a	Armature resistance	[1,2]	Ω
L_a	Armature inductance	0.15	H
k_m	Motor constant	2	$N-m/A$
R_{mp}	Motor-pump combined resistance	1.5	$N-m-s/rad$
J_{mp}	Motor-pump combined inertia	0.025	$kg-m^2$
k_p	Pump constant	[1,1.2]	$N-s/m^2$
L	Length of pipe	2.5	m
D	Pipe diameter	0.25	m
R_d	Discharge resistance	0.3554e-5	kg/m^4-s
A_T	Cross-sectional area of tank	0.0491	m^2
R_T	Tank downstream resistance	4.8240e-6	kg/m^4-s
R_l	Leak resistance	[100,200]	kg/m^4-s
H_{ref}	Nominal height reference	2	m
ω_{ref}	Nominal speed reference	1000	rpm
K_{p1}	Controller-1 proportional gain	5	
K_{p2}	Controller-2 proportional gain	1e5	
K_i	Controller-2 integral gain	1	
ρ	Liquid density	1000	kg/m^3
g_H	Level sensor gain	[0.95,1.05]	

However, it is not necessary and practical to consider all the possible faults. Only the most critical or frequent faults should be modeled. The following faults modes are considered for the Motor-Pump-Hydraulic system.

1. Actuator faults. The faults associated with the motor and the centrifugal pump are termed as actuator faults.

    - DC Motor faults. For the fault associated with the DC motor, they can be factored as the change of the motor armature resistance. These faults include overheating of the motor, winding shortened or broken, brush aging (contact resistance between commuter and brush is higher than normal), etc.
    - Centrifugal pump failure. Many faults may occur in a centrifugal pump, such as seal related problems (leakages, loss of flushing, cooling, quenching systems, etc), pump and motor bearings related problems (loss of lubrication, cooling, contamination of oil, abnormal noise, etc), leakages from pump casing, very high noise and vibration levels, or driver (motor or turbine) related problems. The most common fault associated with centrifugal pump is the inability to deliver the desired flow and head. This can be factored into the changing of the pump constant k_p.

2. Sensor faults. A sensor can be modeled as a gain. For example, the liquid level sensor can be modeled as $H_s = g_H \times H_r$, where H_r is the real liquid level, g_H is the sensor gain and H_s is the sensor reading. Ideally

the value of the gain should be one. Sensor faults include degrading versus time, too much offset or totally lose sensing ability. These faults can be factored by changing of the value of the gain.

3. Hydraulic circuit failure. The fault mode considered here is leakage in the hydraulic circuit. Leakage in the hydraulic circuit can be factored into the changing of the leakage resistance R_l from a large value (no leakage) to a relative small value.

In order to model above faults, the system state equations Eq. 5.1 are modeled as interval ODEs with the following four set value parameters, R_a, k_p, g_H, R_l. Their normal range is given in Table 5.2. The range basically reflects the performance limit of the component that can be treated as normal. For example, g_H in the range between $[0.95, 1.05]$ means the sensor's allowable tolerance is about five percent. If the sensor's tolerance is out of this range, it is treated as a faulty sensor.

5.5.4 Envelope Generation

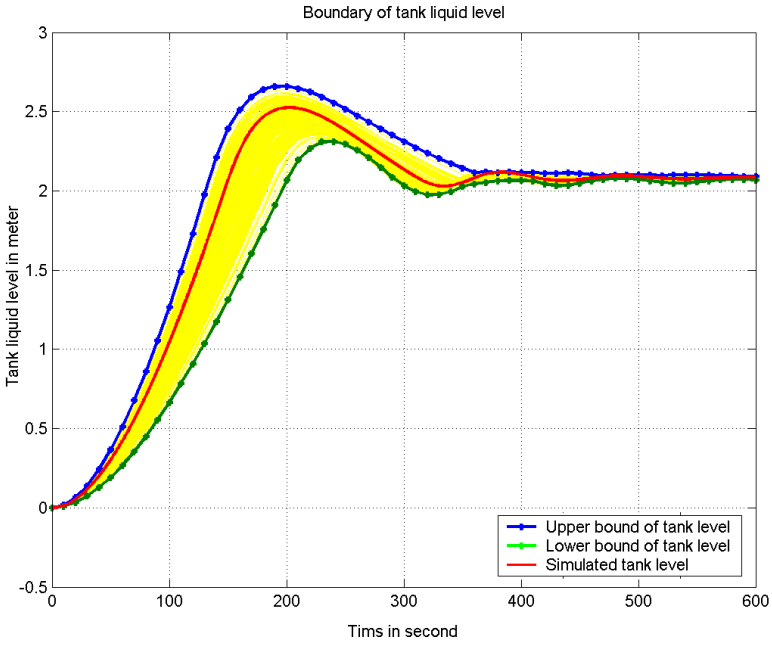
Note that the boundary of each state variable can be used for fault detection purpose. However, in the real physical system only two physical sensors are implemented in the system; i.e., a speed senor monitors the DC motor speed, and a level sensor monitors the tank liquid level. Thus only the boundaries of the speed and tank water level are used for fault detection. By doing this, sensor used for control purpose can also be used for fault detection.

5.5.5 Results

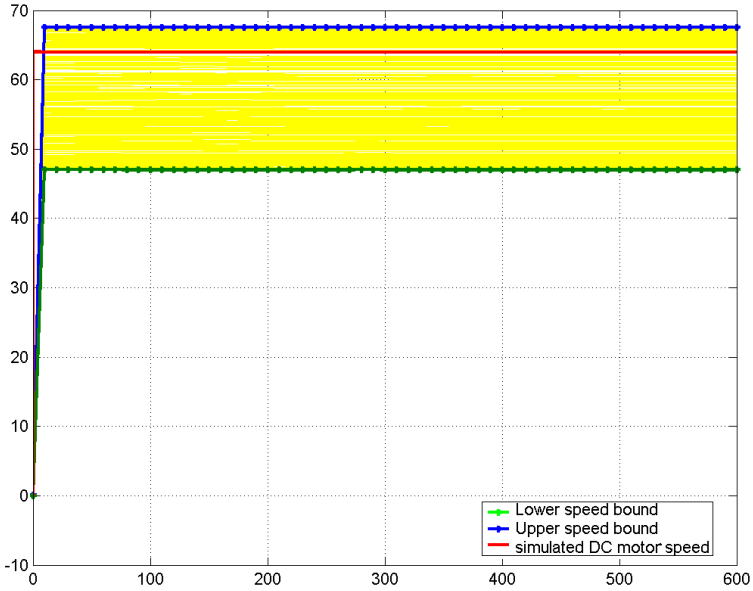
Figure-5.6 shows the envelopes generated using gradient enhanced moving least square method and the result of Monte Carlo method, along with the nominal system performance curves. The boundaries of two measurable signals: the tank liquid level and DC motor speed are shown. It can be found that the GEMLS approach can predict the envelope very accurately.

The DC motor speed curve is almost a constant because of the P controller. The following faults scenarios are simulated and the alarms are generated when the simulated curves intersect with the boundary. Also, we assume at a certain time that only one fault has occurred, and the other parameters remain at their nominal value. This assumption is necessary if fault isolation is needed.

1. DC motor fault. The armature resistance is increasing with time as $R_a = 1.5 + 0.01 * t$ but saturates at $R_a = 4\Omega$. This scenario may represent overheating of the rotor of the DC motor. The simulated results are shown in Figure-5.7. It can be found from speed response that the final DC motor speed is determined by the value of the armature resistance, which limits the DC motor output torque. For this fault scenario, the fault occurred at $t = 150s$, at which the armature resistance is larger than 3Ω. The speed curve will generate an alarm almost immediately when R_a became 3Ω, but the alarm given by the tank level signal will have roughly 80 seconds delay.

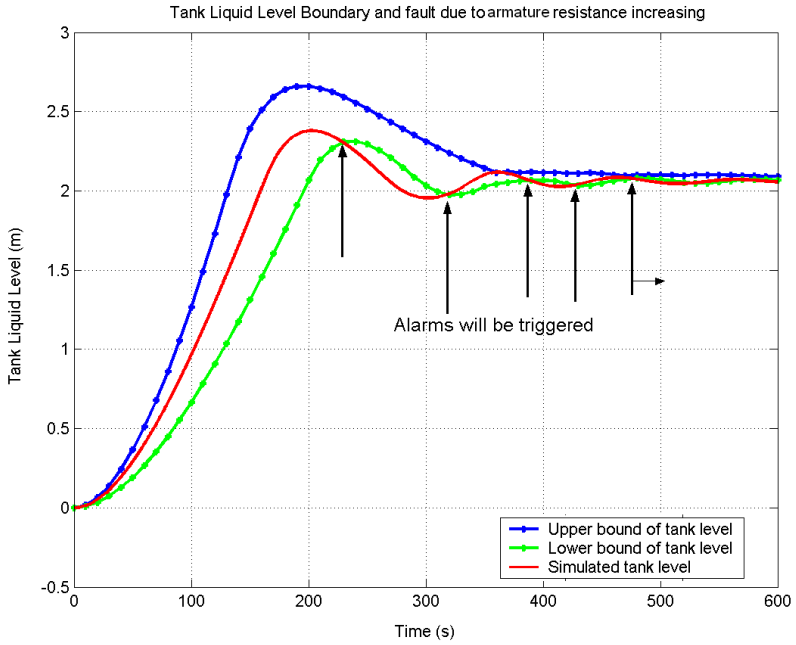


(a) Envelope of tank liquid level

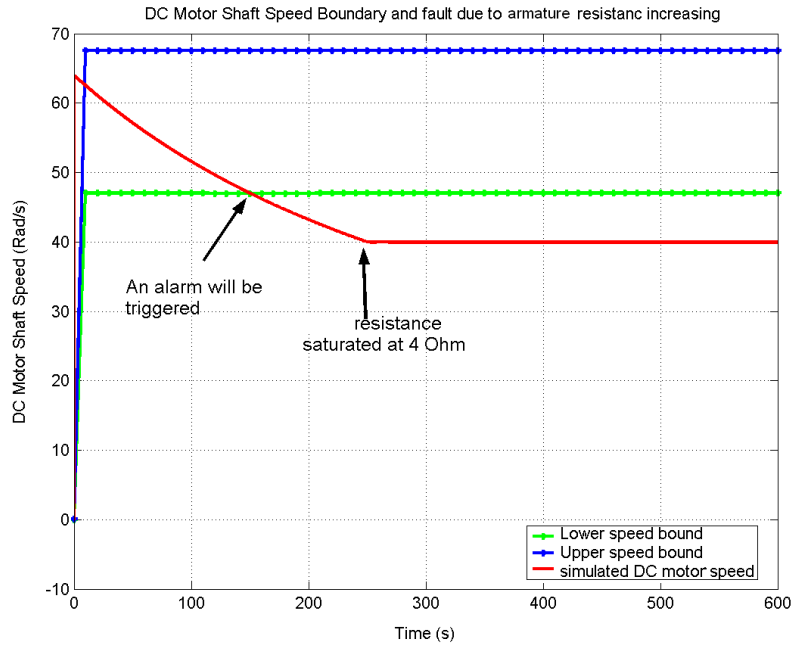


(b) Envelope of DC motor speed

Figure 5.6: Envelope of system performance



(a) Tank level and fault due to armature resistance



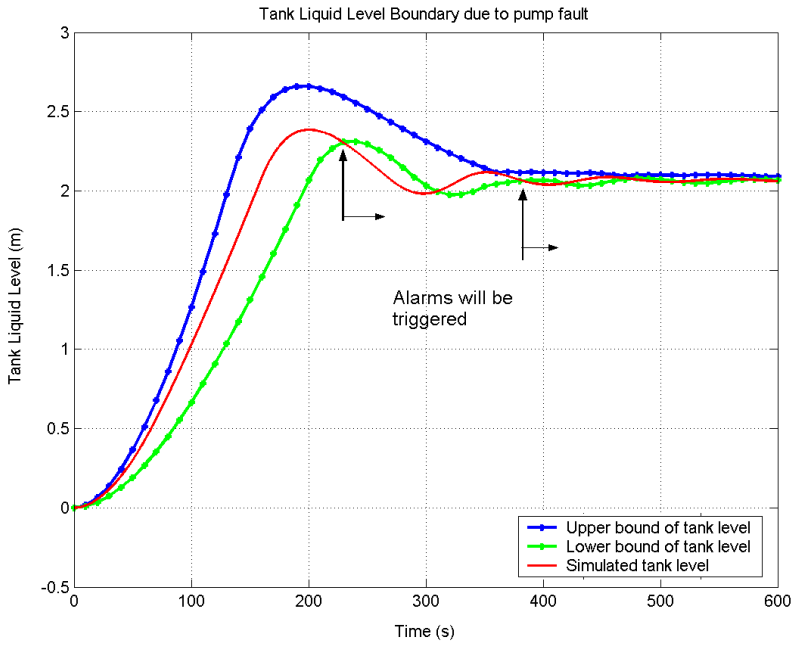
(b) DC Motor speed and fault due to armature resistance

Figure 5.7: Fault due to DC motor armature resistance change

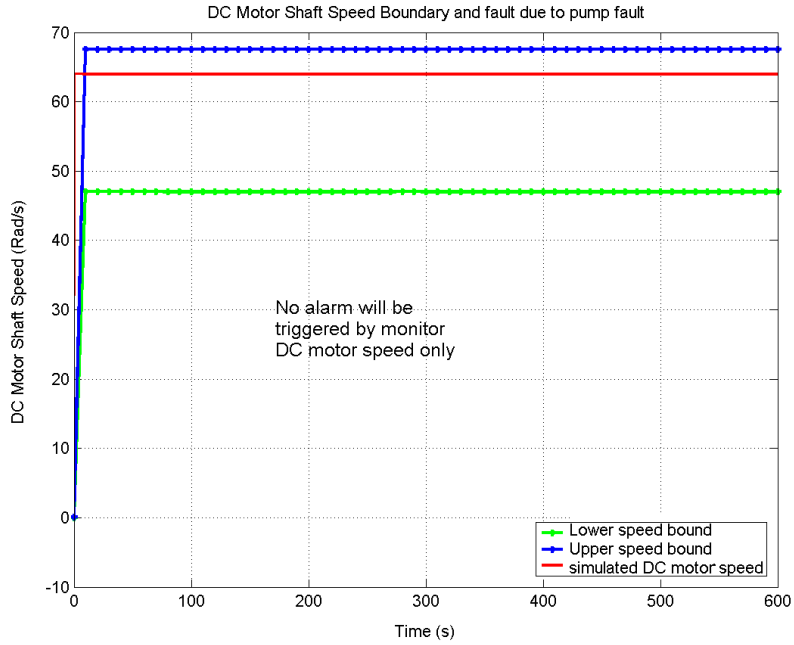
2. Centrifugal pump fault. The pump constant has a sudden change from $k_p = 1.1$ to $k_p = 0.8$ at $t = 100 seconds$. The sudden change of pump constant may be due to many reasons described in the previous section. The simulated results are shown in Figure-5.8. It is found that this fault will not be detected from the DC motor speed curve. This is because the P controller used to maintain the speed of the DC motor will compensate for this fault so that motor speed will remain in the normal range. However, the tank level signal will generate alarms. Sensitivity analysis in the next section will also show that the speed signal is not capable of helping detect the pump constant change fault. The fault occurred at $t = 100s$ and the delay is about 130 seconds.

3. Level sensor fault. The gain of the level sensor degrades with time as $g_H = 1 - 0.01 * t$, and saturates at $g_H = 0.85$. This degradation model is used to simulate a sensor degradation as might result from many reasons. The simulated results are shown in Figure-5.9. As for the pump fault, the motor speed curve is not capable of reporting this fault. The fault occurred at $t = 5s$ when $g_H = 0.95$, but the system reported the alarm at $t = 390s$.

4. Leakage in pipe. The leakage resistance, R_l suddenly jumps from $1e2$ to $1e-5$ at $t = 200s$, which corresponds to a sudden break in the pipe. The simulated results are shown in Figure-5.10. The system reported this fault at $t = 220s$. Again, the DC motor speed curve can not report this fault.

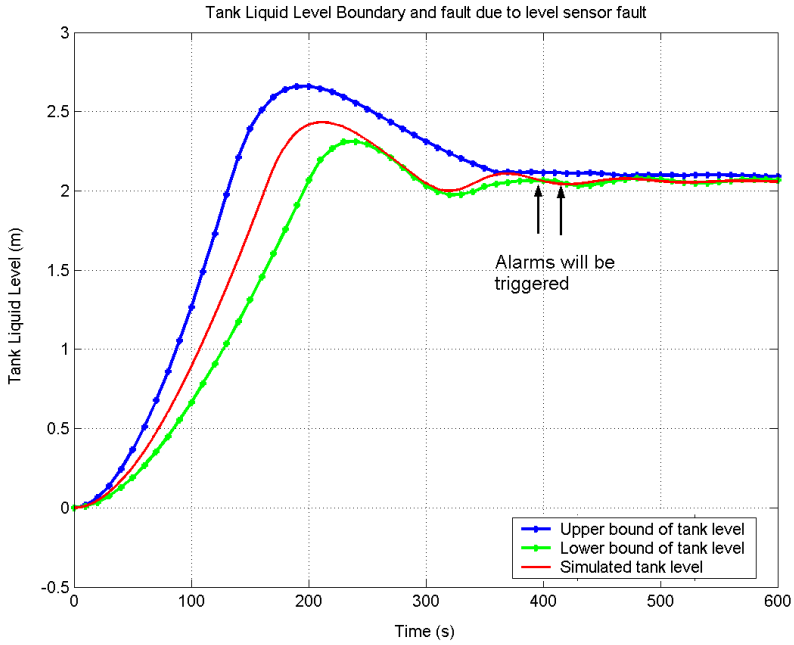


(a) Tank level and fault due to pump constant change

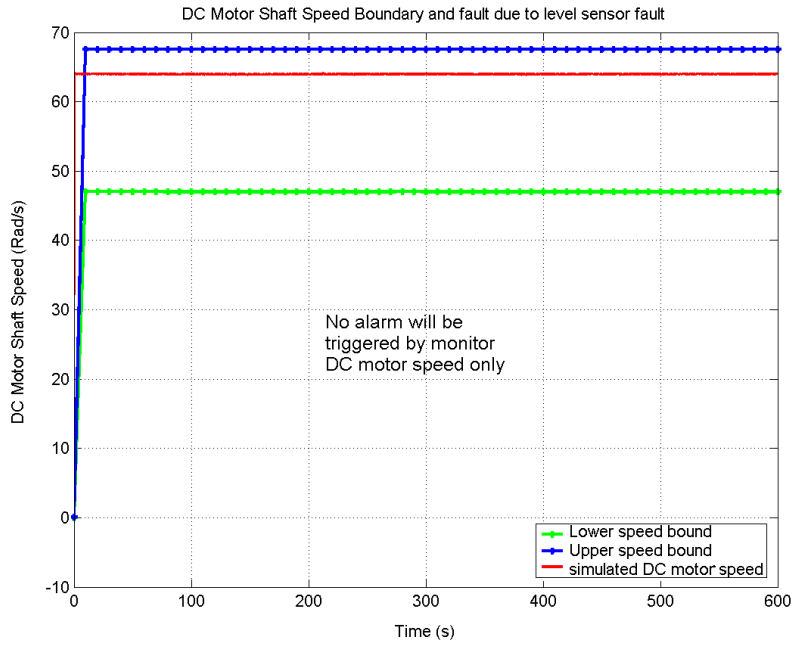


(b) DC Motor speed and fault due to pump constant change

Figure 5.8: Fault due to pump constant change

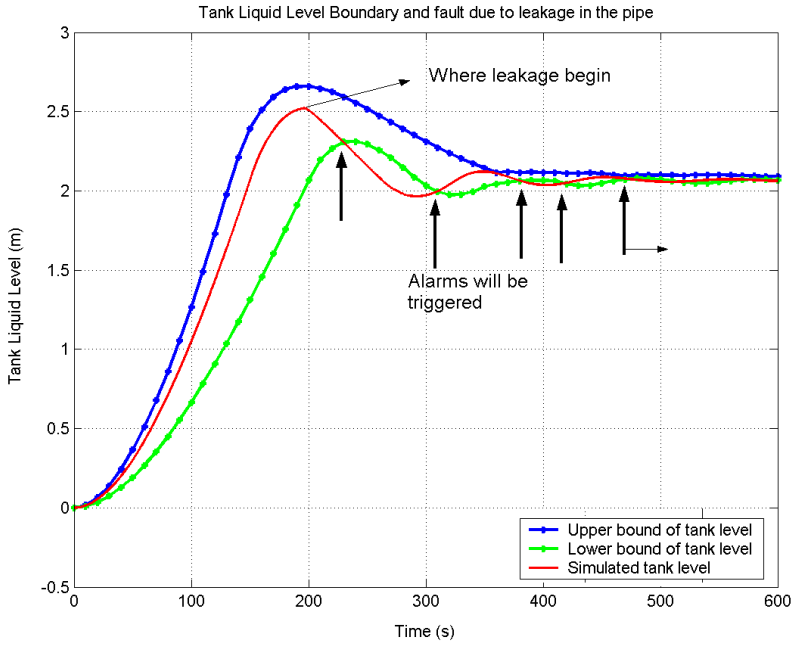


(a) Tank level and fault due to level sensor degradation

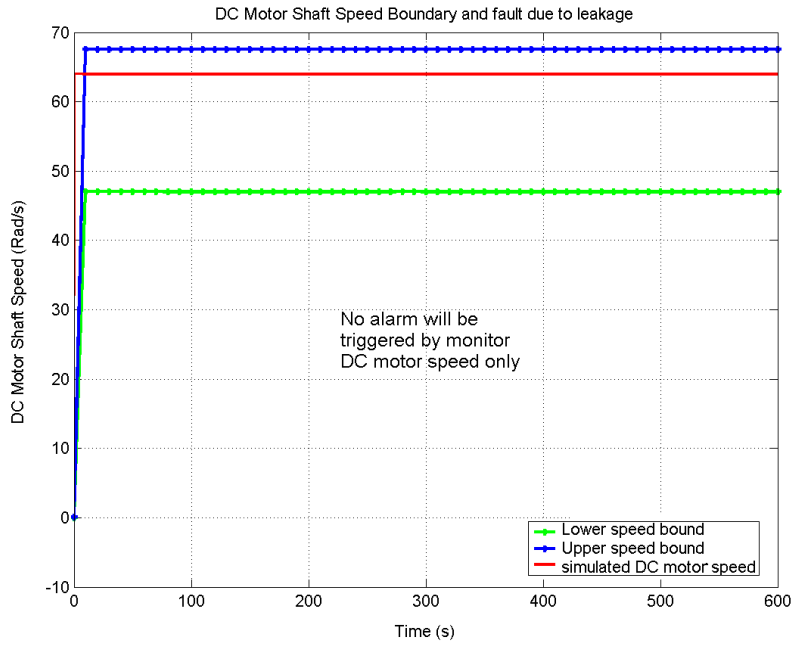


(b) DC Motor speed and fault due to level sensor degradation

Figure 5.9: Fault due to level sensor degradation



(a) Tank level and fault due to pipe leakage



(b) DC Motor speed and fault due to pipe leakage

Figure 5.10: Fault due to pipe leakage

Notice that for discussion purpose, how the fault might occur are described in this section. However, this is not necessary if this approach is applied for fault detection. The only condition for this approach is that what fault may occur should be known in advance so that the parameter associated with that fault will be treated as an interval rather than a fixed value.

5.5.6 Discussion

The following observations are made from the above simulated results.

1. The system seems to respond to the faults too slowly.

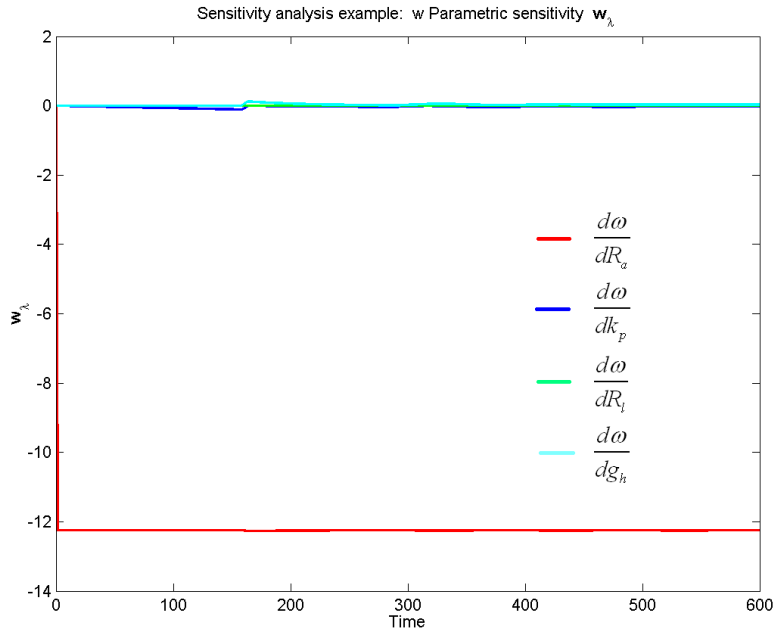
2. Only the tank level signal is able to report all the faults.

For the first observation, it is because the uncertain model is built to cover all four fault scenarios and thus the boundary is generated with four parameters varying from their min to max values simultaneously. This approach can actually cover the worst case when the four faults occur together. In this example, the fault scenarios were simulated so that only one parameter changed and the other three parameters remained fixed. Thus, the envelopes are quite conservative for a single fault. One way to improve the response time is to generate four set of envelopes, each set for one fault scenario. For example, a set of envelopes just to detect sensor fault can be generated with the armature resistance, pump constant and leakage resistance just varying in a very narrow range. However, this may increase the false alarm ratio. So the

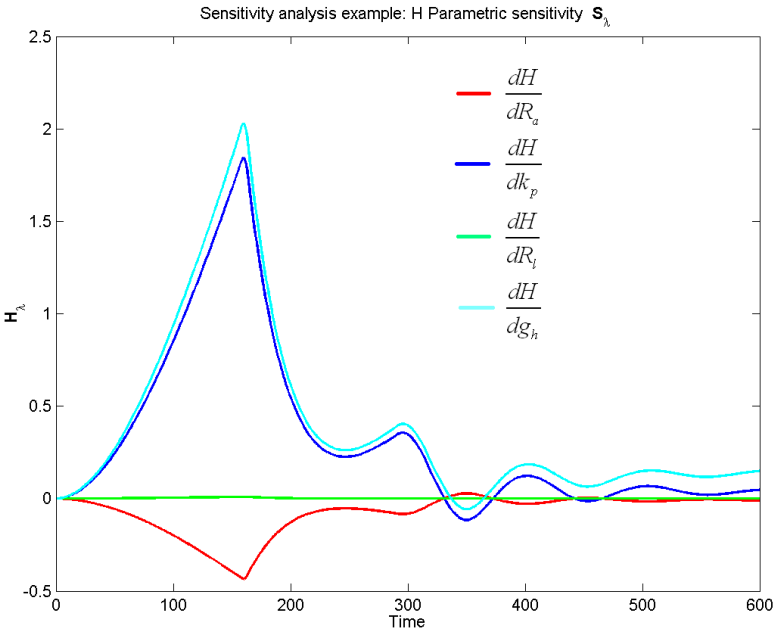
response time of the system will be a trade off among several factors such as model complexity, miss alarm ratio and false alarm ratio.

For the second observation, sensitivity analysis can also show that speed signal is able to report only the first fault of armature resistor change. Figure-5.11(a) shows the sensitivity of speed against the four parameters, armature resistor, R_a, pump constant, k_p, leakage resistance, R_l and level sensor gain, g_H. Clearly, it shows that it is only sensitive to R_a and the sensitivity coefficients of the other parameters are almost zero. Thus it is not able to report the other three faults.

Also, the envelopes of the system are generated by searching in 4 dimensions since four fault scenarios are to be detected. However, the sensitivity analysis results also indicate that the search space can be reduced. Figure-5.11(a) shows that in order to generate the DC motor speed envelope, only R_a need to be considered since there other three parameters will not influence the speed. The sensitivity coefficient to R_a is always less than zero and thus only the boundary value of R_a need to be sampled, as discussed in Chapter 3. For the tank level signal, the situation is more complicated but we can find from Figure-5.11(b) that before $t = 300s$, all the sensitivity coefficients (at nominal value) show strong monotone character. Thus, sensitivity band analysis procedures described in Chapter 3 can also be used to reduce computational cost.



(a) DC Motor speed sensitivity analysis at nominal values



(b) Tank level sensitivity analysis at nominal values

Figure 5.11: Sensitivity analysis to reduce search space

5.6 Summary

This chapter treats fault detection as a simulation of a parametric uncertain system. A fault free system is modeled as a parametric uncertain system whose parameters are intervals. The system boundaries are used as threshold for fault detection. A fault is defined when system parameters that are not in the normal set. A fault is reported whenever the monitored system performance interacts with the system boundary. The advantages of this method, compared to other model-based fault detection are: 1) The uncertain model can naturally reflect the uncertainties associate with the values of the components in the system, and 2) Using boundary of the uncertain system as an adaptive threshold can give better miss alarm and false alarm ratio than fixed threshold.

Chapter 6

Summary, Future Work and Conclusions

This chapter summarizes this dissertation and outlines directions for future work.

6.1 Summary

This dissertation describes a study into numerical methods for representation and simulation of dynamic systems with time invariant uncertain parameters. Simulation is defined in the context of parametric uncertain system as the process of computing the boundary of a system response that contains all the possible behavior of an uncertain system. This research is motivated by the facts that modeling and simulation of physical systems is often complicated by the presence of uncertainties.

Existing and past work on uncertainty categorization, representation and propagation are reviewed. The metrics used to compare different uncertain propagation approaches are presented. The review revealed that simulation of systems with uncertain parameters remains a challenging problem, particularly because of the computational cost. Also, boundary of the output of an uncertain system has a wide range of applications. The review also motivated

developing of a numerical method to simulate parametric uncertain nonlinear dynamic systems with the ability to deal with large time invariant uncertainties. The method should improve the accuracy and reduce the computational cost for these types of problems.

Based on the review and the computational cost analysis of the problem, a method that is a trade-off between efficiency and accuracy is developed. The basic idea is to study an approximation or a surrogate of the original system model, instead of the original system. To construct the surrogate model, the response surface method (RSM) is employed. Since the optimization problem to be solved is generally non-convex, there may be multiple local optima. Conventional RSM using polynomials which provides global approximation is not able to deal with the non-convex problem. Thus a local approximation approach called Moving Least Square (MLS) is used for response surface construction. For more complicated systems, a gradient enhanced moving least square (GEMLS) response surface method is used to solve the global optimization problem more efficiently. This method takes advantage of the fact that parametric sensitivity of an ODE system can be calculated as a by-product when solving the original system with less computational cost. With the help of sensitivity information, the number of samples needed to construct the response surfaces is further decreased, and the quality of the response surface can be improved. Furthermore, global sensitivity analysis for monotonic testing to further reduce the number of samples is introduced.

Once the approach is developed, it has been applied to several engi-

neering applications. The first application is hybrid system verification by reachable set computing. The reachable set computing/approximation problem is formulated as a simulation of a dynamic system with uncertain initial conditions. Methods developed for parametric uncertain system simulation can be directly applied to this problem. For reachable set computing, a more accurate boundary should be found. Thus more complicated geometries, such as polyhedra should be used rather than hyper rectangle to represent the reachable set. The computational burden of current methods for reachable set approximation, such as polyhedral approximation, is studied. It shows that these methods involve global optimization techniques that embed numerical simulations of the dynamic system response into the routine for evaluating the objective function. The search space is the entire uncertain initial state in $\mathbb{R}^n$. It is general but computationally expensive, and thus not applicable if simulation of the system is already computationally burdensome. The applicability of some existing approaches that avoid the global optimization problem by employing the fundamental inequality theorem are shown to be very limited.

To reduce the computational burden, we first proved a boundary theorem that reveals that the boundary of the reachable set is formed only by the trajectories from the boundary of the initial state region. This result reduces the search space from $\mathbb{R}^n$ to $\mathbb{R}^{n-1}$. For more complicated systems, the GEMLS method proposed is used to solve the global optimization problem more efficiently. Finally, it is shown that it will be more efficient and accurate if GEMLS is integrated with Principal Component Analysis (PCA) to find an

oriented rectangular hull for reachable set representation and approximation.

The other engineering application is model based fault detection. Model-based fault detection can also be treated as simulation of parametric uncertain systems. First, a fault free system is modeled as a parametric uncertain system whose parameters belong to a given bounded parameter set, which is called normal set. The performance of a fault free system is bounded by the boundary that can be acquired by using the approach described in previous chapters. A fault is defined when system parameters do not belong to the normal set due to malfunction or degradation. Once such fault(s) occur, the monitored system performance will extend beyond the system boundary predicted by the parametric uncertain model. A fault is reported whenever the monitored system performance interacts with the system boundary. Compared to conventional model-based fault detection methods that use a fixed threshold, the method used in this research uses the boundary as an adaptive threshold that can give better miss alarm and false alarm ratio.

This dissertation can be summarized by Figure-6.1.

6.2 Future Work

This dissertation has developed a hybrid numerical method that integrates RSM and local/global sensitivity analysis for simulation of parametric uncertain dynamic systems. Using this method as a starting point, a number of future research directions can be found.

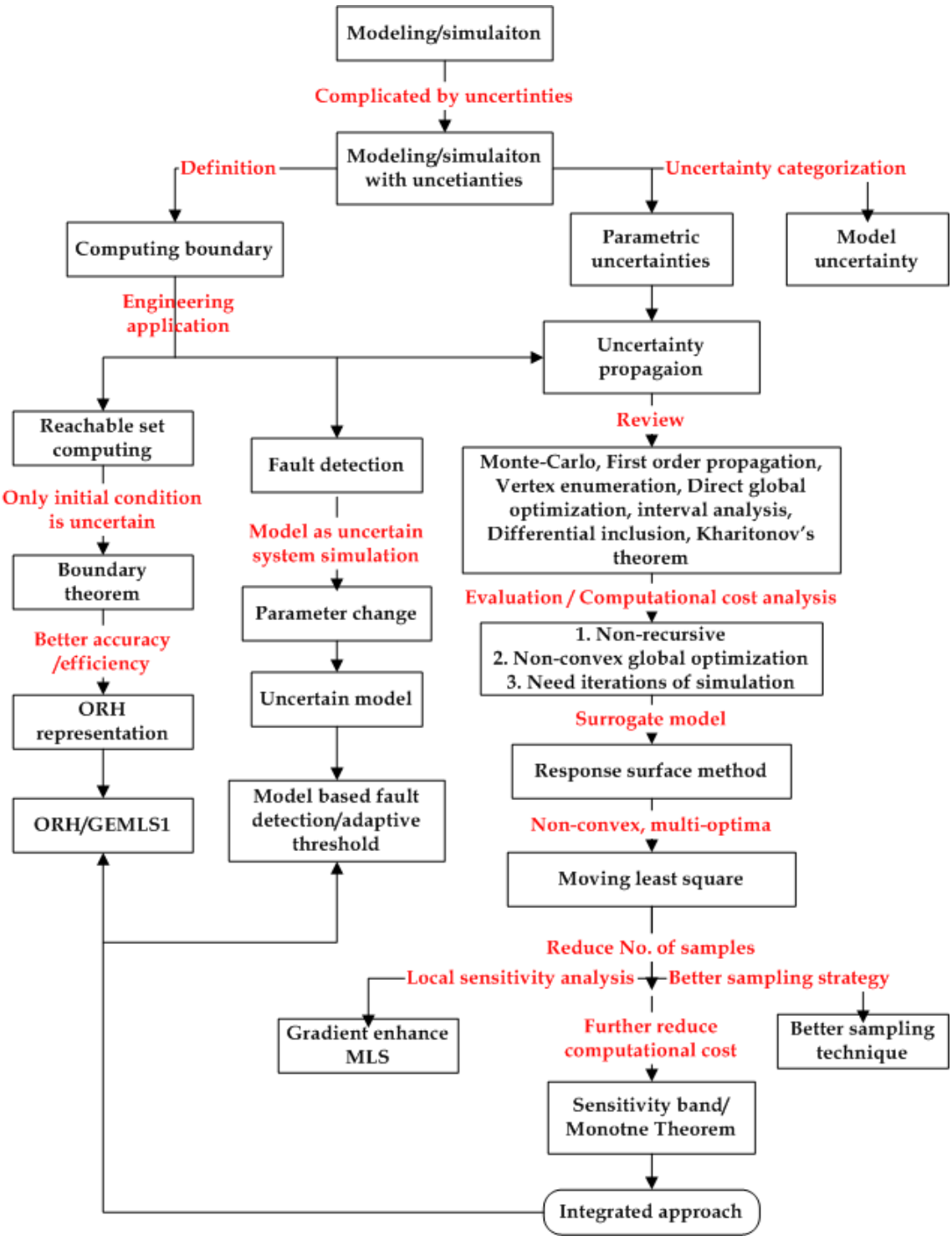


Figure 6.1: Summary of the dissertation

1. More engineering applications should be studied. One possible engineering application that this research can be applied to is to support worst case design. Worst-case design ensures that hardware meets or exceeds the design specifications over all possible component variations. Conventionally, it is done by using vertex enumeration. However, as we pointed in the previous discussion and by other researchers, the results given by the 'corner value' simulation has been found to be overly pessimistic. Clearly, worst case simulation can be transformed to simulation of parametric uncertain systems and the proposed method can be applied.

2. Methods that could generate more accurate optimization results should be studied. One disadvantage of RSM is that it is hard to validate the results. It can not guarantee to give an very accurate global optimization result unless a large number of samples are taken. For the reachable set problem, RSM will not guarantee an overestimation of the real reachable set unless the optima given by RSM is accurate or larger than the real optima. For fault detection problem, a non-accurate envelope will increase the false or miss alarm ratio. One way to improve the accuracy is by a two level algorithm: 1) Construct a RSM and find a level-1 global optima point, due to its approximation error, level optima is not quite match with the real optima. 2) use the level 1 optima as a starting search point and construct a new response surface in the small region around this starting point and find a level-2 optima which will be much more accurate. Above step can be repeated several times until the result con-

verges; or at level-2 gradient-based method around level-1 optima can be applied since point since the gradient information is available [83].

3. Other multi-variable regression methods should be explored. Although MLS is a very powerful regression method, other multi-variable regression methods, such as Kriging [54], Multivariate Adaptive Regression Splines (MARS) [30], Neural Networks [55] should also be considered, especially for systems with a large number of uncertain parameters.

4. Although a theorem to test the monotonicity of ODE systems is given, it has very limited applicability. New methods that can quickly determine the monotonicity of ODE systems should be developed.

6.3 Conclusions

Limited by the available computing power, uncertainty propagation for systems with a large number of uncertain parameters will remain a challenging problem. For systems withe less complexity, this study shows that a hybrid approach composed of RSM, local/global sensitivity analysis provides an effective solution.

Bibliography

[1] http://csep1.phy.ornl.gov.

[2] P.K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. ACM Computing Surveys, 30(4):412–458, December 1998.

[3] P. Antsaklis and X. Kout-soukos. Enabled Control: Information Technology for Dynamical Systems, chapter Hybrid Systems: Review and Recent Progress, pages 273–298. IEEE Press, 2003.

[4] J. Armengol. Application of Modal Interval Analysis to the simulation of the behaviour of dynamic systems with uncertain parameters. PhD thesis, Universitat de Girona,Catalonia, Spain, 2000.

[5] K. O. Arras. An introduction to error propagation: Derivation, meaning and examples of equation $C_Y = F_X C_X F_X^T$. Technical Report EPFL-ASL-TR-98-01 R3, Autonomous Systems Lab, Institute of Robotic Systems, Swiss Federal Institute of Technology Lausanne (EPFL), Sep 1998.

[6] E. Asarin, T. Dang, and A. Girard. Reachability analysis of nonlinear systems using conservative approximation. In A. Pnueli O. Maler, editor, Hybrid Systems: Computation and Control: 6th International Workshop, HSCC 2003, volume 2623 of Lecture Notes in Computer Science, pages 20–35. Springer-Verlag Heidelberg, April 2003.

[7] R.R. Barton. Meta modelling: A state of the art review. In J.D. Tew, S. Manivannan, D.A. Sadowski, and A.F. Seila, editors, Proceedings of the 1994 Winter Simulation Conference, pages 237–244, 1994.

[8] R.R. Barton. Simulation metamodels: A state of the art review. In D.J. Medeiros, E.F. Watson, J.S. Carson, and M.S. Manivannan, editors, Proceedings of the 1998 Winter Simulation Conference, pages 167–172, 1998.

[9] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, W. Yi, and C. Weise. New generation of uppaal. In Proceedings of the International Workshop on Software Tools for Technology Transfer, Aalborg, Denmark, July 1998.

[10] S.P. Bhattacharyya, H. Chapellat, and L.H. Keel. Robust Control: The Parametric Approach. Prentice Hull, 1995.

[11] G.E.P. Box and n.R. Draper. Empirical model-building and response surfaces. New York, John Wiley & Sons.

[12] M.S. Branicky. Studies in Hybrid Systems: Modelling, Analysis, and Control. PhD thesis, MIT, June 1995.

[13] H. Brüggemann and U. Kiencke. Uncertainty theory in vehicle dynamics simulation. In Proceedings of the American Control Conference, volume 1, pages 298–303, May 2002.

[14] D. Chaniotis, M.A. Pai, and I.A. Hiskens. Sensitivity analysis of differential-algebraic systems using the gmres method - application to power systems.

In Proceedings of the IEEE International Symposium on Circuits and Systems, volume 3, pages 117–120, Sydney, Australia, May 2001.

[15] J. Chen and R.J. Patton. Robust model-based fault diagnosis for dynamic systems. Kluwer Academic Publishers, New York, Boston, 1999.

[16] H.S Chung and J.J. Alonso. Design of a low-boom supersonic business jet using cokriging approximation models. In 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, Georgia, September 2002.

[17] H.S. Chung and J.J. Alonso. Using gradients to construct cokriging approximation models for high-dimensional design optimization problems. In 40th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, January 2002.

[18] A. Chutinan. Hybrid System Verification Using Discrete Model Approximations. PhD thesis, Carnegie Mellon University, May 1999.

[19] A. Chutinan and B.H. Krogh. Computing polyhedral approximations to flow pipes for dynamic systems. In Decision and Control, 1998. Proceedings of the 37th IEEE Conference on, volume 2, pages 2089–2094, December 1998.

[20] A. Chutinan and B.H. Krogh. Computational techniques for hybrid system verification. IEEE Trans. on Automatic Control, 48(1):64–75, 2003.

[21] P. Cugueró, V. Puig, J. Saludes, and T. Escobet. Avoiding possible instability in robust simulation of stable parametric uncertain time-invariant systems. In Proceedings of the 40th IEEE Conference on Decision and Control, volume 2, pages 1993–1994, 2001.

[22] B. Cullimore. Automated determination of worst-case design scenarios. In ICES 2003, 2003.

[23] T. Dang and O. Maler. Reachability analysis via face lifting. In HSCC, pages 96–109, 1998.

[24] A.M. Dunker. The decoupled direct method for calculating sensitivity coefficients in chemical kinetics. J. of Chem. Phys, 81(5):2385–2393, September 1984.

[25] L. Egiziano, N. Femia, and G. Spagnuolo. New approaches to the true worst-case evaluation in circuit tolerance analysis-part i: Calculation of the inner solution by genetic algorithms. In Proc. of 7th IEEE Workshop on Computers in Power Electronics, pages 133–139, Como, Italy, Jul 1998. IEEE.

[26] L. Egiziano, N. Femia, and G. Spagnuolo. New approaches to the true worst-case evluation in circuit tolerance analysis-part i: Calculation of the outer solution by affine arithmetic. In Proc. of 7th IEEE Workshop on Computers in Power Electronics, pages 141–147, Como, Italy, Jul 1998. IEEE.

[27] G.E. Fasshauer. Approximate moving least-squares approximation for time-dependent PDEs. In H.A. Mang, F.G. Rammerstorfer, and J. Eberhardsteiner, editors, Fifth World Congress on Computational Mechanics, Vienna, Austria, July 2002.

[28] N. Femia and G. Spagnuolo. True worst-case circuit tolerance analysis using genetic algorithms and affine arithmetic. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 47(9):1285–1297, September 2000.

[29] T.F. Filippova, A.B. Kurzhanski, K. Sugimoto, and I. Vàlyi. Ellipsoidal state estimation for uncertain dynamical systems. In Mario Milanese, John Norton, Hélène Piet-Lahanier, and Éric Walter, editors, Bounding Approachs to System Identification, pages 213–238. Plenum Press, New York, 1996.

[30] J.H. Friedman. Multivariate adaptive regression splines (with discussion). Annals of Statistics, 19(1), 1991.

[31] G.J. Gaston and A.J. Walton. The integration of simulation and response surface methodology for the optimization of ic processes. IEEE Transactions on Semiconductor Manufacturing, 7(1):22–33, February 1994.

[32] A. Giunta, L. Watson, and J. Koehler. A comparison of approximation modeling techniques: Polynomial versus interpolating models, 1998.

[33] H. Hashemolhosseini, H. Dalayeli, and M. Farzin. Correction of hydrostatic pressure obtained by the finite element flow formulation using moving least squares method. Journal of Materials Processing Technology, 125-126:588–593, 2002.

[34] J. Heeks, E.P. Hofer, B. Tibken, and K. Thorwart. An interval arithmetic approach for discrete time nonsmooth uncertain systems wtih application to an antilocking system. In Proceedings of the American Control Conference, volume 2, pages 1849–1854, May 2002.

[35] T.A. Henzinger, P.H. Ho, and H. Wong-Toi. HYTECH: A model checker for hybrid systems. International Journal on Software Tools for Technology Transfer, 1(1–2):110–122, 1997.

[36] I.A. Hiskens and M.A. Pai. Trajectory sensitivity analysis of hybrid systems. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 47(2):204–220, February 2000.

[37] S.L. Ho, S. Yang, P. Ni, and H.C. Wong. Developments of an efficient global optimal design technique - a combined approach of mls and sa algorithm. Int J for Computation and Maths. in Electrical and Electronic Eng., 21(4):604–614, 2002.

[38] M. Hofbaur and N. Dourdoumas. Lyapunov based reasoning methods. IEEE Transactions on Systems, Man and Cybernetics-PART A: Systems and Humans, 31(6):546–558, Nov 2001.

[39] J.H. Hubbard. Differental Equations, A dynamical System Appaoch, Part I: Ordinary Differential Equations, chapter 4, pages 169–172. Number 5 in Text in Applied Mathematics. Springer-Verlag, New York, 1991.

[40] Z. Hurak and M. Šebek. New tools for spherical uncertain systems with polynomial toolbox for matlab. In Proc. of Matlab'00 conference, pages 116–121, 2000.

[41] E.H. Isaaks and R.M. Srivastava. An Introduction to Applied Geostatistics. Oxford University Press, New York, USA, 1989.

[42] S.S. Isukapalli and P.G. Georgopoulos. Computational methods for sensitivity and uncertainty analysis for environmental and biological models. Technical Report EPA/600/R-01-068, Computational Chemodynamcis Laboratory, Environmental and Occupational Health Science Institute, Rutgers University and UMDN RWJ Medical School, 170 Frelinghuysen Road, Piscataway, NJ 08854, December 2001.

[43] L. Jaulin, M. Kieffer, O. Didrit, E. Walter, and O. Didrit, editors. Applied Interval Analysis. Springer, London; New York, 2001.

[44] I.T. Jolliffe. Principal Component Analysis. Spring-Verlag, New York, 1986.

[45] H. Kay. Refining Imprecise Models and Their Behaviors. PhD thesis, The University of Texas at Austin, 1996.

[46] R. B. Kearfott. Interval computations: Introduction, uses, and resources.

[47] C. Kim, S. Wang, and K.K. Choi. Efficient response surface modelling using MLSM and sensitivity. In WCSMO1, Dalian China, June 2001.

[48] T. Krishnamurthy. Response surface approximation with augmented and compactly supported radial basis functions. In 44th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, number AIAA-2003-1748, Norfolk, Virginia, April 2003.

[49] B. Kuipers. Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge. MIT Press, 1994.

[50] B. Kuipers. Encyclopedia of Physical Science and Technology, chapter Qualitative Simulation, pages 287–300. Academic Press, New York, 3rd edition, 2001.

[51] S. Lauridseny, R. Vitaliz, F. van Keulenx, R. T. Haftkaz, and J. I. Madsen. Response surface approximation using gradient information. In World Congress of Structural and Multidisciplinary Optimization:WCSMO01, Dalian China, June 2001.

[52] D. Levin. The approximation power of moving least-squares. MATHEMATICS OF COMPUTATION, 67(224):1517–1531, October 1998.

[53] D. Liberzon. Switching in Systems and Control. System and Control:Foundations and Applications. Birkhauser, 2003.

[54] W. Liu and S.M. Batill. Gradient-enhanced response surface approximations using kriging models. In Proceedings of 9th AIAA/ISSMO

Symposium and Exhibit on Multidisciplinary Analysis and Optimization, Atlanta, Georgia, USA, September 2002.

[55] W. Liu, S.M. Batill, and J.E. Renaud. Implementation issues in gradient-enhanced neural network response surface approximations. In Proceedings of the 4th World Congress of Structural and Multidisciplinary Optimization, Dalian, China, June 2001.

[56] Z. Malik, D. Dyck, J. Nelder, R. Spence, and D. Lowther. Response surface models using function values and gradient information, with application to the design of an electromagnetic device. In Proc. of Design Reuse EDC98, Brunel, 1998.

[57] M.D. McKay. Latin hypercube sampling as a tool in uncertainty analysis of computer models. In J.J. Swain, D. Goldsman, R.C. Crain, and J.R. Wilson, editors, Proceedings of the 1992 Winter Simulation Conference, pages 557–564, 1992.

[58] I. Mitchell. Application of Level Set Methods to control and Reachability Problems in Continuous and Hyrbid Systems. PhD thesis, Stanford University, August 2002. Page 85.

[59] R.E. Moore. Interval Analysis. Prentice Hall, New Jersey, 1968.

[60] R.H. Myers and D.C. Montgomery. Response Surface Methodology. John Wiley and Sons Ltd., 2 edition, 2002.

[61] T.B. Nguyen, M.A. Pai, and I.A. Hiskens. Sensitivity approaches for direct computation of critical parameters in a power system. International Journal of Electrical Power and Energy Systems, 24(337-343), July 2002.

[62] W.L. Oberkampf, J.C. Helton, C.A. Joslyn, S.F. Wojtkiewicz, and S. Ferson. Challenge problems: Uncertainty in system response given uncertain parameters. http://www.sandia.gov/epistemic/prob.statement.12-01.pdf, November 2001.

[63] P.Y. Papalambros and D.J. Wilde. Principles of Optimal Design. Cambridge University Press, New York, 1988.

[64] V. Puig, J. Saludes, and J. Quevedo. Worst-case simulation of discrete linear time-invariant interval dynamic systems. Reliable Computing, (9):251–290, 2003.

[65] S. Raczynski. Differential inclusios in system simulation. Transaction of The Society for Computer Simulation, 13(1):47–55, 1996.

[66] S. Raczynski. Differential inclusion solver. In International Conference on Grand Challenges for Modeling and Simulation, San Antonio, TX, USA, Jan 2002. The Society for Computer Simulation.

[67] A. Saltelli, K. Chan, and E. M. Scott. Sensitivity Analysis. Wiley Series in Probability and Statistics. John Wiley and Sons Ltd., 2000.

[68] S. Saludes, V. Puig, and J. Quevedo. Determination of window length for a new algorithm in adaptive threshold generation. In TEMPUS

Workshop on Systems Modelling, Fault Diagnosis and Fuzzy Logic Control, pages FD1–13–FD1–18, Hungrial Fecha, June 1997.

[69] H. Scholten. Good modelling practice. In 13th JISR-IIASA Workshop on methodologies and tools for complex system modeling and integrated policy assessment, pages 57–59, Laxenburg, 1999.

[70] D. Shepard. A two-dimensional interpolation function for irregularly spaced points. In Proc. 23rd National Conference ACM, pages 517–524, 1968.

[71] C.J.R. Shi and M.W. Tian. Simulation and sensitivity of linear analog circuits under parameter variations by robust interval analysis. ACM Transactions on Design Automation of Electronic Systems, 4(3):280–312, Jul 1999.

[72] B. I. Silva, O. Stursberg, B. H. Krogh, and S. Engell. An assessment of the current status of algorithmic approaches to the verification of hybrid systems. In Proceedings of the IEEE Conference on Decision and Control, pages 2867–2874, Orlando, FL, 2001.

[73] B.I. Silva and B.H. Krogh. Formal verification of hybrid systems using checkmate: A case study. In Proceedings of the American Control Conference, pages 1679–1683, Chicago, IL, 2000.

[74] G. V. Smirnov. Introduction to the theory of differential inclusions, volume 41 of Graduate studies in mathematics 1065-7339. American

Mathematical Society, 2002.

[75] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, Berkeley, CA, 1995.

[76] O. Stursberg and B.H. Krogh. Efficient representation and computation of reachable sets for hybrid systems. In A. Pnueli O. Maler, editor, Hybrid Systems: Computation and Control: 6th International Workshop, HSCC 2003, volume 2623/2003 of Lecture Notes in Computer Science, pages 482–497. Springer-Verlag Heidelberg, April 2003.

[77] M.W. Tian and R.C.J Shi. Worst case tolerance analysis of linear analog circuits using sensitivity bands. IEEE Transactions on Circuits and Systems, I: Fundamental Theory and Applications, 47(8):1138–1145, August 2000.

[78] B. Tibken and E.P. Hofer. Simulation of controlled uncertain nonlinear systems. Applied Mathematics and Computation, (70):329–338, 1995.

[79] C. Tubb and E. Omerdic. Fault detection and handling. IMPROVES technical report FD 001(n), Mechatronics Research Center, University of Wales College, Newport, PO BOX 180, Newport, NP20 5XR, Oct 2001.

[80] M. Vescovi, A. Farquhar, and Y. Iwasaki. Numerical interval simulation. In International Joint Conference on Arttficial Intelligence IJCAI 95, pages 1806–1813, Montreal, Quebec, Canada, 1995.

[81] R. Vinter. Optimal Control. Birkasuser Boston, 2000.

[82] J. Vlach and K. Singhal. Computer Methods for Circuit Analysis and Design. Van Nostrand Reinhold, New York, NY, 2nd edition, 1994.

[83] S. Wang and C. Kim. Design optimization with response surface method and sensitivity. In WCSMO1, Dalian China, June 2001.

[84] F. Zhang, R.G. Longoria, R.F. Thelen, and D. Wardell. A simulation-based design study for a locomotive electric power system. In Electrimacs 2002, Montreal, CAN, August 2002.

Vita

Fu Zhang was born in Hernan, China on 26 April 1971, the son of Yijie Zhang and Liping Fan. He received the Bachelor of Engineering degree in Mechanical Engineering from XIDIAN University, Xi'an, China in 1992 and then received his Master of Engineering in Robotics and Industrial Automation from Beijing Institute of Technology in 1995. He then received his Master of Engineering in Mechanical Engineering from National University of Singapore in 1997.

He applied to the University of Texas at Austin and started graduate studies in August, 1999.

Permanent address: XIDIAN UNIVERSITY, South 36-106
ShanXi, Xi'an 710071
P.R. China
zhangfu@hotmail.com

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.