MODELING WITH RENORMALIZATION GROUP AND RANDOMIZATION

YU CHAO
(B.Eng., Harbin Institute of Technology)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2014

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirely.

I have duly acknowledged all the sources of information which have been used in this thesis.

This thesis has also not been submitted for any degree in any university previously.

Yu Chao

YU CHAO
14 May 2014

Acknowledgments

First and foremost I would like to express my sincere gratitude to my supervisor, Professor Wang Qing-Guo, who always graciously and patiently guides me throughout my research. He has been supportive since the days I began working and it has been an honor to be his Ph.D. student. His enthusiasm for research was contagious and motivational for me. His consideration and valuable advices inspires me to finish this thesis.

This thesis would not have been possible without the financial, academic and technical supports from National University of Singapore as well as Chinese Ministry of Education.

I also wish to acknowledge my friends in Singapore, China and elsewhere in the world for their support and concern. They are always there whenever I need help or advices. Especially, I want to express my appreciation to Ms. Gan Tian who always stays with me and spares no effort to give strong backing to me over the years.

Last but not least, I would like to thank my parents for their tremendous love, support, understanding and encouragement throughout my 20 years study. I sincerely hope this work makes you proud.

Contents

Summary

This thesis develops some new techniques to help assess models in statistical learning, obtain improved results in system identification, solve global optimization problems and find stabilizing parameter regions for control systems.

First, we propose a new method for model assessment based on Renormalization Group. A transformed data set is obtained by applying Renormalization Group to the original data set. The assessment is first performed on the data level by comparing two data sets to reveal informative content of the data. Then, the assessment is carried out at the model level, and the predictions are compared between two models learnt from the original and transformed data sets, respectively. The computational burden for model assessment is small since the proposed method requires only two models.

Second, we propose an improved system identification method with Renormalization Group. A coarse data set is obtained by applying Renormalization Group to a fine data set. The least squares algorithm is performed on the coarse data set. The theoretical analysis under certain conditions shows that the parameter estimation error could be reduced.

Then, we solve an outlier detection problem for dynamic systems. The outlier detection problem is formulated as a matrix decomposition problem and further recast as a semidefinite programming (SDP) problem. A fast algorithm is presented to solve the

SDP with less computational cost than the standard interior-point method. Construction of subsets of the raw data helps further reduce the computational burden. The proposed method can make exact detection of outliers when output observations contain no or little noise. In case of significant noise, a novel approach based on under-sampling with averaging is developed to denoise while retaining the salient behaviors of outliers, which enables successful outlier detection with the proposed method.

Next, we propose a brand-new method for global optimization through randomized group search in contracting regions. A population is randomly generated within the search region in each iteration. A small subset of them with top-ranking fitness values are selected as good points, whose neighborhoods are used to form a new and smaller search region, in which a new population is generated. The convergence of the proposed algorithm is analyzed.

Last, we propose a method for determining the stabilizing parameter regions for general delay control systems based on randomized sampling. We convert a delay control system into a unified state-space form and develop the numerical stability condition which is checked for sample points in the parameter space. These points are separated into stable and unstable regions by the decision function obtained from some learning method.

List of Figures

List of Tables

Chapter 1

Introduction

With the rapid development of science and technology, modeling methods become more and more important and have been applied in many fields such as industry, medicine, biology and finance. A model built from some modeling technique refers to a schematic description of a system, theory, or phenomenon that accounts for its known or inferred properties and may be used for further study of its characteristics. In the field of system identification, which has developed considerably since last century, mathematical models of dynamic systems are built from measured data. Information retrieved from an identified model enables researchers and engineers to carry out some modifications to improve its performance. For example, in control engineering, control techniques are designed to cause system variables to conform to some desired values. Present-day approaches [1] to control include classical control methods and modern control methods, which both have wide applications. Another example is to adopt optimization techniques, which are mathematically designed to find best values of some objective function under a set of constraints, to make a system as effective or functional as possible. Nowadays, abundant research efforts have been devoted to the study of optimization techniques for solving real life problems.

1.1 Modeling

With rapid advances in information technology, abundant data are generated in industry, medicine, finance and everywhere. Statistical learning is to find information in the data through modeling and solves the inference problems such as classifications and regressions [2, 3]. Great progress has been made in this field and there are many types of models available in the literature such as neural networks, decision trees and support vector machines with related training algorithms, and numerous successful applications have been reported. One may choose a model and apply a learning method to train its parameters so as to fit the data. A more complex model with more training usually gives a better data fit. However, the model performance is evaluated not only on how well a model fits the training data, but also on whether or not the model can give good predictions on the unseen test data.

Given a data set, one can always fit it to some model. Is such a model useful or reliable for prediction purpose? It is a very challenging problem to assess and select a model. C_p statistic, Akaik information criterion (AIC) and Bayesian information criterion (BIC) are well-known model assessment methods. They try to estimate in-sample error for model selection. It is shown [2, 4] that AIC tends to choose complex models whereas BIC often chooses simple ones since it penalizes heavily on complexity. However, the assessment based solely on in-sample error may lead to an overfitting problem as the unseen test data set is not taken into consideration and eventually only the performance on the test data matters in real environments.

Cross-Validation (CV) is probably the simplest and most popular method in model assessment and selection. It estimates the out-of-sample error. In the K-Fold Cross-Validation, the data are split roughly into K equal-sized folds. $K - 1$ folds are used for training while the remaining fold is for test to estimate the out-of-sample error. This is repeated for different combinations of training and testing folds. Another popular method which also estimates the out-of-sample error and suits for any loss function is the Bootstrap method, which uses all the data for resampling [5, 6]. The idea of Bootstrap is to draw data from the original set randomly with replacement till a new data set is formed with the same size as the original data. The above procedure is repeated for K times and a model is fitted to each set of K bootstrap data sets, respectively. Then the behavior of the fits can be examined [2]. It is seen that these methods need to split data into K sets and the resulting data sets are partially overlapped. For example, in a 10-Fold CV, every training set has 8/9 of its samples same as each of the other nine [7]. They require the training algorithm to be run for K times, which will increase the computational burden as much as K times [8]. One also needs to choose the parameter K which is a trade-off between bias and variance of the prediction errors [2]. A large K will usually obtain low bias and high variance prediction errors, whereas a small K will make prediction errors with high bias and low variance. Some researchers have tried to find substitutive criteria that provide same information as CV but do not need validation sets [9].

Compared with tremendous developments in learning techniques, there seems less recent progress on model assessment. Therefore, Chapter 2 tries to fill in this gap by presenting a totally new method for data and model assessment using Renormalization Group (RG).

Although the information obtained from model assessment helps to choose a relatively reliable model, it is also important to improve the stability and reliability of the model itself. To achieve this, one may employ proper techniques in specific applications. System identification is concerned with building mathematical models of dynamical systems from measured data [10–12]. The ordinary least squares (OLS) method has since been the dominant algorithm for parameter estimation due to its simplicity in concept and convenience in implementation [13, 14]. Given a data set, one can always get an estimate of OLS. It is known that the OLS estimate could be biased for a regression model with noise [15, 16]. And it is a very challenging problem to analyze the properties of OLS estimate analytically.

With regard to the asymptotic properties of the least square estimates, it is known that the OLS estimate will converge to its real value when the system is disturbed with white noise. Otherwise, when the system is disturbed with correlated noise, the OLS estimate could be biased. Griliches [17] gives the expression of the bias for ARAR(1,1) models. Phillips and Inder [18, 19] analyze the bias for simple first order ARARX models. Stocker [20] presents an expression of the bias for a common model but only gives very simple examples for illustration. He argues that for complicated models (e.g. models with higher order or with several exogenous variables), it is not practicable any more to get fully parameterized bias formulas. These would become very extensive even if the order of the model or the number of exogenous variables increases only slightly. Zheng [21, 22] proposes a bias-eliminated least squares (BELS) algorithm to identify system parameters. The instrumental variable (IV) method [23–25] is also a very efficient way to avoid correlated noise and reduce the estimate bias.

When the number of data points is limited, it is more difficult to analyze the bias. Many papers in the literature discuss the finite-sample bias of OLS estimate for very simple models. Even when the system is with white noise, Hurwicz [26] proves that the OLS method yields biased estimates of regression coefficients, and it is possible to obtain explicit formula for the bias in very small samples. The general structure of the bias is revealed by Shaman and Stine [27, 28]. Breton and Pham [29] provide an exact formula for the bias. Patterson [30] exploits Shaman and Stines characterization of the bias in higher order models. When the system is disturbed with correlated noise, unfortunately, it has been proved difficult to investigate finite-sample bias analytically [31]. In the absence of such results, Monte Carlo experiments will provide an alternative source of information [31]. Sargent [31], Tjøstheim and Paulsen [32] analyze the bias through simulation. Maeshiro [33] shows that the bias of OLS estimate is determined by two effects, the dynamic effect and the correlation effect. The former is the bias of the parameters when the disturbance is white noise; the latter is the effect that contaminates the parameters when the disturbance is correlated with the lagged dependent variable. When the two effects have opposite signs, the OLS estimate performs well in terms of bias.

It is desirable to reduce the noise effect when estimating parameters. Chapter 3 is to present an improved system identification method with Renormalization Group. By theoretical analysis and simulation, it is shown that the proposed method could get a better estimate under certain conditions.

In system identification, observations may not only be disturbed by noise, but also by outliers. An outlier may be defined informally as the one which deviates remarkably from the bulk of the available data [34, 35]. Outliers occur frequently in real life and may cause serious consequences in a wide variety of application fields such as network packet dropouts [36, 37], signal processing [38], image processing [39], mechanical devices [40], credit card fraud detection [41] and medical data [42]. In context of control and automation, outliers may occur in the observed signal due to sensor malfunctions and data transmission errors and could lead to poor performance of system identification [34]. It is imperative to detect and eliminate outliers for better signal processing. Additive outliers [43], which affect single observations, are our major concern. A common empirical way to pick up outliers is visual inspection of data charts based on engineers experience. Such a method is subjective and inaccurate. It becomes inappropriate in today's world of large and complex systems with huge data. The three-sigma rule [44] is a popular statistical technique for outlier detection. However, this procedure is not always effective in practice because the variance estimate is likely to be inflated by outliers. Recent statistic techniques includes linear and nonlinear filtering for data cleaning [34]. The linear filters change the character of the normal operating episodes and are generally ineffective, while the nonlinear filters, such as the MT-cleaner [45] and the Hampel filter [46], are more effective. However, the MT-cleaner must assume that the clean data obeys Gaussian distribution and the Hampel filter can behave badly with coarsely quantized data [34]. Robust regression methods such as the least median of squares (LMS) [47], the least trimmed squares (LTS) [48], the least absolute deviations (LAD) [49] and the iterative reweighted least squares (IRLS) [50], are inherently less sensitive to outliers. However, they are often difficult to implement [34]. Further, LMS and LAD are unstable [51], i.e., a small change of the data can produce a relatively large change in the identification result.

To compare different methods in the context of system identification, consider the following discrete-time system,

$$y_t - 1.9y_{t-1} + 0.95y_{t-2} = 0.05u_t, \quad t = 1, 2, \ldots, N,$$

where u_t is a step function. Let $\theta = [1.9, -0.95, 0.05]^T$. Suppose the observed output is disturbed by noise and outliers as

$$\bar{y}_t = y_t + z_t + e_t,$$

where $e_t \sim N(0, 0.05)$ is a white noise and z_t emulates outliers:

$$z_t = \begin{cases} 0.5, & \text{if } t = 40, \\ -0.7, & \text{if } t = 120, \\ 0, & \text{otherwise.} \end{cases}$$

The resulting output response under zero initial conditions is generated and is shown in Fig. 1.1 as the dotted cyan line. If this original data is used for system identification, the parameter estimation error of $||\Delta\theta||_2$ is 1.76 for the ordinary least square (OLS) method, and 0.23 for the instrumental variable (IV) method. If the extended three-sigma rule based on the residual series of dynamic modeling (the detail in Section 4.5) is used, 8 points are detected as outliers because some normal samples at peak and trough are mistaken as outliers, and the parameter estimation with the data excluding such points gives the error of 0.19. The corresponding response is the dash-dot blue line. If the Hampel filter is used, 5 points are detected as outliers with some normal samples being mistaken as outliers, and the parameter estimation with the data excluding such points gives the error of 0.068, which improves but is still not satisfactory. The corresponding response is the dashed green line. The parameter estimation errors with LMS, LTS (10% of trimming), LAD and IRLS are much large and shown in Table 1.1, where these four methods are all implemented in Matlab R2013a [52–55] with default parameter settings. It is seen from Table 1.1 that IV is preferred over other methods, which also explains why it is commonly used for colored noise in regression equation in the area of system identification. The parameter estimation error with IV is still significant. Therefore, new automatic and reliable outlier detection is highly desirable in system identification to obtain better parameter estimation. If the proposed method in Chapter 4 is applied, the estimation error is reduced to 0.0066 with the solid red line as the corresponding response.

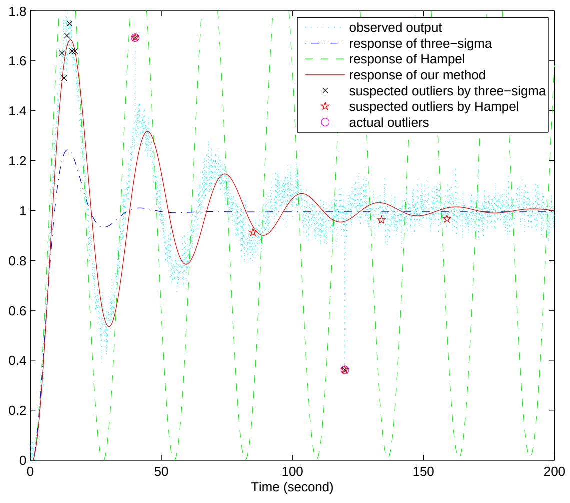


Figure 1.1: Illustrative example.

In some branches of engineering other than control, there have been recently vast amounts of literature on outlier detection. Hodge et al. [35] surveyed this area and

Table 1.1: Parameter estimation errors with different methods

Methods	$\|\Delta\theta\|_2$	$\frac{\|\Delta\theta\|_2}{\|\theta\|_2} \times 100\%$
OLS	1.76	82.92%
IV	0.23	10.82%
Extended three-sigma + IV	0.19	8.94%
Hampel + IV	0.068	3.20%
LMS	2.11	99.30%
LTS	1.52	71.53%
LAD	1.62	76.24%
IRLS	1.52	71.53%
Proposed Method	0.0066	0.31%

grouped relevant techniques into three types: supervised classification, semi-supervised recognition and unsupervised clustering. The first type techniques assume availability of the labels for both normal and outlying instances in a training data set. The drawback is that the accurate labels of training data, which are usually determined manually, might be exorbitantly expensive to be obtained [56]. Also, the assumption that outliers are available in training set is not very popular [56]. The second type of the techniques assume availability of the labelled instances for only one class: outliers or normal data points. The techniques that assume the availability of only the outliers are limited in use since it is difficult to cover every possible outlying behavior in the data [56]. In contrast, the techniques that assume availability of normal instances are relatively more popular, even though the full scope of normality needs to be known for generalization [35]. Unsupervised outlier detection techniques are most widely used since they do not assume availability of the labels for instances. These techniques usually assume that the normal instances occur much more frequently than outliers and obey some parametric statistical distribution [56].

Recently, Candès et al. [57] and Wright et al. [58] proposed a new unsupervised technique, Robust Principal Component Analysis (RPCA), which has already had a lot of important applications such as video surveillance [58], face recognition [59] and latent semantic indexing [60]. RPCA recovers a low-rank matrix from corrupted observations, that is, a given corrupted matrix is decomposed into a low-rank matrix and a sparse matrix, where the sparse matrix is thought of as a collection of outliers. The matrix decomposition problem is recast as a semidefinite programming (SDP) problem and researchers have proposed some first-order fast algorithms [61–64] to solve it. Although these methods are fast, there is a limitation that if the ideal matrices contain special structures such as Hankel and Toeplitz, the resulting matrices might not preserve these structures due to the use of singular value decomposition (SVD) in the algorithms. Another drawback of RPCA is that analysis of noise is neglected, though noise is likely to accompany with the signal.

Fazel et al. [65] tried to recover a low-rank matrix which is contaminated by noise by solving a rank minimization problem. This problem is also converted to an SDP problem. The general way to solve the SDP problem is using the interior-point method, which preserves the linear matrix structure. However, the interior-point method is quite slow and limited by the problem size. Liu et al. [66] developed a more efficient implementation of the interior-point method by exploiting the problem structure in the SDP formulation. This implementation is fast and performs well on large scale data. Liu et al. [66] assume that the signal is only corrupted by noise and solve a rank-minimization problem of a certain matrix,

$$\min_{L} \quad \mathbf{rank}(L),$$

where L is a low-rank matrix.

After comparing these methods, it is seen that a fast algorithm solving an outlier detection problem and preserving special matrix structure is highly desired. Chapter 4 aims to present such an algorithm. In addition, a realistic but complex situation is addressed where the observations are in presence of both noise and outliers.

1.2 Optimization and Control

Nowadays, optimization problems are ubiquitous in our daily life. In science and engineering, many practical problems such as decision making and system design and analysis can be formulated as optimization ones. To optimize is to find the best solution of a certain problem, such as minimizing cost or maximizing efficiency. A global optimum of an optimization problem, which is optimal among all possible solutions, is usually preferred than a local one, which is optimal within a neighboring set of candidate solutions. Global optimization is a very challenging problem and has attracted great research attention over decades. The standard form of an optimization problem is

$$\begin{aligned} \min_{x} \quad & f(x), \\ \text{subject to} \quad & g_k(x) \leq 0, \qquad k = 1, \ldots, p, \\ & h_l(x) = 0, \qquad l = 1, \ldots, q, \end{aligned} \tag{1.1}$$

where $x = [x_1, x_2, \ldots, x_n]^T$, $f(x)$ is the objective function to be minimized over the vector variable x, $g_i(x) \leq 0$ are inequality constraints and $h_i(x) = 0$ are equality constraints. The constraints determine the feasible region of (1.1). By convention, (1.1) defines a minimization problem. A maximization problem can be obtained by negating

$f(x)$. In the past decades, researchers have devoted great efforts to find the solution of the problem specified in (1.1) with a good deal of optimization techniques. The techniques are classified as either local or global algorithms. Most local optimization algorithms are gradient-based. Different gradient-based algorithms differ in the logic used to determine the search direction [67], and they only yield good results with functions which are continuous, convex and unimodal [68]. However, the problems in engineering sciences are usually complex, non-linear, non-convex and sometimes described by non-differentiable functions, demanding more efficient numerical methods for their solutions.

Global optimization algorithms, which are typically not gradient-based, provide a much better chance of finding the global or near global optimum than the local algorithms. It is important to note that no algorithm so far can surely guarantee convergence to a global optimum, and it may be more accurate to refer to these algorithms as having global properties [67]. Global optimization algorithms may be classified as either metaheuristics or deterministic algorithms [67]. One popular general purpose deterministic global optimization algorithm is the DIRECT algorithm [69]. The DIRECT algorithm makes use of Lipschitzian optimization to locate promising subregions in the design space. Each of these subregions is then further explored using a local search technique [67]. The DIRECT algorithm is only effective for low-dimensional cases. The computational burden will become extremely large when the problem size increases.

Metaheuristics for global optimization have become very popular since last century. These methods are typically inspired by some phenomena from nature and have the advantages of being robust, easy to implement and well suited for discrete optimization problems [67]. The drawbacks associated with these algorithms are high computational costs, poor constraint-handling abilities, problem-specific parameter tuning and limited problem size [67]. Boussaïd et al. [70] provides a comprehensive review of existing metaheuristics. Generally, the metaheuristics are classified either as single-solution based or population based.

The single-solution based metaheuristics start with a single initial solution and move away from it, describing a trajectory in the search region [70]. Among these single-solution based algorithms, the simulated annealing (SA) [71] and the tabu search [72] (TS) are representative and have been studied a lot. The major strengths of SA are that it optimises functions with arbitrary degrees on non-linearity, stochasticity, boundary conditions and constraints [73]. It is also statistically guaranteed of finding an optimal solution. However, it has its disadvantages too. It is very slow. Its efficiency depends on the nature of the surface it is trying to optimize [73]. However, the availability of supercomputing resources mitigate these drawbacks and makes SA a good candidate [73]. TS does not use hill-climbing strategies and its performance could be enhanced by branch and bound techniques [73]. However, the mathematics behind this technique was not strong. Furthermore, TS requires a knowledge of the entire operation at a detailed level and extra overhead in terms of memory usage and adaptation mechanisms compared with SA [73]. In-depth comparisons between TS and SA can be found in [74] and [75].

Population-based metaheuristics deal with a set of solutions rather than with a single one. There are three main steps in all the population-based metaheuristics. The first step is to randomly generate the initial population of individuals according to some solution representation. Each solution in the population is then evaluated for fitness value in the second step. The third step is to generate a new population by perturbation of the solutions in the existing population.

The most studied population-based methods are related to Evolutionary Computation (EC) and Swarm Intelligence (SI) [70]. EC algorithms are inspired by Darwins evolutionary theory, where a population of individuals is modified through recombination and mutation operators [70]. Popular EC includes the genetic algorithm (GA) [76] and differential evolution (DE) [77]. GA perhaps seems to be the most popular algorithm at present. Its advantage lies in the ease of coding and inherent parallelism [73]. It often locates good solutions. The use of mutation introduces new information gene pool to makes GA less likely to get stuck in local optima. However, it has some drawbacks. GA requires very intensive computation and hence is slow [73]. It also has the coding accuracy problem if solutions are represented in binary or Gray code. DE was proposed by Storn and Price [77] for global optimization over continuous search region. Its theoretical framework is simple and requires a relatively few control variables but performs well in convergence [78]. Lately, DE has been applied and shown its strengths in many application areas though it is slow [78]. The performance of DE is sensitive to the mutation strategy and control parameters. Mallipeddi et al. [79] employed an ensemble of mutation strategies and control parameters from a pool of distinct mutation strategies along with a pool of values for each control parameter to produce offspring through competition. They showed that their method outperformed conventional DE and several state-of-the-art parameter adaptive DE variants. In Swarm Intelligence (SI), the idea is to produce computational intelligence by exploiting simple analogs of social interaction, rather than purely individual cognitive abilities [70]. A representative of SI is the particle swarm intelligence (PSO) [80], which was proposed about the same time as DE. PSO is simple in concept, easy to implement and computationally efficient. However, it is more natural for continuous optimizations than discrete ones [78]. Recently, some researchers make combination of different methods to obtain more satisfactory optimization results. For example, Moradi et al. [81] proposed a novel optimization method combining GA and PSO to optimize the locations and sizes of distributed generation sources in distribution systems. Valdez et al. [82] described a hybrid approach for optimization combining PSO and GA using fuzzy logic, which is shown to outperform both individual optimization methods on a set of benchmark functions.

The designer should be aware that a metaheuristic will be successful on a given optimization problem if it can provide a balance between the exploration (diversification) and the exploitation (intensification) [70]. In addition, no single optimization algorithm exists that will solve all optimization problems. The performance of each algorithm would be heavily dependent on the nature of the problem itself and the heuristics used [73]. Chapter 5 aims to present a brand-new population-based method for global optimization problems, which is stand-alone and not related to any of the existing population-based methods.

Besides optimization techniques, control related designs also retrieve information from a model to make modifications to improve its performance. Finding stabilizing regions for control systems in parameter space becomes important in recent years. Stabilizing parameter regions will be instructive for controller tuning with greatest robustness or controller optimization with regard to other specific indices. Most papers in the literature discuss about the stabilizing parameter regions for proportional-integral-derivative (PID) controllers. Wang et al. [83] designed a quasi-Linear Matrix Inequality method to compute the stabilizing parameter regions of multi-loop PID controllers, but it only dealt with systems with no time delays. Lee et al. [84–86] established some stability conditions by simple P or PI controllers for a class of unstable processes with time delays, but the application of their methods is confined to single-input single-output (SISO) systems whose transfer functions only have one zero. Nie et al. [87] gave a frequency method to calculate the loop gain margins of multivariable feedback system. Liu et al. [88] introduced a fast calculation approach for PI controller stable region based on D-partition method. Wang et al. [89] presented an effective graphical method to obtain exact P controller gain ranges for two input two output (TITO) systems with input time delay. However, this approach could not handle systems with state-delays. Some other methods can be found in [90–95]. All the methods seek the solutions for the stabilizing parameter regions for limited classes of plants or controllers. Therefore, Chapter 6 designs a general algorithm for determining stabilizing parameter regions for delay control systems based on randomized sampling.

1.3 The Scope of This Thesis

In Chapter 2, we present a totally new method for data and model assessment using Renormalization Group (RG). The proposed method produces a new data set from the given data set with the members of the former different from those of the latter, which differentiates our method from the existing ones that only divide the given data into subsets, and thus create no new data. An assessment is performed at the data level without employing any learning method, whereas the domain works only make assessments at model level. For data assessment, new indices are introduced to quantify the consistency of two data sets and non-randomness of the given data. The computational cost is extremely small since it involves no learning. If the assessment result shows that the given data is random, modeling it is meaningless. Only when the given data is informative, should one proceed to model it. For model assessment, our method compares predictions of two models leant from the give data and the transformed data, respectively. The prediction consistency and model reliability are defined accordingly. This assessment relies on two models one of which has much smaller data size and thus much less computational burden, whereas K-fold CV or similar methods train K models with K usually much greater than 2, typically set at 10.

In Chapter 3, we present an improved system identification method with Renormalization Group (RG). The proposed method forms a data set based on the system inputs and outputs. A new data set is produced from the given data set with the members of the former different from those of the latter. Performing the least squares algorithm, we obtain an estimate based on the given data and another based on the new data. Comparing the two estimates through theoretical analysis and simulation, we find that the proposed method could get a better estimate under certain conditions.

In Chapter 4, we consider an outlier detection problem for a signal from a dynamic system, which is formulated as a low-rank and sparse matrices decomposition problem:

$$\begin{aligned} \min_{S} \quad & \textbf{rank}(L) + \gamma \|S\|_0 \\ \text{subject to} \quad & L + S = D, \end{aligned}$$

where D is a Hankel matrix formed from the measurement signal of a dynamic system, L is a low-rank matrix, S is a sparse matrix and γ is a trade-off parameter. This problem is further recast as a semidefinite programming (SDP) problem. To solve the SDP problem, a fast algorithm is presented which preserves Hankel matrix structure with great reduction of computational cost over the standard interior-point method. The computational burden is further reduced by proper construction of subsets of the raw data without violating low rank property of the involved matrix. In addition, a realistic but complex situation is addressed where the output observations are corrupted by both noise and outliers. In this case, we propose a novel approach based on under-sampling and averaging to reduce noise while keeping the salient behaviors of outliers, whereas the existing filtering methods smooth both noise and outliers. Better parameter estimation is obtained with the recovered "clean" data than that with the raw data

In Chapter 5, we present a brand-new population-based method for global optimization problems. The proposed method is stand-alone and not related to any of the existing population-based methods. It has two key novelties. Firstly, the region in which each population lies changes and contracts exponentially, which guarantees convergence of the proposed algorithm. Secondly, each population is generated with randomization, where the size of random samples, is chosen [96] to ensure that the empirical minimum is an estimate of the true minimum within a predefined accuracy with a certain confidence. It is shown that the proposed method converges and the convergence to local or global optima is analyzed. Our method has no restrictions on the properties of objective functions. It works on both constrained and unconstrained problems. Also, our method applies to both continuous and combinatorial optimization problems. The implementation of the proposed method is easy. Extensive simulation on benchmark problems shows that the proposed method is fast and reasonably accurate.

In Chapter 6, we propose a method for determining stabilizing parameter regions for general delay control systems based on randomized sampling. We assume that each unknown parameter follow the uniform distribution in a given range. Then, we generate a certain number of random sample points in the parameter space. Next, we convert a delay control system into a unified state-space form and develop an efficient LMI stability criterion. Each point in the parameter space is checked with the developed stability criterion. These points are separated into stable and unstable regions by the decision function obtained from some learning method. The proposed method is general and applied to a much broader range of systems than the existing methods in the literature.

Chapter 2

Model Assessment through Renormalization Group in Statistical Learning

2.1 Introduction

In statistic learning, model assessment is vital for evaluating the usefulness and reliability of a model. Existing methods [2] for model assessment includes C_p statistic, Akaik information criterion (AIC), Bayesian information criterion (BIC), cross-validation (CV) and the Bootstrap method. C_p statistic, AIC and BIC estimate in-sample error, which may lead to overfitting problems. CV and Bootstrap estimate out-of-sample error but split data into different sets, which are partially overlapped. Training models on different data sets may increase the computational burden. There seems less recent progress on model assessment compared with vast developments in learning techniques. This chapter aims to present a totally new method for data and model assessment using Renormalization Group (RG). RG was first proposed to study the critical phenomena in the quantum field in 1971 by Kenneth G. Wilson, who won the Nobel Prize for physics in 1982 [97], due to this great contribution. RG is widely used to analyze various physical problems [98, 99]. It is observed that some physical system may enter a critical point where certain physics quantities such as the Hamiltonian, the transformation function and the coupling constants have the property of "scale invariance" [100]. Renormalization Group designs some Renormalization Group transformation (RGT) to relate macroscopic physics quantity to microscopic one and invokes "scale invariance" to solve the problem. To see quickly our RG idea in model assessment, imagine that 100 data points are taken on the function $y = x^3$ with x in $[0, 1]$ and are fitted to some model. Now every 10 points nearby are grouped to one new point by averaging and the resulting 10 new points are fitted to a new model. Obviously, one expects such two models to perform similarly in the given interval. On the other hand, a pure random data set will produce two models by chance and they perform totally differently.

Technically, the proposed method groups the given data set into a RG data set, train one model with the given data and another model with the RG data, and compare their predictions. The consistent predictions between two models indicates informative data and reliable models. The contributions of this chapter can be summarized as follows.

- The proposed method produces a new data set from the given data set with the members of the former different from those of the latter, whereas the existing methods mentioned above only divide/separate the given data and the data points in the new sets are the same as those in the given set, and thus create no new data.

- An assessment is made at the data level before any learning method is applied to train a model, whereas the domain works carry out model assessment only. For this, we introduce new indices to quantify the consistency of two data sets and non-randomness of the given data. The required computation is extremely fast as it involves no learning. With this assessment, if the given data is random, it is meaningless to model it. Only when the data is not random, should one proceed to model it.

- At the model level, our assessment is to compare predictions of two models leant from the give data and the transformed data, respectively. The prediction consistency and model reliability are defined accordingly. This assessment relies on two models one of which has much smaller data size and thus much less computational burden, whereas K-fold CV or similar methods train K models with K usually much greater than 2, typically set at 10.

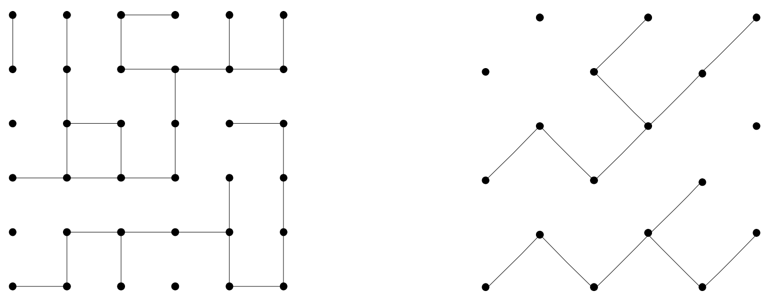
The rest of this chapter is organized as follows. In Section 2.2, Renormalization Group is reviewed. Section 2.3 presents the proposed method. Section 2.4 details the implementation issue of the method. Section 2.5 addresses the theoretical issues of the method with assessment criteria. Section 2.6 gives simulation studies of some well-known examples. Section 2.7 discusses rich variants of RG. Section 2.8 concludes the chapter.

2.2 Review of Renormalization Group

Renormalization Group (RG) was first introduced in [97] to study the critical phenomena in physics. A physically different state of a substance is called a phase. A second-order phase transition means that a substance transforms from one phase to another with continuous energy change. A critical point refers to a situation where the substances of two states are fully mixed with each other both in macroscopic and microscopic views. At a critical point, a phase boundary does not exist anymore. For example, in the classical Heisenberg-type model [101], all lattice spins tend to align at a low temperature and this state is one phase whereas at a high temperature, lattice spins orient randomly and this state is another phase. At a certain temperature T_c, there is no boundary between these two phases and this temperature is called the critical temperature. The critical phenomena are unusual and attracted great attentions for study. RG was invented [97] to analyze the critical behavior of a physical system. It was shown that when a second-order phase transition occurs, some physical quantities of the system such as the Hamiltonian, the free energy, the transformation function and the coupling constants would not change under a RG. This property is used to determine the critical values of relevant parameters at a critical point.

Let us look at bond percolation [102] to have a concrete idea of RG. For a system with a two-dimensional (2D) square lattice, a bond exists between two neighboring sites with a probability p, as exhibited in Figure 2.1a. If a set of sites are connected by bonds, we call this set a cluster. When p is large enough, there will be a cluster extending from one side of the lattice to the other, for instance, from left to right. When p is small, this may not be possible. Thus, there must be a critical probability p_c, such that when $p > p_c$, a cluster exists and extends the whole lattice from one side to another whereas for $p < p_c$, there is no such cluster. Therefore, $p > p_c$ and $p < p_c$ define two phases of this model and $p = p_c$ is the critical probability. Binney [102] showed that $p_c = \frac{1}{2}$ for a 2D square lattice.

A Renormalization Group Transformation (RGT) is now applied to the original lattice in Figure 2.1a to get a transformed lattice. Every second site in the original lattice is knocked out and a bond is set up between two neighboring sites of the new lattice if there are two bonds joining those two sites on the old lattice, which gives rise to the renormalized lattice in Figure 2.1b.



(a) Original lattice. (b) Renormalized lattice.

Figure 2.1: Original and renormalized lattices.

Figure 2.2 shows all the bond configurations on a square on the old lattice which can give rise to a bond on the new lattice and a probability is also labeled for the occurrence of each configuration. Then the probability p_t that a bond exists between two sites on the

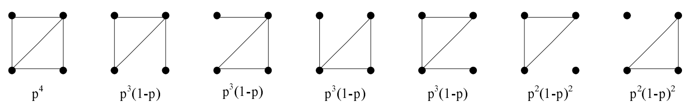


Figure 2.2: The bond configurations on a square.

new lattice is obtained as

$$\begin{aligned} p_t &= p^4 + 4p^3(1-p) + 2p^2(1-p)^2 \\ &= 2p^2 - p^4. \end{aligned} \tag{2.1}$$

When it is at the critical point, it follows from the "scale invariance" property under RGT that $p_t = p = p_c$, and (2.1) is then solved with

$$p_c = \frac{\sqrt{5}-1}{2} = 0.618. \tag{2.2}$$

However, this number is not equal to 0.5. The errors may be due to the following two causes [102].

- Some pairs of sites are connected on the original lattice, but they are separated on the renormalized lattice, for example, the sites A and B in Figure 2.3.
- The bond marked with 1 in Figure 2.3 on the original lattice affects the occurrences of both $1'$ and $2'$ on the renormalized lattice. In other words, $1'$ and $2'$ are not completely independent with each other, which does not fully meet the requirements of a percolation system.

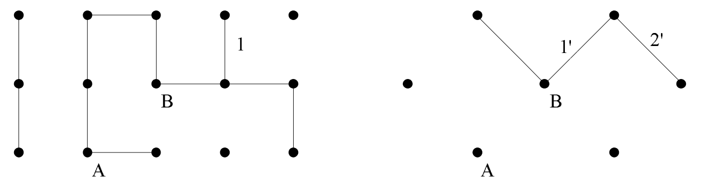


Figure 2.3: Errors in p_c estimation from renormalization.

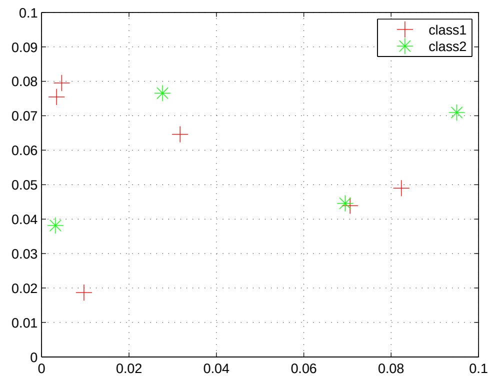
In short, the RG may lead to errors but is simple and effective to estimate critical parameter values. RG has been also applied successfully to dynamical systems [103], Ricci flow [104] and stock markets [105].

2.3 The Proposed Method

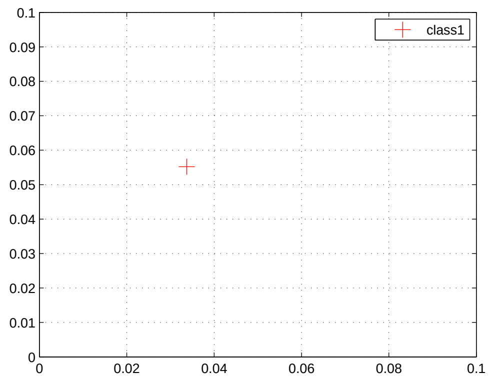
Consider the classical binary classification problem in statistical learning with a data set, $\mathbf{S} = \{S_1, S_2, ..., S_N\}$, with $S_i = (x_i, y_i), i = 1, 2, ..., N$, $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$. A learning method is to determine the decision function $f(x)$, which can classify the data with prediction or generalization capacity. The present chapter is not to develop a new learning method, but to propose a new approach based on RG to assess the degree of randomness of the data and reliability of a model built from a learning method for prediction, assuming that the data are from a process with a fixed probability density. The core of this approach is the "scale invariance" property of RGT. It follows from this property that the information contained in the original data set is invariant under RGT, that is, the transformed data set obtained from RGT should contain the same information as in the original data set. In the context of statistical learning, the information of the data set refers here to the probability density function, or the relationship from x to y. Its two extremes are the purely random and completely deterministic cases. We will define and compute an index on measure of such information on the data set. Furthermore, a model learnt from the data set reflects its information. If the information is same for the original and transformed data sets, then two models learnt from them, respectively, should be consistent with each other in terms of their predictions. Hence, we will also define and compute an index on measure of model consistency. It is found that this index will not be zero for the pure random data. This makes it necessary to find the index with the case of completely randomized labels and re-scale it for a general case to define another index on measure of model reliability. The idea and steps of the proposed approach will be described and illustrated by a 2D example in this section, while design of RGT and relevant indices will be discussed in the next two sections, respectively.

The first step of the proposed method is to perform a RGT on the given data set to obtain a transformed data set. For easy reference, we call the given data set, **S**, as the fine data set and the model obtained from it as the fine model. The data set $\hat{\mathbf{S}} = \{\hat{S}_1, \hat{S}_2, ..., \hat{S}_{\hat{N}}\}$, with $\hat{S}_j = (\hat{x_j}, \hat{y_j}), j = 1, 2, ..., \hat{N}$, $\hat{x_j} \in \mathbb{R}^p$ and $\hat{y_j} \in \{-1, 1\}$ obtained from a RGT on the fine data set is called as the coarse data set and the resulting model as the coarse model. A RGT on the fine data set is to group a number of data points of the fine data set into one data point in the coarse data set in a systematic way. There are different ways to do this grouping, which will be discussed in detail in the next section. One example of RGT is to group all the data of the fine set in a geometric unit of the fixed shape and size to one point of the coarse set. In general, one RGT transforms the fine set to many groups and one group will become one data point in the coarse set. For each group, one has to specify its representative label ($\hat{y}$) and its representative feature values ($\hat{x}$) to define one data point in the coarse set. The appealing way to assign the group label seems the majority rule of the labels of the points in that group, while its representative $\hat{x}$ may be determined by the simple average of the x_i of the major class. For example, Figure 2.4a and Figure 2.4b show such a group of 10 points in the fine set and the resulting point in the coarse set using the above rules.

For clarity and illustration, we construct a 2D example to explain each step of the proposed method in detail with numerical results. Let the fine data have two features, $x_i = (x_i(1), x_i(2))$. Let the fine set have 4000 data points in the square of [-1,1], which are drawn randomly with uniform distribution. Specify a decision function as $x_i(2) =$



(a) A group of data in the fine set.



(b) One point in the coarse set.

Figure 2.4: RGT.

$x_i^2(1) - 0.5$. Consider first the deterministic case where each data point is assigned with the green label if it is located on and above the parabola, otherwise with the red label. The resulting fine set is depicted in Figure 2.5a. Suppose that we use the majority and average rules mentioned above. The coarse set is then obtained and shown in Figure 2.5b.

The second step of the proposed method is to assess the information of the given data. Suppose that the label for a coarse unit is assigned according to the majority rule and the coarse label for each fine data point in this unit is defined as the same as its unit label. This enables us to compare the fine and coarse labels at each fine data point and define the data consistency as follows:

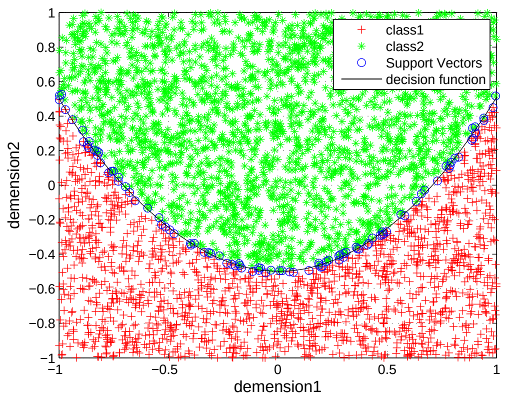
$$\mathcal{C}_D = 1 - \frac{1}{2}\frac{\sum_{i=1}^{N} |L(x_i) - \hat{L}(x_i)|}{N}, \tag{2.3}$$

where $L(\cdot)$=1 or -1, stands for the fine labels and $\hat{L}(\cdot)$=1 or -1 for the coarse labels. A high index value means a high extent of uniformity of two data sets, which should imply that the given data is informative. For the above example, the data consistency index is 0.9818, indicating a high consistency of two data sets.

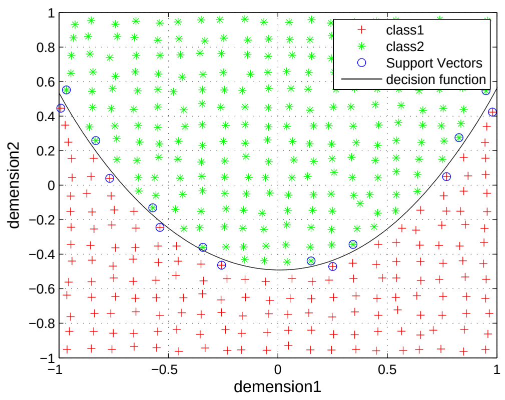
The third step of the proposed method is to apply some learning method on both fine and coarse data sets to get the fine and coarse models, $f(x)$ and $\hat{f}(x)$, respectively. Any learning method can do but the same one should be used on both data sets to avoid inconsistency due to application of different methods. For this study, the support vector machines (SVM) is chosen as our classification tool, since it is popular and representative and has many attractive features and emphatic performance in many applications [106–109]. The SVM sovles the following optimization problem [110, 111]:

$$\begin{aligned} \min_{\omega,b,\xi} \quad & \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\omega^T\phi(x_i)+b) - 1 + \xi_i \geq 0 \\ & \xi_i \geq 0. \end{aligned}$$

where ϕ is a mapping from x_i to a higher dimensional space and $C > 0$ is the penalty parameter. In our simulation, the LibSVM kit [112] is employed for the above 2D example and produces the fine and coarse models in Figure 2.5 marked with black curves. Then



(a) The fine set and model.



(b) The coarse set and model.

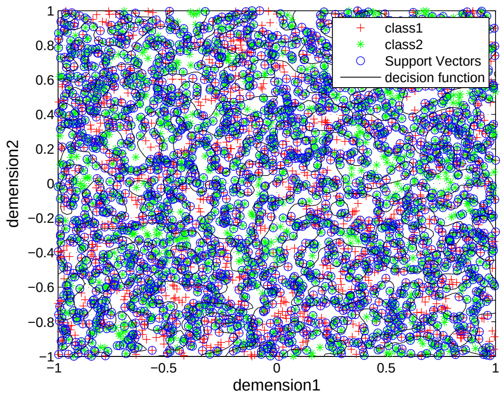
Figure 2.5: Deterministic 2D case.

we assess the reliability of the fine model, comparing predictions of the fine model with those of the coarse model. A model consistency index $\mathcal{C}_M$ is defined as

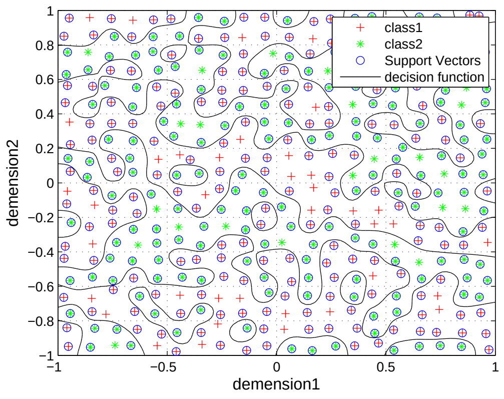
$$\mathcal{C}_M = 1 - \frac{1}{2}\frac{\sum_{i=1}^{N} |\text{sign}(f(x_i)) - \text{sign}(\hat{f}(x_i))|}{N}, \tag{2.4}$$

where $\text{sign}(f(x_i))$ and $\text{sign}(\hat{f}(x_i))$ are the prediction labels of $f(x_i)$ and $\hat{f}(x_i)$ on x_i in the fine data set, respectively. A high index value means a high degree of agreement of predictions of two models and indicates a high similarity of two models, which should imply that the learnt models are reliable. The model consistency index also captures the information contained in the data. For our example, the model consistency index is 0.9970, indicating a very high reliability of the models, which is of course true, as the data are constructed with a perfect decision function.

Next, we consider a purely random data case for comparison (more realistic cases will be shown in Section 2.6). Take all x_i from the above case or randomly choose a new set of 4000 of x_i, which does not matter for this study. Assign now their labels randomly, which yields the data shown in Figure 2.6a. The majority and average rules above are applied to get the coarse set shown in Figure 2.6b. The data consistency index is 0.6282, a much smaller value, which actually indicates that the given data is not informative as will be shown in Section 2.5. Figure 2.6 also shows the fine and coarse models marked with black curves. For this case, the model consistency index is 0.6743, a much smaller value compared with the previous case. The model is not reliable though the consistency index is not equal to zero but about the lowest, which will be shown in Section 2.5. This is also obvious since the decision functions in Figure 2.6 are messy and useless.



(a) The fine set and model.



(b) The coarse set and model.

Figure 2.6: Pure random 2D case.

2.4 Design of RGT

A RGT is a way to group the fine data and choose a representative in each group [105]. There are many methods for designing RGT. For example, Kadanoff's "spin decimation" transformation was very successful in the traditional Ising model. Other methods include Migdal-Kadanoff approximation, cumulant expansion and cluster expansion [100, 113]. The choice of methods depends on particular applications. For our problem of assessment of data-driven models, we present two broad methods of grouping data for RGT based on geometric and distributional considerations in the feature space, respectively.

2.4.1 Geometrical Grouping

Choose a geometric unit of fixed size. For the 2D case, possible geometric units include, but not limited to, a square, rectangle, and triangle. It is trivial to have their n-dimensional counterparts. We have used squares in the previous section for a 2D example. We can have a cube for 3D and a hypercube for n-D. Suppose a hypercube for illustration. Fill the given feature space with a grid of hypercubes of equal size, one by one without gap or overlap. Include in the coarse set a non-empty hypercube which has at least one fine data point in it and exclude all empty hypercubes. The resulting non-empty hypercubes are numbered as $j = 1, 2, ..., \hat{N}$.

The next task of RGT is to form a coarse data set. One has to specify its features, $\hat{x}_j$ and label, $\hat{y}_j$, for unit j, based on all $S_i = (x_i, y_i)$ of the fine set in the same unit. The label, $\hat{y}_j$, for unit j, is usually determined according to the majority rule on all (x_i, y_i) of the fine set in this same unit. Let unit j have N_j^+ of such x_i with $y_i = 1$ and N_j^- of x_i with $y_i = -1$. Then the majority rule is defined as follows.

- if $N_j^+ > N_j^-$, then $\hat{y}_j = 1$;
- if $N_j^+ < N_j^-$, then $\hat{y}_j = -1$;
- if $N_j^+ = N_j^-$, then $\hat{y}_j$ is the same as the first counted y_i in unit j.

Now turn to the issue of setting features. Possible ways to get $\hat{x}_j$ are

- calculating the average of all x_i in units with the major class only,
- calculating the average of all x_i in units,
- choosing the geometric center of the unit, and
- randomly choosing one x_i with the major class.

It is found that these different rules make little difference in model assessment, basically because the unit is of small size and the resultant x_i from different rules are quite close to each other. Thus, in the rest of this chapter, the first rule above to get $\hat{x}_j$ is adopted.

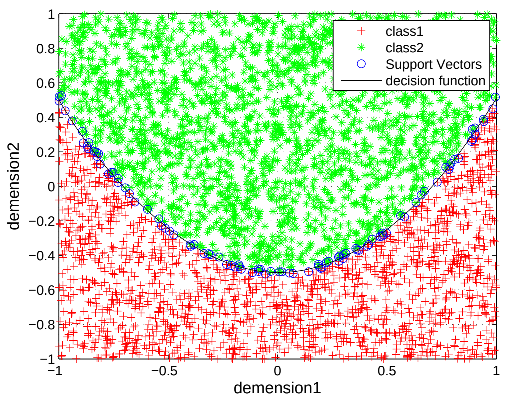
It follows from the above description of RGT that the major parameter of a RGT is the number of units, or the size of a unit in each geometric grouping method. It depends on the transformation ratio, r, which we define as the number of geometric units (the data points in the coarse set) to the number of data points in the fine set. If this ratio is small, there will be more fine data points in each unit on average. The majority makes good sense with less chance error, but it will give fewer data points in the coarse set, which may have negative effect on modeling accuracy. On the other hand, a large ratio will make less sense of majority but have more coarse data points. In the previous example, the side of each small square is chosen 0.1 and the ratio is approximately to 1/10. To see effects of r, we vary r and run simulations for the 2D example in the previous section. It follows that for the deterministic case, the consistency indices, $\mathcal{C}_D = 0.9700, 0.9818, 0.9515$ and $\mathcal{C}_M = 0.9985, 0.9970, 0.9922$, for $r = \frac{1}{5}, \frac{1}{10}, \frac{1}{20}$, respectively, while for the pure random case, $\mathcal{C}_D = 0.5740, 0.6282, 0.6560$ and $\mathcal{C}_M = 0.7107, 0.6743, 0.6275$, for $r = \frac{1}{5}, \frac{1}{10}, \frac{1}{20}$, respectively. It is seen that the ratio in the range of $\frac{1}{5} - \frac{1}{20}$ does not affect the assessment much. In the rest of this chapter, the ratio of $\frac{1}{10}$ is adopted.

2.4.2 Distributional Grouping

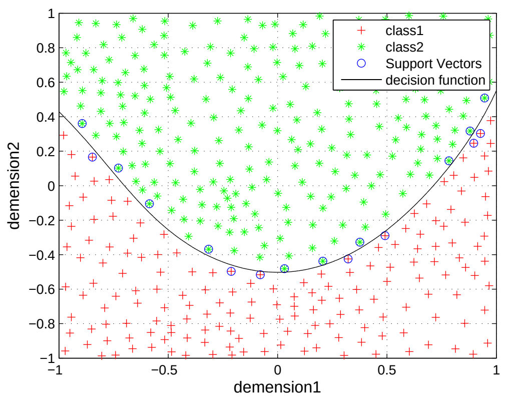
We may also design a RGT based on data distribution by using some clustering method. For example, the k-means clustering algorithm may be adopted. The k-means clustering groups N observations into k clusters [114, 115]. The first step is to define k centroids, one for each cluster. The second step is to associate each x_i with the nearest centroid. The third step is to re-calculate k new centroids as barycenters of the clusters obtained from the previous step, until the k centroids do not change any more. This algorithm minimizes the within-cluster sum of squares.

When a clustering method is applied to group the fine data to clusters, the resulting clusters play the same role as the units in the geometrical grouping. k/N is equal to the transformation ratio r and k is chosen based on the same guidelines as for the latter. And the rules to determine the features and labels of a cluster to form a coarse data point can be also same as those for the units before.

To see how the clustering method works as RGT, let us re-visit the 2D example in the previous section. Take $k/N = 0.1$, which is equivalent to the transformation ratio of 0.1, used before. The fine data set and the fine model are shown in Figure 2.7a (same as Figure 2.5a). Perform the k-means clustering algorithm, which produces 400 clusters. The corresponding coarse data and coarse model are shown in Figure 2.7b. The consistency



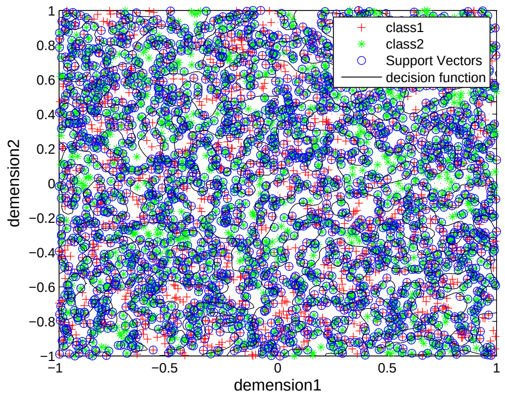
(a) The fine set and model.



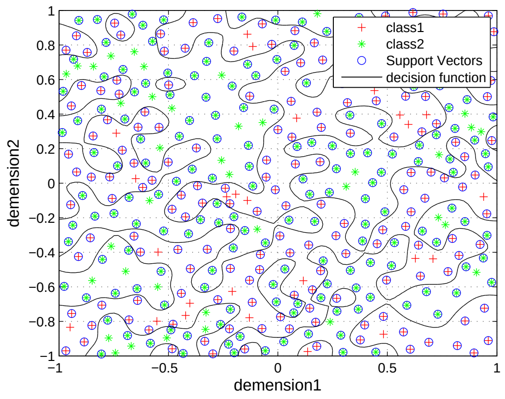
(b) The coarse set and model.

Figure 2.7: The k-means clustering algorithm—deterministic.

indices are computed as $\mathcal{C}_D$=0.9832 and $\mathcal{C}_M$=0.9960, almost same as in Section 2.3, very high values. The fine and coarse data and models for the pure random case are shown in Figure 2.8. Its consistency indices are found to be $\mathcal{C}_D$=0.6300, nearly the same as in



(a) The fine set and model.



(b) The coarse set and model.

Figure 2.8: The k-means clustering algorithm—pure random.

Section 2.3, and $\mathcal{C}_M$=0.4973, a low value, which is lower than before.

2.5 Assessment Criteria

With a RGT, one gets a coarse data set. It is possible to compare it with the fine set to assess the information contained in the fine data. When a learning method is applied to both data sets, one obtains two models and can compare their predictions to assess reliability of the fine model. This section will develop relevant indices to measure data information and model reliability with illustrations.

2.5.1 Data Information

There are many different learning methods available to train a model from a data set. Each method has its assumptions and tuning parameters, and will inevitably bring in its error or bias. Thus, the performance of any learnt model will depend, not only on the data, but also on the method and parameters chosen. It will be highly desirable to analyze only data to know how informative it is, regardless of a learning method adopted. It would be meaningless to model a data set if it is purely random. To find a suitable information index on the fine data, we look at how random the labels assigned by the majority rule for the coarse data are. Imagine that if the fine data is purely random, then the labels on the fine data are random, and each unit or cluster in the coarse feature space has $1/r$ fine data points with equal number of each label on average, which gives rise to equal probability for each label of the coarse unit/cluster. In other word, labels of the coarse data are random. Suppose that the coarse labels for all the fine data points in a coarse unit/cluster are same and defined as that of the majority. One can then compare the coarse label with the fine label on each fine data point. Their consistency will be a low value determined by chance.

On the other hand, if the fine data is deterministic, then the labels on the fine data are divided by the decision function, and each unit or cluster in the coarse feature space has $1/r$ fine data points with the same label, which in turn assigns that label to the coarse unit/cluster, unless the unit/cluster intersects with the decision function. In other word, labels of the coarse data are deterministic except for boundary units/clusters. Then, when one compares two labels of two data sets at each fine data point, and their consistency will be almost one. In the view of the above observations, we thus define the consistency index on the given fine data in (2.3).

To make the value of $\mathcal{C}_D$ in (2.3) more meaningful, we also evaluate the index for the corresponding random case. Given the fine data set **S**, keep x_i unchanged, but assign their labels randomly. Apply the same RGT as before, which results in the same units/clusters as before, but the majority rule will assign a new set of coarse labels due to changes of labels of fine data. We then evaluate the consistency index on the so-formed fine data, denoted by $\underline{\mathcal{C}_D}$. This number shall give the lower bound for this consistency index. The non-randomness index of a given data set is then defined as

$$\mathcal{D} = \frac{\mathcal{C}_D - \underline{\mathcal{C}_D}}{1 - \underline{\mathcal{C}_D}} \times 100\%. \tag{2.5}$$

The smaller $\mathcal{D}$ is, the more randomly the data distributes. The larger $\mathcal{D}$ is, the less randomly the data distributes. For the 2D example, the deterministic case yields $\mathcal{D} = 95.10\%$ from (2.5) with $\mathcal{C}_D = 0.9818$ and $\underline{\mathcal{C}_D} = 0.6282$ from Section 2.3. The deterministic case gives a high value of $\mathcal{D}$, indicating a high non-randomness of the data, whereas the pure random case gives $\mathcal{D}$ equal to 0.

For applications, one needs to know $\underline{\mathcal{C}_D}$. We now present the geometric grouping case for its calculation. Suppose that a RGT randomly groups N data into $\hat{N}$ identical units with the distribution, $Z = N_1 \times N_2 \times ... \times N_{\hat{N}}$, where $N_j = N_j^+ + N_j^-$, N_j denotes the number of data in the jth unit, while N_j^+ and N_j^- denote the numbers of '1' class and '-1' class data in the jth unit, respectively. It follows that

$$\underline{C_D} = \frac{\sum_{j=1}^{\hat{N}} max(N_j^+, N_j^-)}{N}. \tag{2.6}$$

We use the expectation $\boldsymbol{E}(\mathcal{C}_D)$ in place of $\underline{C_D}$ in (2.5). Since the $\hat{N}$ units are identical, (2.6) reduces to

$$\boldsymbol{E}(\mathcal{C}_D) = \frac{\hat{N}}{N}\boldsymbol{E}(max(N_1^+, N_1 - N_1^+)), \tag{2.7}$$

where the random variables N_1 and N_1^+ obey the binomial distributions, $N_1 \sim B(N, \frac{1}{\hat{N}})$ and $N_1^+ \sim B(N_1, \frac{1}{2})$. It is easy to verify that

$$max(N_1^+, N_1^-) = \frac{N_1}{2} + |N_1^+ - \frac{N_1}{2}|. \tag{2.8}$$

Substituting (2.8) into (2.7) gives

$$\boldsymbol{E}(\mathcal{C}_D) = \frac{\hat{N}}{N}\boldsymbol{E}(\frac{N_1}{2} + |N_1^+ - \frac{N_1}{2}|). \tag{2.9}$$

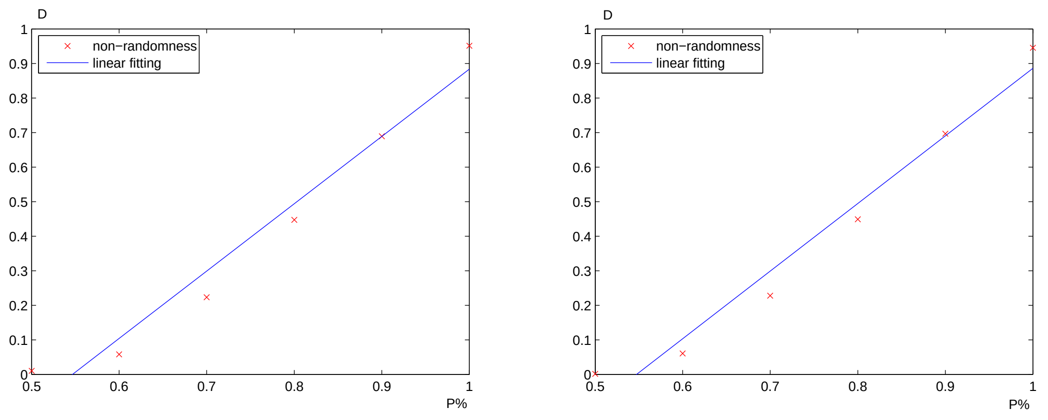
Note that if $N_j = 0$, (2.9) still holds. However, the proposed method excludes empty units. We can use (2.9) when the transformation ratio r is less than 0.1, where the probability of appearance of an empty unit is very small. To get exact $\boldsymbol{E}(\mathcal{C}_D)$ which excludes empty units, it is possible to use the enumeration method for simple cases. For example, when $N = 20$ and $\hat{N} = 2$, then it can be shown that $\boldsymbol{E}(\mathcal{C}_D) = 0.6254$. For a general case where N and $\hat{N}$ are large, we employ the Monte Carlo method to simulate $\boldsymbol{E}(\mathcal{C}_D)$ based on all possible data distributions. Simulation results show that $\boldsymbol{E}(\mathcal{C}_D)$ is related to $\frac{\hat{N}}{N}$, which equals to the transformation ratio r. Table 2.1 shows the computation of $\boldsymbol{E}(\mathcal{C}_D)$ given different values of r. It is observed from Table 2.1 that the larger r is, the nearer

Table 2.1: Computation of $\boldsymbol{E}(\mathcal{C}_D)$

r	0.01	0.05	0.1	0.2	0.5
$\boldsymbol{E}(\mathcal{C}_D)$	0.48	0.57	0.62	0.66	0.78

$\boldsymbol{E}(\mathcal{C}_D)$ approaches 1, and the smaller r is, the nearer $\boldsymbol{E}(\mathcal{C}_D)$ approaches 0.5.

It is useful to know how the index $\mathcal{D}$ behaves with regard to randomness of data, hopefully it functions from 0 to 1 linearly. For the 2D example, suppose that $P\%$ denotes the percentage of the fine data points with the major class label. For instance, $P\%$=80% means that 80% of the data above $x_i(2) = x_i^2(1) - 0.5$ are randomly chosen and assigned with '1' while the remaining 20% data with '-1', and a similar assignment is done for the data below the parabola. Let $P\%$=50%, 60%, 70%, 80%, 90% and 100%, respectively. For each value, we apply the RGT and compute $\mathcal{D}$. The results are shown in Figure 2.9a with crosses, where the solid line is its linear fitting. Construct similarly the 3D case as follows. Take 80000 data points in a cube of [-1,1] uniformly and apply a RGT to group the cube into 8000 identical units. Specify the decision function as the paraboloid $x_i(3) = x_i^2(1) + x_i^2(2) - 0.6625$ and assign their labels to a given level of $P\%$. Calculate then $\mathcal{C}_D$, $\underline{\mathcal{C}_D}$ and $\mathcal{D}$ and show $\mathcal{D}$ vs $P\%$ in Figure 2.9b. It follows from Figure 2.9 that $\mathcal{D}$



(a) 2D.

(b) 3D.

Figure 2.9: Non-randomness indices and linear fitting curves.

functions almost linearly and gives a good gauge of non-randomness.

2.5.2 Reliability Index

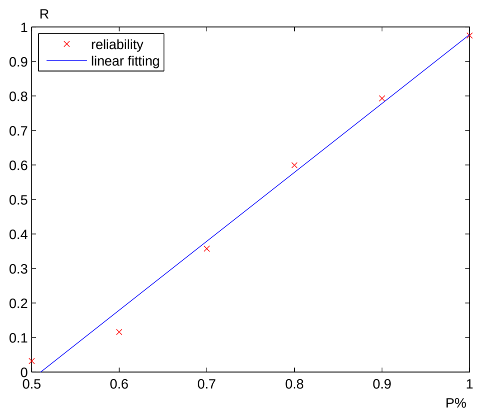
In practice, one needs to use some learning method to train a model. Thus, it requires evaluation of models, in addition to data assessment. If the fine and coarse models are trained from the purely random data, then their prediction consistency will be a low value. However if the models are trained by the deterministic data, then their prediction consistency will be almost one. Thus, the consistency index $\mathcal{C}_M$ on the two models is defined in (2.4). Like the data assessment, we also evaluate this index for the corresponding random case. Keep x_i unchanged, but assign their labels randomly. Apply the same RGT and learning method as before, the models will change due to changes of labels of data. We then evaluate the consistency index between the so-formed models: $\underline{\mathcal{C}_M}$. This number shall give the lower bound for this consistency index. Then we define the reliability index as

$$\mathcal{R} = \frac{\mathcal{C}_M - \underline{\mathcal{C}_M}}{1 - \underline{\mathcal{C}_M}} \times 100\%. \tag{2.10}$$

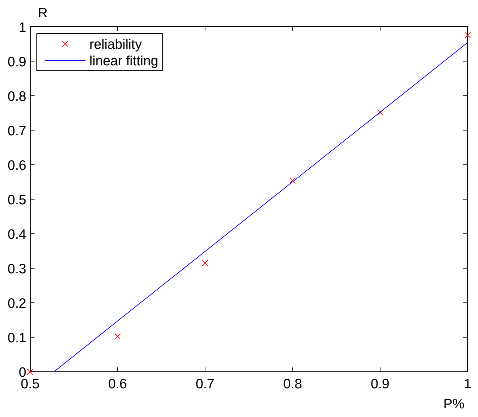
The smaller $\mathcal{R}$ is, the less reliable the models are. The larger $\mathcal{R}$, the more reliable the models are. For the 2D example, the deterministic case yields $\mathcal{R} = 99.08\%$ from (2.10) with $\mathcal{C}_M = 0.9970$ and $\underline{\mathcal{C}_M} = 0.6743$ from Section 2.3. The deterministic case gives a high value of $\mathcal{R}$, which shows a high reliability of the models. On the other hand, the pure random case gives $\mathcal{R}$ equal to 0.

It should be noted that $\mathcal{C}_M$ and $\underline{\mathcal{C}_M}$ have different values from $\mathcal{C}_D$ and $\underline{\mathcal{C}_D}$, respectively. This is because the formers are affected by a chosen learning method while the latter are not. Parameters of a classification tool may also be a factor and should be chosen to have minimal impact on the final assessment.

We also want to know how the index $\mathcal{R}$ behaves with regard to randomness of data, hopefully it functions from 0 to 1 linearly. For the 2D example, let $P\%$=50%, 60%, 70%, 80%, 90% and 100%. For each value, we apply the RGT and the learning method to compute its $\mathcal{R}$. The results are shown in Figure 2.10a with crosses, where the solid line is its linear fitting. The same procedure is performed on the 3D case above and $\mathcal{R}$ vs $P\%$ is shown in Figure 2.10b. It follows from Figure 2.10 that $\mathcal{R}$ functions almost linearly and makes a good sense of reliability.



(a) 2D.



(b) 3D.

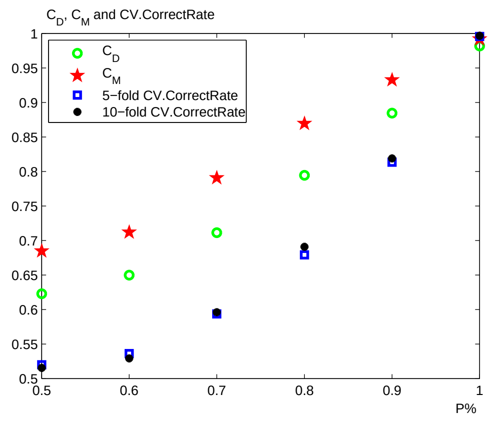
Figure 2.10: Reliability indices and linear fitting curves.

The proposed method is very different from the cross validation (CV) and will be now compared with the latter, since CV is the most popular model assessment methodology in the learning domain. For the 2D case, Table 2.2 shows the non-randomness index $\mathcal{D}$, the reliability index $\mathcal{R}$, the correct prediction rate of 5-fold CV and 10-fold CV, and their respective computational time for data sets. At the data level, the proposed method is

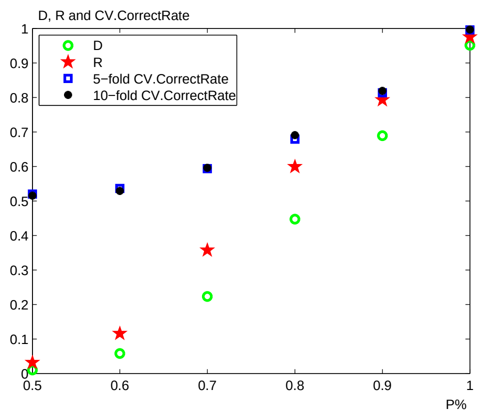
Table 2.2: Comparisons between the proposed method and CV

$P\%$	Proposed method				5-fold CV		10-fold CV	
	$\mathcal{D}$	Time	$\mathcal{R}$	Time	rate	Time	rate	Time
50%	1.05%	$0.08s$	3.19%	$85.42s$	0.5200	$265.33s$	0.5155	$623.02s$
60%	5.81%	$0.08s$	11.61%	$80.60s$	0.5363	$233.98s$	0.5293	$608.37s$
70%	22.35%	$0.08s$	35.74%	$70.70s$	0.5938	$194.91s$	0.5962	$518.59s$
80%	44.73%	$0.08s$	59.93%	$51.57s$	0.6793	$125.26s$	0.6910	$378.81s$
90%	68.93%	$0.08s$	79.28%	$14.18s$	0.8135	$31.25s$	0.8190	$88.70s$
100%	95.10%	$0.08s$	97.48%	$2.16s$	0.9955	$4.57s$	0.9965	$10.22s$

super fast to get $\mathcal{D}$ and is independent of any learning method. At the model level, our method only needs to train two models and takes short computational time whereas a K-fold CV needs to train K models. Therefore, our method is more efficient. Also, it is observed from Table 2.2 that $\mathcal{D}$ and $\mathcal{R}$, and the correct prediction rates of CV have a similar rising trend vs $P\%$. To see more clearly, the indices $\mathcal{C}_D$, $\mathcal{C}_M$, $\mathcal{D}$ and $\mathcal{R}$, and the correct prediction rates of 5-fold CV and 10-fold CV are shown in Figure 2.11. It follows



(a) $\mathcal{C}_D$, $\mathcal{C}_M$ and CV.CorrectRates.



(b) $\mathcal{D}$, $\mathcal{R}$ and CV.CorrectRates.

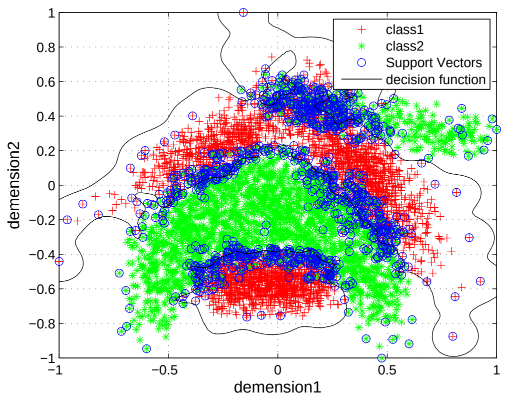
Figure 2.11: Indices and CV.CorrectRates vs $P\%$.

that the correct prediction rate of CV has the same tendency as $\mathcal{C}_D$, $\mathcal{C}_M$, $\mathcal{D}$ and $\mathcal{R}$. But note that unlike CV which splits data to folds with overlapping, our method produces new data from the original data, which avoids the repeated use of data.

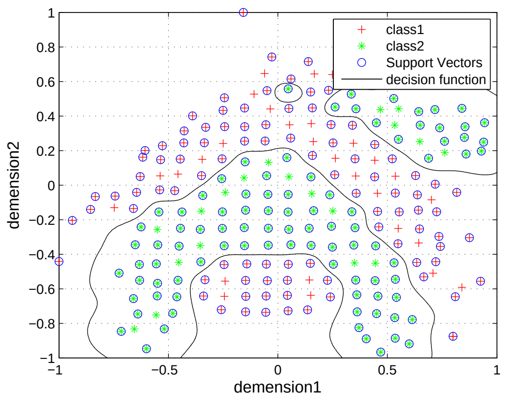
2.6 Simulation Examples

In this section, three practical examples are given to illustrate the proposed method.

Example 2.1. Consider the banana data [116]. This is a popular binary classification problem in machine learning. It has 5300 data points with two features. The fine data with normalization is shown in Figure 2.12a. RGT is done by geometrical grouping with squares of side of 0.1. We have 233 identical squares and the transformation ratio approximately equals to $r = 0.04$. The coarse data under the default RGT of Section 2.4 is obtained and shown in Figure 2.12b. It follows from Section 2.5 that $\mathcal{C}_D = 0.8983$,



(a) The fine model.



(b) The coarse model.

Figure 2.12: The banana example.

$\underline{\mathcal{C}_D} = 0.5764$ and $\mathcal{D} = 75.99\%$, indicating a high non-randomness of the data. With the SVM, the fine and coarse models are trained on two data sets, and exhibited in Figure 2.12a and Figure 2.12b with black curve, respectively. It follows that $\mathcal{C}_M = 0.9551$. With label randomization, one gets $\underline{\mathcal{C}_M} = 0.6713$. Thus, the reliability index for this example is calculated from (2.10) as $\mathcal{R} = 86.34\%$, which implies a reliable fine model. The SVM model given in [116] has the prediction accuracy of 88.44%.

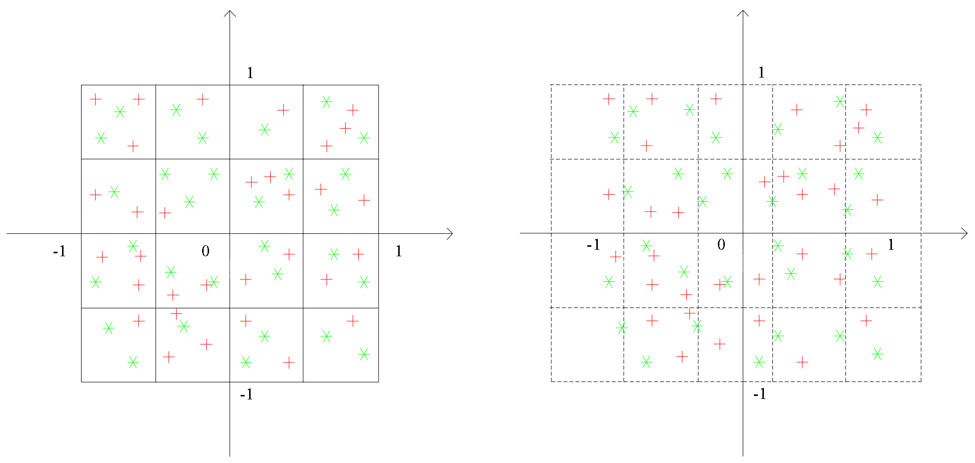
Example 2.2. Consider the "Astroparticle Physics" data [110]. It has 7089 data points with four features. We apply the k-means clustering method to group the fine data to clusters. With 709 clusters and the transformation ratio of $r = 0.1$, it follows from Section 2.5 that $\mathcal{C}_D = 0.9595$, $\underline{\mathcal{C}_D} = 0.6194$ and $\mathcal{D} = 89.36\%$, indicating a high non-randomness of the data. With the SVM, the fine and coarse models are trained on two data sets. It follows that $\mathcal{C}_M = 0.8965$. With label randomization, one gets $\underline{\mathcal{C}_M} = 0.6239$. Thus, the reliability index for this example is calculated from (2.10) as $\mathcal{R} = 72.48\%$, which implies that the reliability of the fine model is good. The raw SVM model in [110] has the prediction accuracy of 75.2%.

Example 2.3. Consider the "Cod RNA" data [117]. It has 59535 data points with eight features. The k-means clustering method is applied to do the RGT. With 5814 clusters and the transformation ratio of $r = 0.1$, it follows from Section 2.5 that $\mathcal{C}_D = 0.8806$, $\underline{\mathcal{C}_D} = 0.6098$ and $\mathcal{D} = 69.40\%$, indicating a mediocre non-randomness of the data. With the SVM, the fine and coarse models are trained on two data sets. It follows that $\mathcal{C}_M = 0.8471$. With label randomization, one gets $\underline{\mathcal{C}_M} = 0.6296$. Thus, the reliability index for this example is calculated from (2.10) as $\mathcal{R} = 58.72\%$, which implies the fine model is not quite reliable. We use the SVM parameters given in [117] to train a SVM model and obtain the prediction accuracy of 74.12%.

2.7 Variants of RGT

It should be noted that it is possible to generate multiple sets of coarse data under a single RGT with a fixed parameter, r, by making geometric perturbations of units/clusters. Figure 2.13 shows simple plots of the squares before and after shifting for illustration. This section is to study robustness with regards to such perturbations.

Look again at the 2D example of Section 2.3. Use one RGT with a fixed parameter, say, the squares with $r = 0.1$. With the first square exactly on the origin, the resulting coarse set with 400 units, $\hat{S}_1$, is obtained. If one shifts each square horizontally to the right by a perturbation, δ, say, $\delta = 0.02$, a new coarse set with 420 units, $\hat{S}_2$, is obtained. Continue this process and a total of five coarse sets, $\hat{S}_m$, $m = 1, 2, ..., 5$, can be found. If $\delta = 0.01$, one gets 10 coarse sets. Alternatively, one can shift squares vertically or with any particular angle with the x-axis. This creats unlimited possibilities. Thus, one can



(a) Original squares. (b) Shifted squares.

Figure 2.13: The squares before and after shifting.

always get as many coarse sets as desired. For illustration, take $\hat{S}_m$, $m = 1, 2, ..., 5$ for further study.

Consider first the deterministic data case. For each $\hat{S}_m$, $m = 1, 2, ..., 5$, as before, the label of each unit of a coarse set is determined by the majority rule and the labels of all the fine data points in the unit follows that unit label. Then this set of labels is compared with the original labels at each fine data point, and the consistency index $\mathcal{C}_D$ can be computed by (2.3). On the other hand, for the random data case, the above procedure yields the consistency index $\underline{\mathcal{C}_D}$. The results are shown in Table 2.3. It is seen that both $\mathcal{C}_D$ and

Table 2.3: $\mathcal{C}_D$ and $\underline{\mathcal{C}_D}$ for $\hat{S}_m$

	$\hat{S}_1$	$\hat{S}_2$	$\hat{S}_3$	$\hat{S}_4$	$\hat{S}_5$
Deterministic	0.9818	0.9805	0.9822	0.9810	0.9825
random	0.6282	0.6252	0.6262	0.6260	0.6228

$\underline{\mathcal{C}_D}$ are very stable with their averages being 0.9816 and 0.6257, respectively. Finally, the two cases are compared to calculate

$$\mathcal{D} = \frac{\mathcal{C}_D - \underline{\mathcal{C}_D}}{1 - \underline{\mathcal{C}_D}} \times 100\% = 95.08\%,$$

which is almost the same as the value calculated in Section 2.5. It is worth pointing out that due to averaging, more sets are utilized in evaluating the non-randomness index in this section than that in Section 2.5 which relies on a single set and the former statistic should be more powerful than the later [118].

It is obvious that the above analysis on the multiple coarse sets can be carried over to the model level following Section 2.5 and the details are omitted for brevity.

2.8 Conclusions

In this chapter, a new model assessment method has been proposed based on Renormalization Group. Its implementation rules and performance criteria are presented accordingly. By Renormalization Group, a coarse data set is created from the original fine data set and it is different from the latter in terms of their features and labels. This new data set enables us to assess data information without using any learning method. The proposed method does not require a separate validation data set. If the assessment shows that the data is random, the modeling or learning is useless and should not be performed at all. These characterize the proposed method and distinguish it from the existing methods in the domain. In addition, a model is trained on the coarse set, which is impossible with other methods which only divide the data to different sets with the same data points as the original ones. The predictions from the coarse model are compared with those of the model based on the fine data to evaluate model consistency and thus reliability of the learnt model. The proposed method is shown to work well with examples. It should be pointed out that the proposed method requires sufficient data. Furthermore, its theoretical properties and additional potential values are to be investigated.

In this chapter, we illustrate the proposed method for classification problem. The method can be also applied to regression problem with trivial modifications. In this case, simple averaging or a weighted average of responses, y_i, of fine points in a unit may be used to obtain the coarse response, $\hat{y}_j$, instead of the majority rule for classification, whereas the coarse feature can be determined in the same way as in classification. The comparison of fine and coarse data and models can be made with regard to, say, the standard squared errors for regression.

Chapter 3

Improved System Identification with Renormalization Group

3.1 Introduction

The previous chapter discusses about model assessment methods, which help with evaluating the informativeness and usefulness of a given model. A relatively good model may be chosen based on the assessment information. However, to get a good model, it is important to improve the stability and reliability of the model itself by employing some techniques. This chapter presents a technique helping to get good models in system identification, which is concerned with building mathematical models of dynamical systems from measured data [10]. In the field of system identification, the ordinary least squares (OLS) method [10] has since been the dominant algorithm for parameter estimation, while its estimate could be biased for a regression model with noise [15]. It is very challenging to analyze the properties of OLS estimate analytically. When the number of data points is unlimited, the OLS estimate will converge to its real value if the system is disturbed with white noise. For correlated noise, the OLS estimate could be biased and researchers have proposed some methods [21, 23] to avoid correlated noise and reduce the estimate bias. When the number of data points is unlimited, analyzing the bias becomes more difficult. Many papers [26, 27, 30, 33] in the literature only discuss the finite-sample bias of OLS estimate for very simple models.

In this chapter, we present an improved system identification method with Renormalization Group (RG). Technically, the proposed method groups the given data set into a RG data set. Performing the least squares algorithm, the proposed method obtains an estimate based on the given data and another estimate based on the RG data. Through comparisons of the two estimates, we find the proposed method could get a better estimate under certain conditions. The contributions of this chapter are summarized as follows.

- The proposed method forms a data set based on the system inputs and outputs, and produces a new data set from the given data set with the members of the former different from those of the latter, whereas none of the existing methods has a similar idea.
- We present theoretical analysis and simulation results for an academic model to illustrate the effectiveness of our method.

The rest of this chapter is organized as follows. Section 3.2 states our problem and motivation. Section 3.3 details the asymptotic analysis. Section 3.4 discusses the finite-sample case. Section 3.5 presents the simulation examples. Section 3.6 concludes the chapter.

3.2 Problem Statement and Motivation

In this section, we first describe the system and give some details of OLS estimation. Then we state the problem and give our motivation. In the end, we illustrate the idea of the proposed method by a simple example.

3.2.1 System Description

Consider a linear dynamic model

$$y_t = \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \ldots + \alpha_n y_{t-n} + \beta_0 u_t + \beta_1 u_{t-1} + \beta_2 u_{t-2} + \ldots + \beta_m u_{t-m} + \varepsilon_t, \quad (3.1)$$

where u_t is input, y_t is output and ε_t is noise. Let e_t be a white noise which has the properties:

- $E[e_t] = 0$,
- $E[e_t^2] = \sigma_e^2$,
- $E[e_t e_s] = 0$ for all $t \neq s$.

In (3.1), ε_t may have three forms:

$$\varepsilon_t = e_t, \quad (3.2)$$

$$\varepsilon_t = \lambda_1 \varepsilon_{t-1} + \lambda_2 \varepsilon_{t-2} + \ldots + \lambda_n \varepsilon_{t-n}, \quad (3.3)$$

$$\varepsilon_t = e_t + \lambda_1 e_{t-1} + \lambda_2 e_{t-2} + \ldots + \lambda_n e_{t-n}. \quad (3.4)$$

According to [10], we summarize some special cases of (3.1) in Table 3.1.

Table 3.1: Some models as special cases

α_i	β_i	ε_t	Name of model structure
$\alpha_i \neq 0$	$\beta_i = 0$	(3.2)	AR
$\alpha_i \neq 0$	$\beta_i = 0$	(3.3)	ARAR
$\alpha_i \neq 0$	$\beta_i = 0$	(3.4)	ARMA
$\alpha_i \neq 0$	$\beta_i \neq 0$	(3.2)	ARX
$\alpha_i \neq 0$	$\beta_i \neq 0$	(3.3)	ARARX
$\alpha_i \neq 0$	$\beta_i \neq 0$	(3.4)	ARMAX

3.2.2 OLS Estimation

We rewrite (3.1) as

$$y_t = \varphi_t^T \theta + \varepsilon_t, \tag{3.5}$$

where

$$\theta^T = \left[\begin{array}{c} \alpha_1, \ldots, \alpha_n, \beta_0, \ldots, \beta_m \end{array} \right], \tag{3.6}$$

and

$$\varphi_t^T = \left[\begin{array}{c} y_{t-1}, \ldots, y_{t-n}, u_t, \ldots, u_{t-m} \end{array} \right]. \tag{3.7}$$

The parameter vector θ is chosen to minimize the loss function

$$J = \frac{1}{2} \sum_{t=1}^{N} \left(y_t - \varphi_t^T \theta \right)^2. \tag{3.8}$$

The OLS estimate is given by

$$\hat{\theta} = \left(\Phi^T \Phi \right)^{-1} \left(\Phi^T Y \right), \tag{3.9}$$

where $Y = [y_1, \ldots, y_N]^T$, $\Phi = [\varphi_1, \ldots, \varphi_N]^T$ and $\mathcal{E} = [\varepsilon_1, \ldots, \varepsilon_N]^T$. The estimation error is given by

$$\begin{aligned} \Delta\theta &= \hat{\theta} - \theta \\ &= \left(\Phi^T\Phi\right)^{-1}\left(\Phi^T Y\right) - \theta \\ &= \left(\Phi^T\Phi\right)^{-1}\left(\Phi^T\left(\Phi\theta + \mathcal{E}\right)\right) - \theta \\ &= \left(\Phi^T\Phi\right)^{-1}\left(\Phi^T\mathcal{E}\right). \end{aligned} \tag{3.10}$$

The bias of the estimate is the expectation of $\Delta\theta$, that is

$$E\left[\Delta\theta\right] = E\left[\hat{\theta}\right] - \theta = E\left[\left(\Phi^T\Phi\right)^{-1}\left(\Phi^T\mathcal{E}\right)\right], \tag{3.11}$$

and the variance of the estimate is given by

$$cov\left[\hat{\theta}\right] = E\left[\left(\hat{\theta} - E\left[\hat{\theta}\right]\right)\left(\hat{\theta} - E\left[\hat{\theta}\right]\right)^T\right]. \tag{3.12}$$

When ε_t is white noise, we have [119]

$$E\left[\Delta\theta\right] = 0,$$

and

$$cov\left[\hat{\theta}\right] = \sigma_e^2\left(\Phi^T\Phi\right)^{-1}.$$

When ε_t is correlated noise, the OLS estimate could be biased.

With different weights w_t assigned to the measurements [10], the criterion (3.8) becomes

$$J = \frac{1}{2}\sum_{t=1}^{N} w_t\left(y_t - \varphi_t^T\theta\right)^2. \tag{3.13}$$

Then the resulting estimate is given by

$$\hat{\theta}_W = \left(\Phi^T W\Phi\right)^{-1}\left(\Phi^T WY\right), \tag{3.14}$$

where $W = \text{diag}(w_1, \ldots, w_N)$ is the weighting matrix. The estimation error is given by

$$\begin{aligned} \Delta\theta_W &= \hat{\theta}_W - \theta \\ &= \left(\Phi^T W \Phi\right)^{-1} \left(\Phi^T W \mathcal{E}\right). \end{aligned}$$

The bias of the estimate is given by

$$E\left[\Delta\theta_W\right] = E\left[\left(\Phi^T W \Phi\right)^{-1} \left(\Phi^T W \mathcal{E}\right)\right], \tag{3.15}$$

and the variance of the estimate has the same form with (3.12). When ε_t is white noise with $E(\mathcal{E}) = 0$ and $E(\mathcal{E}\mathcal{E}^T) = Q$, the estimate is unbiased. Furthermore, it has been shown [120] that when $W = Q^{-1}$, $\hat{\theta}$ is a best linear unbiased estimate (BLUE) and the variance of the estimate is given by

$$cov\left[\hat{\theta}_W\right] = \left(\Phi^T Q^{-1} \Phi\right)^{-1}. \tag{3.16}$$

When ε_t is correlated noise, the weighted least squares (WLS) estimate could be biased. The weighted least squares method is efficient but it depends on knowing the variance structure, which is seldom available in practice.

3.2.3 Idea of the Proposed Method

For system identification, one can always fit some model based on system inputs and outputs by OLS. However, when the system is equipped with correlated noise, the OLS estimate may be biased. This chapter studies the estimation error of OLS analytically and find some way to reduce it. We present an improved identification method using Renormalization Group (RG). RG was first proposed to study the critical phenomena in the quantum field in 1971 by Kenneth G. Wilson, who won the Nobel Prize for physics in

1982 [97], due to this great contribution. RG is widely used to analyze various physical problems [98, 99, 121]. Renormalization Group designs some Renormalization Group transformation (RGT) to relate macroscopic physics quantity to microscopic one and invokes “scale invariance” to solve the problem.

Considering the system (3.5), we first form a data $S_t^T = (y_t, \varphi_t^T), t = 1, 2, \dots, N$ and construct a data set $\boldsymbol{S} = \{S_1, S_2, \dots, S_N\}$. Then we perform a RGT on $\boldsymbol{S}$ to obtain a transformed data set $\boldsymbol{S_R} = \{S_{R1}, S_{R2}, \dots, S_{RK}\}$, with $S_{Rj}^T = (y_{Rj}, \varphi_{Rj}^T), j = 1, 2, \dots, K$. A RGT on $\boldsymbol{S}$ is to group a number of data points of $\boldsymbol{S}$ into one data point in $\boldsymbol{S_R}$ in a systematic way. There are different ways to do this grouping. One example is to group all the data of $\boldsymbol{S}$ using K-means clustering method [122]. We define $r = \frac{K}{N}$ as the transformation ratio. For each group, the coarse data may be determined by simple linear superposition of all the fine data that belong to this group. Then the weighted least squares method is performed on the coarse data. For easy reference, we call the data set, $\boldsymbol{S}$, as the fine data set and least squares estimate based on $\boldsymbol{S}$ as the OLS estimate. The data set $\boldsymbol{S_R}$ obtained from a RGT on the fine data set is called as the coarse data set and the resulting estimate as the Renormalization Group weighted least squares (RGWLS) estimate. In this chapter, through both theoretical analysis and simulation examples, we will show that the estimation error of RGWLS could be smaller than that of OLS estimate under certain conditions.

To see quickly the idea of the proposed method, we consider a simple model

$$y_t = 0.8y_{t-1} + 0.5u_t + \gamma(e_t - 0.8e_{t-1}),$$

where $y_0 = 0$, u_t is a unit step signal, e_t is a white noise. Let $E\left[e_t^2\right] = 1$ and $\gamma = 0.0625$.

The model is simulated over 30 steps. We have

$$
\begin{aligned}
\theta^T &= [0.8, 0.5], \\
\varphi_t^T &= [y_{t-1}, u_t], \quad t = 1, 2, \ldots, 30, \\
Y &= [y_1, \ldots, y_{30}]^T, \\
\Phi &= [\varphi_1, \ldots, \varphi_{30}]^T.
\end{aligned}
$$

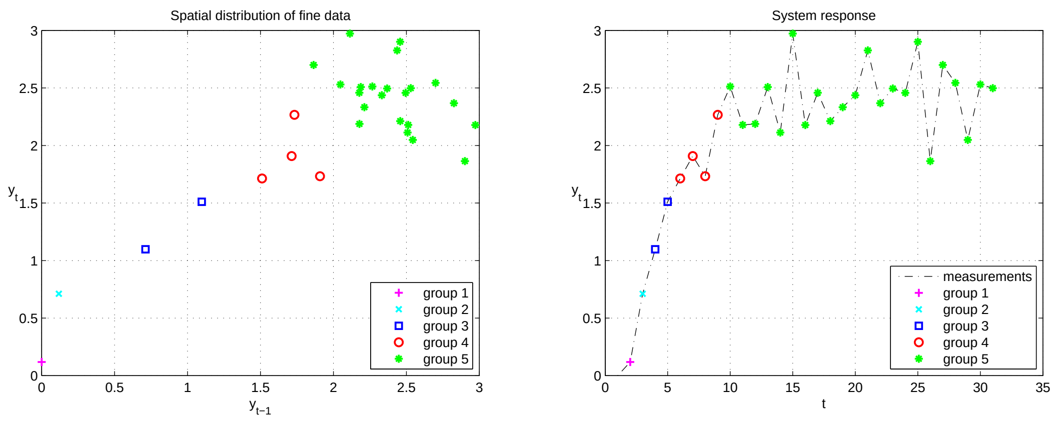
Then the OLS estimate is obtained based on (3.9). Next we carry out the RG method. We first form the fine data as $S_t^T = (y_t, \varphi_t) = (y_t, y_{t-1}, u_t) = (y_t, y_{t-1}, 1)$ and the fine data set as $\boldsymbol{S} = \{S_1, S_2, \ldots, S_{30}\}$. Then a RGT is performed on the fine data set $\boldsymbol{S}$ to get the coarse data set $\boldsymbol{S_R}$. In this case, it is done by K-means clustering method based on Euclidean distance measure with $K = 5$, and for each group, the coarse data is obtained by the linear superposition of all the fine data in this group. We have

$$
\begin{aligned}
S_{R1} &= S_1, \\
S_{R2} &= S_2, \\
S_{R3} &= S_3 + S_4, \\
S_{R4} &= S_5 + \ldots + S_8, \\
S_{R5} &= S_9 + \ldots + S_{30}.
\end{aligned}
$$

It follows that

$$
\begin{aligned}
&y_{R1} = y_1, && \varphi_{R1} = \varphi_1, && \varepsilon_{R1} = \varepsilon_1, \\
&y_{R2} = y_2, && \varphi_{R2} = \varphi_2, && \varepsilon_{R2} = \varepsilon_2, \\
&y_{R3} = y_3 + y_4, && \varphi_{R3} = \varphi_3 + \varphi_4, && \varepsilon_{R3} = \varepsilon_3 + \varepsilon_4, \\
&y_{R4} = y_5 + \ldots + y_8, && \varphi_{R4} = \varphi_5 + \ldots + \varphi_8, && \varepsilon_{R4} = \varepsilon_5 + \ldots + \varepsilon_8, \\
&y_{R5} = y_9 + \ldots + y_{30}, && \varphi_{R5} = \varphi_9 + \ldots + \varphi_{30}, && \varepsilon_{R5} = \varepsilon_9 + \ldots + \varepsilon_{30}.
\end{aligned}
$$

The spatial distribution of different groups of the fine data is shown in Figure 3.1a and the system response is shown in Figure 3.1b. It is obvious that y_{Rj}, φ_{Rj} and ε_{Rj} satisfy



(a) Spatial distribution of fine data

(b) System response

Figure 3.1: Data grouping

$$y_{Rj} = \varphi_{Rj}^T \theta + \varepsilon_{Rj}, \quad j = 1, 2, 3, 4, 5.$$

Choosing the weighting matrix as $W = \mathrm{diag}(1, 1, \frac{1}{4}, \frac{1}{16}, \frac{1}{484})$, the RGWLS estimate is obtained based on (3.14). Employing the Monte Carlo method, the simulation is repeated 100 times. Then we get $E\left(\|\Delta\theta\|_2\right) = 0.38$ and $E\left(\|\Delta\theta_R\|_2\right) = 0.19$, where $\|\cdot\|_2$ denotes the l^2-norm. Therefore, the RGWLS estimate is better than the OLS estimate.

3.3 Asymptotic Analysis

It is known that the bias of the OLS estimate is related to the number of data N if N is finite. If the system is disturbed by white noise, it is possible to obtain explicit formula for the bias in very small samples [26–30]. However, if the system is disturbed by colored noise, it has been proved to be difficult to investigate finite-sample bias analytically [31].

In the absence of such results, researchers [31–33] perform Monte Carlo experiments to provide alternative sources of information. In this section, we discuss the properties of the OLS and RGWLS estimates as $N \to \infty$, analytically.

With (3.10) being multiplied and divided by N, we have

$$\lim_{N\to\infty} \Delta\theta = \lim_{N\to\infty} \left[\left(\frac{\Phi^T \Phi}{N} \right)^{-1} \left(\frac{\Phi^T \mathcal{E}}{N} \right) \right]. \tag{3.17}$$

Property 1 [123]: If the limits $\lim\limits_{N\to\infty} H(N)$ and $\lim\limits_{N\to\infty} G(N)$ both exist, and $\lim\limits_{N\to\infty} G(N)$ is nonsingular, then

$$\lim_{N\to\infty} \left[G^{-1}(N) H(N) \right] = \left[\lim_{N\to\infty} G(N) \right]^{-1} \left[\lim_{N\to\infty} H(N) \right]. \tag{3.18}$$

With the above property, (3.17) is equivalent to

$$\begin{aligned} \lim_{N\to\infty} \Delta\theta &= \left[\lim_{N\to\infty} \left(\frac{\Phi^T \Phi}{N} \right) \right]^{-1} \left[\lim_{N\to\infty} \left(\frac{\Phi^T \mathcal{E}}{N} \right) \right] \\ &= \left[\lim_{N\to\infty} \frac{\sum\limits_{t=1}^{N} \left(\varphi_t \varphi_t^T \right)}{N} \right]^{-1} \left[\lim_{N\to\infty} \frac{\sum\limits_{t=1}^{N} \left(\varphi_t \varepsilon_t \right)}{N} \right]. \end{aligned} \tag{3.19}$$

When the system (3.1) is asymptotically stable with ε_t a stationary stochastic process that is independent of the input signal, Söderström [23, 124, 125] show that the sums $\frac{\sum_{t=1}^{N} \left(\varphi_t \varphi_t^T \right)}{N}$ and $\frac{\sum_{t=1}^{N} \left(\varphi_t \varepsilon_t \right)}{N}$ tend to the corresponding expected values with probability one as the number of data points, N, tends to infinity. Then (3.19) becomes

$$\lim_{N\to\infty} \Delta\theta = \left[E\left(\varphi_t \varphi_t^T \right) \right]^{-1} \left[E\left(\varphi_t \varepsilon_t \right) \right]. \tag{3.20}$$

As can be seen from (3.20), when $E\left(\varphi_t \varphi_t^T \right)$ is non-singular and $E\left(\varphi_t \varepsilon_t \right) = 0$, $\hat{\theta}$ will converge to θ. Otherwise, the OLS estimate could be biased.

The instrumental variable (IV) method, which is an efficient method to reduce the bias of OLS estimate as $N \to \infty$, was introduced by Reiersøl [126]. This method has been popular for quite a long period. The basic IV method [23] for estimating θ in (3.5) is given by

$$\lim_{N\to\infty} \hat{\theta} = \left[\lim_{N\to\infty} \frac{1}{N}\sum_{t=1}^{N} Z_t\varphi_t^T\right]^{-1} \left[\lim_{N\to\infty} \frac{1}{N}\sum_{t=1}^{N} Z_t y_t\right], \tag{3.21}$$

where the elements in matrix Z_t are called instruments or instrumental variables. They can be chosen in different ways. A calculation for the IV estimate (3.21) gives

$$\begin{aligned} \lim_{N\to\infty} \Delta\theta &= \left[\lim_{N\to\infty} \frac{1}{N}\sum_{t=1}^{N} Z_t\varphi_t^T\right]^{-1} \left[\lim_{N\to\infty} \frac{1}{N}\sum_{t=1}^{N} Z_t\varepsilon_t\right] \\ &= \left[E\left(Z_t\varphi_t^T\right)\right]^{-1}\left[E\left(Z_t\varepsilon_t^T\right)\right]. \end{aligned} \tag{3.22}$$

In order to let $\hat{\theta}$ converge to θ, it is necessary that $E\left(Z_t\varphi_t^T\right)$ is nonsingular and $E\left(Z_t\varepsilon_t\right) = 0$. As to the accuracy properties of the IV estimates, it is shown that $\sqrt{N}\left(\hat{\theta}-\theta\right)$ converges in distribution to a gaussian distribution, which is denoted by

$$\sqrt{N}\left(\hat{\theta}-\theta\right) \sim N\left(0, P_{IV}\right).$$

Explicit expressions for the covariance matrix P_{IV} are given in [23, 127].

3.3.1 The Asymptotic Properties of OLS Estimate

It can be seen from (3.20) that when ε_t is white noise, the OLS estimate will converge to its real value. However, when ε_t is colored noise, the OLS estimate could be biased. It is known that $\Delta\theta$ is related to model structure, input signal and noise properties. But these factors cannot be reflected in (3.20) explicitly. Therefore, for formulation of the estimation error and easy illustration of the proposed method, we consider a simple ARMA(1,1) model with zero initial conditions, which is given by

$$y_t = \alpha_1 y_{t-1} + \varepsilon_t \quad |\alpha_1| < 1, \tag{3.23}$$

where $\varepsilon_t = e_t + \lambda_1 e_{t-1}$. According to (3.6) and (3.7), we have $\theta = \alpha_1$ and $\varphi_t = y_{t-1}$. It follows from (3.20) that

$$\lim_{N\to\infty} \Delta\alpha_1 = \left[E\left(y_{t-1}^2\right)\right]^{-1}\left[E\left(y_{t-1}\varepsilon_t\right)\right]. \tag{3.24}$$

Introducing the lag operator $Lz_t = z_{t-1}$ into (3.23), we obtain

$$y_t = \frac{\varepsilon_t}{(1-\alpha_1 L)} = \sum_{p=0}^{\infty} \alpha_1^p \varepsilon_{t-p}. \tag{3.25}$$

Then

$$\begin{aligned}
E(y_{t-1}^2) &= E\left[\left(\sum_{p=0}^{\infty} \alpha_1^p \varepsilon_{t-1-p}\right)^2\right] \\
&= E\left(\sum_p \sum_q \alpha_1^p \alpha_1^q \varepsilon_{t-1-p}\varepsilon_{t-1-q}\right) \\
&= \sum_p \sum_q \alpha_1^p \alpha_1^q E(\varepsilon_{t-1-p}\varepsilon_{t-1-q}) \\
&= \sum_p \sum_q \alpha_1^p \alpha_1^q E\left[(e_{t-1-p} + \lambda_1 e_{t-2-p})(e_{t-1-q} + \lambda_1 e_{t-2-q})\right] \\
&= \sum_{p=q}\sum \alpha_1^p \alpha_1^q (1+\lambda_1^2)\sigma_e^2 + 2\sum_{q=p+1}\sum \alpha_1^p \alpha_1^q \lambda_1 \sigma_e^2 \\
&= \sum_{p=0}^{\infty} \alpha_1^{2p}(1+\lambda_1^2)\sigma_e^2 + 2\sum_{p=0}^{\infty} \alpha_1^{2p+1}\lambda_1\sigma_e^2 \\
&= \frac{(1+\lambda_1^2)}{1-\alpha_1^2}\sigma_e^2 + \frac{2\lambda_1\alpha_1}{1-\alpha_1^2}\sigma_e^2 \\
&= \frac{(1+\lambda_1^2+2\lambda_1\alpha_1)}{1-\alpha_1^2}\sigma_e^2.
\end{aligned} \tag{3.26}$$

We also have

$$\begin{aligned}
E(y_{t-1}\varepsilon_t) &= E\left(\sum_{p=0}^{\infty} \alpha_1^p \varepsilon_{t-1-p}\varepsilon_t\right) \\
&= \sum_{p=0}^{\infty} \alpha_1^p E[(e_{t-1-p} + \lambda_1 e_{t-2-p})(e_t + \lambda_1 e_{t-1})],
\end{aligned} \tag{3.27}$$

where

$$E[(e_{t-1-p} + \lambda_1 e_{t-2-p})(e_t + \lambda_1 e_{t-1})] = \begin{cases} \lambda_1 \sigma_e^2, & \text{for } p = 0; \\ 0, & \text{for } p > 0. \end{cases}$$

Hence

$$E(y_{t-1}\varepsilon_t) = \lambda_1\sigma_e^2. \tag{3.28}$$

Substituting (3.26) and (3.28) into (3.24) gives

$$\begin{aligned} \lim_{N\to\infty} \Delta\alpha_1 &= \left[\frac{(1+\lambda_1^2+2\lambda_1\alpha_1)}{1-\alpha_1^2}\sigma_e^2\right]^{-1} (\lambda_1\sigma_e^2) \\ &= \frac{\lambda_1(1-\alpha_1^2)}{1+\lambda_1^2+2\lambda_1\alpha_1}. \end{aligned} \tag{3.29}$$

In order to obtain a neater result, we consider a special case that $\lambda_1 = -\alpha_1$. Then (3.29) becomes

$$\lim_{N\to\infty} \Delta\alpha_1 = -\alpha_1. \tag{3.30}$$

3.3.2 The Asymptotic Properties of RGWLS Estimate

RG is performed on the fine data set to obtain the coarse data set. The RGT groups all fine data based on the K-means clustering method and takes linear superposition of all the fine data as the coarse data in each group. Introduce the notations

$$\begin{aligned} y_{Rj} &= \sum_{t\in j} y_t, \\ \varphi_{Rj} &= \sum_{t\in j} \varphi_t, \\ \varepsilon_{Rj} &= \sum_{t\in j} \varepsilon_t, \quad j = 1,\dots,K, \end{aligned} \tag{3.31}$$

where y_{Rj}, φ_{Rj} and ε_{Rj} satisfy

$$y_{Rj} = \varphi_{Rj}^T\theta + \varepsilon_{Rj}. \tag{3.32}$$

The parameter θ is chosen to minimize the loss function

$$J_R = \frac{1}{2}\sum_{j=1}^{K}\left[w_j\left(y_{Rj} - \varphi_{Rj}^T\theta_R\right)^2\right], \tag{3.33}$$

where w_j is the chosen weight. The RGWLS estimate is given by

$$\hat{\theta}_R = \left(\Phi_R^T W \Phi_R\right)^{-1} \left(\Phi_R^T W Y_R\right), \tag{3.34}$$

where $Y_R = [y_{R1} \ldots, y_{RK}]^T$, $\Phi_R = [\varphi_{R1}, \ldots, \varphi_{RK}]^T$, $\mathcal{E}_R = [\varepsilon_{R1}, \ldots, \varepsilon_{RK}]^T$, and $W = \operatorname{diag}(w_1, \ldots, w_K)$. The estimation error of RGWLS is given by

$$\begin{aligned} \Delta\theta_R &= \hat{\theta}_R - \theta \\ &= \left(\Phi_R^T W \Phi_R\right)^{-1} \left(\Phi_R^T W Y_R\right) - \theta \\ &= \left(\Phi_R^T W \Phi_R\right)^{-1} \left(\Phi_R^T W \left(\Phi_R \theta + \mathcal{E}_R\right)\right) - \theta \\ &= \left(\Phi_R^T W \Phi_R\right)^{-1} \left(\Phi_R^T W \mathcal{E}_R\right). \end{aligned} \tag{3.35}$$

Next we analyze the properties of $\lim\limits_{N\to\infty} \Delta\theta_R$. Let n_j be the number of fine data in the jth group so that

$$n_1 + \ldots + n_K = N. \tag{3.36}$$

Suppose n_j tends to infinity. Similarly with (3.17) and (3.19), we have

$$\begin{aligned} \lim_{N\to\infty} \Delta\theta_R &= \lim_{N\to\infty} \left[\left(\Phi_R^T W \Phi_R\right)^{-1} \left(\Phi_R^T W \mathcal{E}_R\right)\right] \\ &= \left(\lim_{N\to\infty} \Phi_R^T W \Phi_R\right)^{-1} \left(\lim_{N\to\infty} \Phi_R^T W \mathcal{E}_R\right) \\ &= \left[\lim_{N\to\infty} \sum_{j=1}^{K} \left(w_j \varphi_{Rj} \varphi_{Rj}^T\right)\right]^{-1} \left[\lim_{N\to\infty} \sum_{j=1}^{K} \left(w_j \varphi_{Rj} \varepsilon_{Rj}\right)\right] \\ &= \left[\sum_{j=1}^{K} \lim_{n_j\to\infty} \left(w_j \varphi_{Rj} \varphi_{Rj}^T\right)\right]^{-1} \left[\sum_{j=1}^{K} \lim_{n_j\to\infty} \left(w_j \varphi_{Rj} \varepsilon_{Rj}\right)\right]. \end{aligned} \tag{3.37}$$

Let $w_j = 1/n_j^2$. Substituting (3.31) into (3.37) gives

$$\begin{aligned} \lim_{N\to\infty} \Delta\theta_R &= \left[\sum_{j=1}^{K} \lim_{n_j\to\infty} \left(\frac{1}{n_j^2} (\sum_{t\in j} \varphi_t)(\sum_{t\in j} \varphi_t^T)\right)\right]^{-1} \left[\sum_{j=1}^{K} \lim_{n_j\to\infty} \left(\frac{1}{n_j^2} (\sum_{t\in j} \varphi_t)(\sum_{t\in j} \varepsilon_t^T)\right)\right] \\ &= \left[\sum_{j=1}^{K} \lim_{n_j\to\infty} \left(\frac{1}{n_j^2} (\sum_{t\in j} \varphi_t)(\sum_{s\in j} \varphi_s^T)\right)\right]^{-1} \left[\sum_{j=1}^{K} \lim_{n_j\to\infty} \left(\frac{1}{n_j^2} (\sum_{t\in j} \varphi_t)(\sum_{s\in j} \varepsilon_s^T)\right)\right]. \end{aligned} \tag{3.38}$$

As $n_j \to \infty$, suppose $\sum\limits_{j=1}^{K} \lim\limits_{n_j\to\infty} \left(\frac{1}{n_j^2} (\sum\limits_{t\in j} \varphi_t)(\sum\limits_{s\in j} \varphi_s^T) \right)$ and $\sum\limits_{j=1}^{K} \lim\limits_{n_j\to\infty} \left(\frac{1}{n_j^2} (\sum\limits_{t\in j} \varphi_t)(\sum\limits_{s\in j} \varepsilon_s^T) \right)$ converge to $E\left(\varphi_t \varphi_s^T\right)|_{t,s\in j}$ and $E\left(\varphi_t \varepsilon_s\right)|_{t,s\in j}$, respectively. Then (3.38) is rewritten as

$$\lim_{N\to\infty} \Delta\theta_R = \left[\sum_{j=1}^{K} E\left(\varphi_t \varphi_s^T\right)|_{t,s\in j}\right]^{-1} \left[\sum_{j=1}^{K} E\left(\varphi_t \varepsilon_s\right)|_{t,s\in j}\right]. \tag{3.39}$$

Compared with (3.20), the estimation error of RGWLS (3.39) is more complicated to be analyzed because of the data grouping.

In order to make a quantitative description of $\lim\limits_{N\to\infty} \Delta\theta_R$, we still consider the simple model (3.23). It follows from (3.39) that

$$\lim_{N\to\infty} \Delta\alpha_{1R} = \left[\sum_{j=1}^{K} E\left(y_{t-1} y_{s-1}\right)|_{t,s\in j}\right]^{-1} \left[\sum_{j=1}^{K} E\left(y_{t-1} \varepsilon_s\right)|_{t,s\in j}\right]. \tag{3.40}$$

In (3.40),

$$E\left(y_{t-1} y_{s-1}\right)|_{t,s\in j} = E\left[(\sum_{p=0}^{\infty} \alpha_1^p \varepsilon_{t-p-1})(\sum_{q=0}^{\infty} \alpha_1^q \varepsilon_{s-q-1})\right]|_{t,s\in j} \tag{3.41}$$

When $t = s$,

$$\begin{aligned} E\left(y_{t-1} y_{s-1}\right)|_{t,s\in j} &= \sum_{p=0}^{\infty}\sum_{q=0}^{\infty} \alpha_1^p \alpha_1^q E\left(\varepsilon_{t-p-1}\varepsilon_{t-q-1}\right)|_{t\in j} \\ &= \sum_p \sum_q \alpha_1^p \alpha_1^q E\left[\left(e_{t-p-1} + \lambda_1 e_{t-p-2}\right)\left(e_{t1-q-1} + \lambda_1 e_{t1-q-2}\right)\right]|_{t\in j} \\ &= (\sum_{p=q}\sum \alpha_1^{2p}\left(1+\lambda_1^2\right)\sigma_e^2 + \sum_{p=q+1}\sum \alpha_1^{2q+1}\lambda_1\sigma_e^2 + \sum_{q=p+1}\sum \alpha_1^{2p+1}\lambda_1\sigma_e^2)|_{t\in j} \\ &= \frac{1+\lambda_1^2+2\lambda_1\alpha_1}{1-\alpha_1^2}\sigma_e^2|_{t\in j}\ ; \end{aligned} \tag{3.42}$$

when $t < s$,

$$\begin{aligned}
E\left(y_{t-1}y_{s-1}\right)|_{t,s\in j} &= \sum_{p=0}^{\infty}\sum_{q=0}^{\infty}\alpha_1^p\alpha_1^q E\left(\varepsilon_{t-p-1}\varepsilon_{s-q-1}\right)|_{t,s\in j} \\
&= \sum_p\sum_q \alpha_1^p\alpha_1^q E\left[\left(e_{t-p-1}+\lambda_1 e_{t-p-2}\right)\left(e_{s-q-1}+\lambda_1 e_{s-q-2}\right)\right]|_{t,s\in j} \\
&= \Big(\sum\sum_{q=s-t+p}\alpha_1^{2p+s-t}\left(1+\lambda_1^2\right)\sigma_e^2 + \sum\sum_{q=s-t+p-1}\alpha_1^{2p+s-t-1}\lambda_1\sigma_e^2 \\
&\quad + \sum\sum_{q=s-t+p+1}\alpha_1^{2p+s-t+1}\lambda_1\sigma_e^2\Big)|_{t,s\in j} \\
&= \frac{\alpha_1^{s-t}\left(1+\lambda_1^2\right)+\alpha_1^{s-t-1}\lambda_1+\alpha_1^{s-t+1}\lambda_1}{1-\alpha_1^2}\sigma_e^2|_{t,s\in j}\ .
\end{aligned} \tag{3.43}$$

when $t > s$,

$$\begin{aligned}
E\left(y_{t-1}y_{s-1}\right)|_{t,s\in j} &= \sum_{p=0}^{\infty}\sum_{q=0}^{\infty}\alpha_1^p\alpha_1^q E\left(\varepsilon_{t-p-1}\varepsilon_{s-q-1}\right)|_{t,s\in j} \\
&= \sum_p\sum_q \alpha_1^p\alpha_1^q E\left[\left(e_{t-p-1}+\lambda_1 e_{t-p-2}\right)\left(e_{s-q-1}+\lambda_1 e_{s-q-2}\right)\right]|_{t,s\in j} \\
&= \Big(\sum\sum_{p=t-s+q}\alpha_1^{2q+t-s}\left(1+\lambda_1^2\right)\sigma_e^2 + \sum\sum_{p=t-s+q-1}\alpha_1^{2q+t-s-1}\lambda_1\sigma_e^2 \\
&\quad + \sum\sum_{p=t-s+q+1}\alpha_1^{2q+t-s+1}\lambda_1\sigma_e^2\Big)|_{t,s\in j} \\
&= \frac{\alpha_1^{t-s}\left(1+\lambda_1^2\right)+\alpha_1^{t-s-1}\lambda_1+\alpha_1^{t-s+1}\lambda_1}{1-\alpha_1^2}\sigma_e^2|_{t,s\in j}\ .
\end{aligned} \tag{3.44}$$

Substituting $\lambda_1 = -\alpha_1$ into (3.42), (3.43) and (3.44) gives

$$E\left(y_{t-1}y_{s-1}\right)|_{t,s\in j} = \begin{cases} \sigma_e^2, & \text{for } t = s; \\ 0, & \text{for } t \neq s. \end{cases} \tag{3.45}$$

Hence

$$\sum_{j=1}^{K} E\left(y_{t-1}y_{s-1}\right)|_{t,s\in j} = K\sigma_e^2. \tag{3.46}$$

From (3.40), we also have

$$E\left(y_{t-1}\varepsilon_s\right)|_{t,s\in j} = E\left(\sum_{p=0}^{\infty}\alpha_1^p\varepsilon_{t-p-1}\varepsilon_s\right)|_{t,s\in j}\ . \tag{3.47}$$

When $t = s$,

$$\begin{aligned} E\left(y_{t-1}\varepsilon_s\right)|_{t,s\in j} &= \sum_{p=0}^{\infty} \alpha_1^p E\left(e_{t-p-1} + \lambda_1 e_{t-p-2}\right)\left(e_t + \lambda_1 e_{t-1}\right)|_{t\in j} \\ &= \lambda_1 \sigma_e^2 \,|_{t\in j}\ ; \end{aligned} \tag{3.48}$$

when $t < s$,

$$\begin{aligned} E\left(y_{t-1}\varepsilon_s\right)|_{t,s\in j} &= \sum_{p=0}^{\infty} \alpha_1^p E\left(e_{t-p-1} + \lambda_1 e_{s-p-2}\right)\left(e_t + \lambda_1 e_{s-1}\right)|_{t,s\in j} \\ &= 0\,|_{t,s\in j}\ ; \end{aligned} \tag{3.49}$$

when $t > s$, let $k \geq 2$ be a positive integer, then

$$\begin{aligned} E\left(y_{t-1}\varepsilon_s\right)|_{t,s\in j} &= \sum_{p=0}^{\infty} \alpha_1^p E\left(e_{t-p-1} + \lambda_1 e_{t-p-2}\right)\left(e_s + \lambda_1 e_{s-1}\right)|_{t,s\in j} \\ &= \Bigg\{ \sum_{\substack{p=0 \\ t=s+1}} \left(1+\lambda_1^2\right)\sigma_e^2 + \sum_{\substack{p=1 \\ t=s+1}} \alpha_1\lambda_1\sigma_e^2 \sum_{j=1}^{K} E\left(y_{t-1}\varepsilon_s\right) \\ &\quad + \sum_{\substack{p=k-1 \\ t=s+k}} \alpha_1^{k-1}\left(1+\lambda_1^2\right)\sigma_e^2 + \sum_{\substack{p=k \\ t=s+k}} \alpha_1^k\lambda_1\sigma_e^2 \\ &\quad + \sum_{\substack{p=k-2 \\ t=s+k}} \alpha_1^{k-2}\lambda_1\sigma_e^2 \sum_{j=1}^{K} E\left(y_{t-1}\varepsilon_s\right) \Bigg\} |_{t,s\in j}\ . \end{aligned} \tag{3.50}$$

Substituting $\lambda_1 = -\alpha_1$ into (3.48), (3.49) and (3.50) gives

$$E\left(y_{t-1}\varepsilon_s\right)|_{t,s\in j} = \begin{cases} -\alpha_1\sigma_e^2, & \text{for } t = s; \\ 0, & \text{for } t < s; \\ \sigma_e^2, & \text{for } t = s+1. \end{cases} \tag{3.51}$$

Denoting $f_j = d_j/(n_j - 1)$ where d_j is the number of S_t such that S_t and S_{t+1} belong to the jth group, we have

$$E\left(y_{t-1}\varepsilon_s\right)|_{t,s\in j} = -\alpha_1\sigma_e^2 + f_j\sigma_e^2. \tag{3.52}$$

Hence

$$\sum_{j=1}^{K} E\left(\varphi_t\varepsilon_s\right)|_{t,s\in j} = -K\alpha_1\sigma_e^2 + \sum_{j=1}^{K} f_j\sigma_e^2. \tag{3.53}$$

Substituting (3.46) and (3.53) into (3.40) gives

$$\lim_{N\to\infty} \Delta\alpha_{1R} = -\alpha_1 + \frac{\sum\limits_{j=1}^{K} f_j}{K}. \tag{3.54}$$

Dividing (3.54) by (3.30), we have

$$\frac{\lim\limits_{N\to\infty} \Delta\alpha_{1R}}{\lim\limits_{N\to\infty} \Delta\alpha_1} = \frac{-\alpha_1 + \frac{\sum_{j=1}^{K} f_j}{K}}{-\alpha_1}. \tag{3.55}$$

With $|\alpha_1| < 1$, it is easy to verify

$$\left| \frac{\lim\limits_{N\to\infty} \Delta\alpha_{1R}}{\lim\limits_{N\to\infty} \Delta\alpha_1} \right| < 1, \tag{3.56}$$

if and only if

$$\begin{cases} \frac{\sum\limits_{j=1}^{K} f_j}{2K} < \alpha_1 < 1, \\ 0 < \frac{\sum\limits_{j=1}^{K} f_j}{K} < 1. \end{cases} \tag{3.57}$$

When $\sum\limits_{j=1}^{K} f_j/K$ approaches to α_1, $\Delta\alpha_{1R}$ tends to zero and the RGWLS estimate will have greatest improvement. In addition, when $K = N$, then $n_j = 1$ and $f_j = 0$. It is easy to see that (3.54) becomes

$$\lim_{N\to\infty} \Delta\alpha_{1R} = -\alpha_1, \tag{3.58}$$

which is the same as (3.30), implying that RGWLS is reduced to OLS.

3.4 Finite-Sample Analysis

In this section, we study the properties of the OLS and RGWLS estimates when N is finite. We discuss the tradeoff between signal to noise ratio (SNR) and N, and compare the RGWLS method with the generalized least squares (GLS) method. If the system is disturbed with colored noise, it is difficult to develop the finite-sample properties of the least squares estimate analytically [31]. We provide examples for illustration in Section 3.5.

3.4.1 Tradeoff between SNR and N

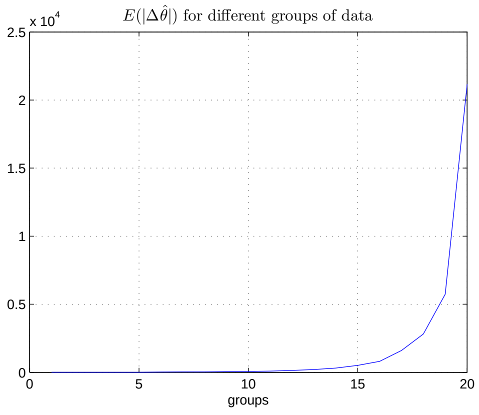
It is known that the finite-sample properties of OLS estimate is related to SNR and the number of data points N. Generally, a higher SNR and a larger N will lead to a better OLS estimate. There might be some cases that SNR is high for part of total data, and when N increases, SNR will decrease. Hence, for such cases, there should be a tradeoff between SNR and N. The following example is given for illustration.

Consider a model

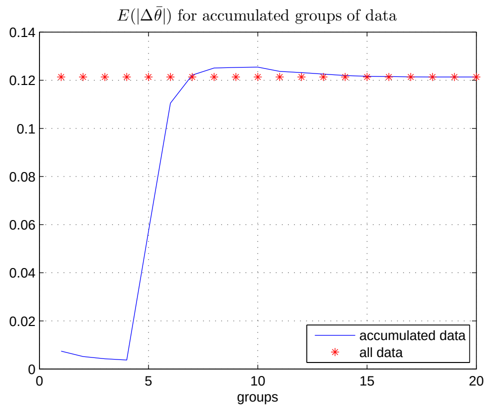
$$y_t = 2u_t + \gamma e_t, \tag{3.59}$$

where $\theta = 2$, $u_t = 0.01t$, e_t is white noise with and γ is a chosen parameter that is used to adjust the noise level, say γ is used to adjust SNR. Suppose the system is simulated for 1000 steps with $\gamma = 100$ when $t \leq 500$ and $\gamma = 1$ when $t > 500$. We study effects of SNR on estimation using OLS. The K-means clustering method is used to obtain 20 groups of data based on SNR levels so that these 20 groups of data have different levels of SNR. We rank them from the $1st$ to $20th$ group with the $1st$ having highest SNR and the $20th$ the lowest SNR. Then, we do two kinds of simulations. Firstly, we perform OLS on each group of data and obtain $\hat{\theta}_1, \ldots, \hat{\theta}_{20}$ and $\Delta\hat{\theta}_1, \ldots, \Delta\hat{\theta}_{20}$, where $\Delta\hat{\theta}_i = \hat{\theta}_i - \theta_i$. Secondly, we perform OLS on the accumulated groups, that is, on the first group to get

$\bar{\theta}_1$ and $\Delta\bar{\theta}_1$ ($\Delta\bar{\theta}_1 = \bar{\theta}_1 - \theta_1$), and then on the first two groups combined to get $\bar{\theta}_2$ and $\Delta\bar{\theta}_2$, and so on. Both kinds of simulations are repeated for 1000 times (Monto Carlo method) and $E\left|\Delta\hat{\theta}_1\right|, \ldots, E\left|\Delta\hat{\theta}_{20}\right|$, and $E\left|\Delta\bar{\theta}_1\right|, \ldots, E\left|\Delta\bar{\theta}_{20}\right|$ are approximately obtained from these 1000 runs each. The resulting $E\left|\Delta\hat{\theta}_i\right|$ of different groups are shown in Figure 2a and the resulting $E\left|\Delta\bar{\theta}_i\right|$ of accumulated groups are shown in Figure 2b. It is



(a) $E\left(\left|\Delta\hat{\theta}\right|\right)$ for different groups of data



(b) $E\left(\left|\Delta\bar{\theta}\right|\right)$ for accumulated groups of data

Figure 3.2: Example for ilustration

seen from Figure 2a that for this example, $E\left|\Delta\hat{\theta}_i\right|$ increases with i, which means that the estimate is better for the group with a higher SNR. Figure 2b shows that the best estimate is $\bar{\theta}_4$. This means that the OLS estimate of a subset of the total data could be better than that of the total data if the SNR of the subset is higher than that of the whole set.

3.4.2 RGWLS and GLS

According to (3.31), the data matrices are expressed as

$$\begin{aligned} Y_R &= TY, \\ \Phi_R &= T\Phi, \\ \mathcal{E}_R &= T\mathcal{E}, \end{aligned} \tag{3.60}$$

where T is a mutation matrix given by

$$T = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & \cdots \\ 1 & 0 & 0 & 0 & 1 & 0 & \cdots \\ 0 & 0 & 1 & 1 & 0 & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}.$$

Substituting (3.60) into (3.35) yields

$$\Delta\theta_R = \left(\Phi^T M \Phi\right)^{-1} \left(\Phi^T M \mathcal{E}\right), \tag{3.61}$$

where $M = T^T W T$. Then we have

$$E\left(\Delta\theta_R \Delta\theta_R^T\right) = E\left(\left(\Phi^T M \Phi\right)^{-1} \left(\Phi^T M \mathcal{E}\mathcal{E}^T M \Phi\right) \left(\Phi^T M \Phi\right)^{-1}\right). \tag{3.62}$$

It is difficult to make further analysis if Φ is stochastic and Φ and $\mathcal{E}$ are correlated.

Assume that Φ is non-stochastic, Φ^T and $\mathcal{E}$ are uncorrelated with $E(\mathcal{E}) = 0$ and $E(\mathcal{E}\mathcal{E}^T) = R$. The GLS estimate $\hat{\theta}_G$ is BLUE of θ. Let G be a $N \times N$ matrix. Consider the transformed specification

$$GY = G\Phi\theta + G\mathcal{E}. \tag{3.63}$$

The resulting OLS estimate is

$$\hat{\theta}_G = \left(\Phi^T G^T G \Phi\right)^{-1} \left(\Phi^T G^T G Y\right), \tag{3.64}$$

and the estimation error is

$$\begin{aligned}\Delta\theta_G &= \hat{\theta}_G - \theta \\ &= \left(\Phi^T G^T G \Phi\right)^{-1}\left(\Phi^T G^T G \mathcal{E}\right) \\ &= \left(\Phi^T Q \Phi\right)^{-1}\left(\Phi^T Q \mathcal{E}\right),\end{aligned} \tag{3.65}$$

where $Q = G^T G$. Then we have

$$\begin{aligned}E\left(\Delta\theta_G \Delta\theta_G^T\right) &= E\left(\left(\Phi^T Q \Phi\right)^{-1}\left(\Phi^T Q \mathcal{E}\mathcal{E}^T Q \Phi\right)\left(\Phi^T Q \Phi\right)^{-1}\right) \\ &= \left(\Phi^T Q \Phi\right)^{-1}\left(\Phi^T Q E\left(\mathcal{E}\mathcal{E}^T\right) Q \Phi\right)\left(\Phi^T Q \Phi\right)^{-1} \\ &= \left(\Phi^T Q \Phi\right)^{-1}\left(\Phi^T Q R Q \Phi\right)\left(\Phi^T Q \Phi\right)^{-1}.\end{aligned} \tag{3.66}$$

Söderström [125] shows that when $Q = R^{-1}$, $\hat{\theta}_G$ is BLUE of θ. And the desired transformation G can be designed based on the matrix Q.

There is a big difference between our method and the GLS method. It is difficult to analyze (3.62) since the assumptions for GLS do not hold. The mathematical expectation cannot be taken through the nonlinear operators. Therefore, we provide examples in Section 3.5 to show the effects of our method for finite-sample cases.

3.5 Simulation Examples

In this part, examples are given to illustrate the proposed method. We take very large N for asymptotic analysis in the first two examples. Finite-sample cases are considered in the remaining examples.

When N is large, we introduce the notation R_{EEI} by

$$R_{EEI} = \frac{\|\Delta\theta_R\|_2}{\|\Delta\theta\|_2},$$

which describes the size of the ratio between the estimation errors of RGWLS and OLS. Apparently, if $R_{EEI} < 1$, the RGWLS estimate does have improvement. We will discuss how the number of data points, the noise level and the transformation ratio affect R_{EEI}.

Similarly, we introduce the notation R_{IV} by

$$R_{IV} = \frac{\|\Delta\theta_{IV}\|_2}{\|\Delta\theta\|_2},$$

where $\Delta\theta_{IV}$ is the estimation error obtained with IV method. The instrumental variable Z_t could be chosen according to [128].

Example 3.1. Consider an ARMA(1,1) model,

$$y_t = 0.95y_{t-1} + \gamma\varepsilon_t, \tag{3.67}$$

where $\varepsilon_t = e_t - 0.95e_{t-1}$, $y_0 = 1000$, and γ is a chosen parameter that is used to adjust the noise level. Choose $Z_t = 1/(t)^2$, where $t = 1, 2, \ldots N-1$. Simulation results are shown in Table 3.2 and Table 3.3.

Table 3.2: R_{EEI} of Example 3.1

	r=1/20			r=1/40		
γ \ N	5000	10000	20000	5000	10000	20000
5	0.056	0.057	0.056	0.029	0.030	0.030
25	0.082	0.088	0.110	0.046	0.053	0.062
50	0.128	0.168	0.241	0.076	0.109	0.151

Table 3.3: R_{IV} of Example 3.1

γ \ N	5000	10000	20000
5	0.160	0.082	0.043
25	0.042	0.026	0.018
50	0.035	0.027	0.023

Example 3.2. Consider a second order system with exogenous input,

$$y_t = 1.5y_{t-1} - 0.7y_{t-2} + u_{t-1} + \gamma\varepsilon_t, \tag{3.68}$$

where $u_t = 1$, $\varepsilon_t = e_t - 1.5e_{t-1} + 0.7e_{t-2}$ and $y_0 = y_1 = 0$. Choose $Z_t = [1/(t)^2, 1, t]$, where $t = 1, 2, \ldots N - 2$. Simulation results are shown in Table 3.4 and Table 3.5.

Table 3.4: R_{EEI} of Example 3.2

	r=1/20			r=1/40		
γ \ N	5000	10000	20000	5000	10000	20000
0.1	0.399	0.472	0.496	0.259	0.347	0.410
0.5	0.598	0.694	0.759	0.505	0.586	0.660
1	0.707	0.868	0.885	0.635	0.789	0.853

Table 3.5: R_{IV} of Example 3.2

γ \ N	5000	10000	20000
0.1	0.101	0.006	0.262
0.5	0.274	0.022	2.460
1	0.732	0.050	1.771

Generally, as can be observed from Table 3.2 to Table 3.5:

- for all simulation cases, $R_{EEI} < 1$;
- for the same γ and r, a larger N leads to a larger R_{EEI};
- for the same N and r, a smaller γ leads to a smaller R_{EEI};
- for the same N and γ, a smaller r results in a smaller R_{EEI};
- We have $R_{EEI} < R_{IV}$ for some cases but not all. Note that the result of our method depends on how to choose grouping methods and the parameter, while the result of

IV relies much on the selected instrumental variables. The RG method provides a new option for system identification under coloured noise with comparable results to the IV one at least. Since the RG is new, future research on it may advance it with much better results.

Although it is difficult to analyze the finite-sample properties of least squares estimate analytically, we still perform Monte Carlo experiments to get approximate results. In the following part, two examples are presented for illustration. The simulation for each example is repeated 100 times to obtain $E\left(\|\Delta\theta\|_2\right)$ and $E\left(\|\Delta\theta_R\|_2\right)$. Denote

$$R_{EEF} = \frac{E\left(\|\Delta\theta_R\|_2\right)}{E\left(\|\Delta\theta\|_2\right)}, \tag{3.69}$$

which describes the size of the ratio between the finite-sample bias of RGWLS estimate and that of OLS estimate. Apparently, if $R_{EEF} < 1$, the RGWLS estimate is better than the OLS estimate. We study how N, γ and r affect R_{EEF}.

Example 3.3. Considering model (3.67), simulation results are shown in Table 3.6.

Table 3.6: R_{EEF} of Example 3.3

	r=1/20			r=1/40		
N / γ	500	1000	2000	500	1000	2000
5	0.381	0.162	0.097	0.715	0.186	0.090
25	0.089	0.057	0.067	0.159	0.049	0.031
50	0.067	0.069	0.090	0.100	0.041	0.042

Example 3.4. Considering model (3.68), simulation results are shown in Table 3.7.

As can be observed from Table 3.6 and Table 3.7, $R_{EEF} < 1$ for all simulation cases. Although it is difficult to make theoretical analysis, the proposed method still leads to a better estimate. In addition, N, γ and r are all not necessarily linked to R_{EEF}, which

Table 3.7: R_{EEF} of Example 3.4

	r=1/20			r=1/40		
N / γ	500	1000	2000	500	1000	2000
0.1	0.119	0.196	0.288	0.087	0.103	0.153
0.5	0.528	0.556	0.575	0.414	0.478	0.500
1	0.577	0.645	0.715	0.537	0.562	0.618

differs from the asymptotic case. In the future, we will try to study how other factors influence the result, such as model structures and input signals.

3.6 Conclusions

In this chapter, an improved system identification method has been proposed based on Renormalization Group. Given a fine data set, it is easy to obtain the OLS estimate. By Renormalization Group, a coarse data set is created from the original fine data set. This new data set enables us to get the RGWLS estimate. Through comparisons, we find that the estimation error of RGWLS estimate could be smaller than that of the OLS estimate. Thus, the proposed method could have improvement. It should be pointed out that the proposed method requires sufficient data. Furthermore, its in-depth theory and additional potential values are to be investigated.

Chapter 4

System Identification in Presence of Outliers

4.1 Introduction

The previous chapter presents an improve system identification method with RG when the observed output is corrupted with noise. In practice, the observations may be contaminated not only by noise, but also by outliers. The existence of outliers may result in poor outcome of system identification. Researchers have proposed some methods for outlier detection. In context of control and automation, visual inspection is an empirical way to pick up outliers. However, it is subjective, inaccurate and thus unreliable. Popular statistical techniques include the three-sigma rule [44], the linear and nonlinear filters [45, 46], but they are not effective [34]. Robust regression methods [47–49] are inherently less sensitive to outliers but often difficult to implement [34]. In the field of image processing, there have been recently some works on outlier detection. Candès et al. [57] and Wright et al. [58] proposed the Robust Principal Component Analysis (RP-CA), which recovers a low-rank matrix from the corrupted data matrix. Researchers have proposed some first-order fast algorithms [61–64] to solve RPCA. Although these methods are fast, there is a limitation that the linear structures such as Hankel and Toeplitz cannot be preserved in the resulting matrices. Liu et al. [66] developed an efficient implementation of the interior-point method to recover a low-rank matrix which reserves the linear matrix structure. However, only noise instead of outliers was considered.

In this chapter, we consider an outlier detection problem for a signal from a dynamic system. It is formulated as a low-rank and sparse matrices decomposition problem:

$$\begin{aligned} \min_{S} \quad & \mathbf{rank}(L) + \gamma \left\|S\right\|_0 \\ \text{subject to} \quad & L + S = D, \end{aligned}$$

where D is a Hankel matrix formed from the measurement signal of a dynamic system, L is a low-rank matrix, S is a sparse matrix and γ is a trade-off parameter. A fast algorithm is developed which preserves Hankel matrix structure and solves the problem accurately. Furthermore, a realistic but complex situation is addressed where the output observations contain both noise and outliers, and the decomposition method in general will fail due to the presence of noise. A novel approach based on under-sampling and averaging is proposed to de-noise while keeping the salient behaviors of outliers. It overcomes the drawback of the existing denoising techniques which smooth both noise and outliers. The effectiveness of proposed method is illustrated with extensive simulation.

The rest of this chapter is organized as follows. Section 4.2 formulates our problem and Section 4.3 gives the general solution. Section 4.4 details the fast algorithm, while Section 4.5 discusses the analysis and implementation. Section 4.6 handles noise and outliers together. Section 4.7 shows simulation studies and Section 4.8 concludes the chapter.

List of notations:

$\mathbb{R}^m$	Real $m \times 1$ vectors.
$\mathbb{R}^{m\times n}$	Real $m \times n$ matrices.
$\mathbb{R}_s^{n\times n}$	Real symmetric $n \times n$ matrices.
$\mathbb{R}_d^{n\times n}$	Real diagonal $n \times n$ matrices.
σ_i	ith largest singular value of a given matrix.
$\mathbf{1}_n$	n-dimensional vector with all components one.
$\lVert w\rVert_2$	2-norm of the vector w: $\lVert w\rVert_2 = \sqrt{\sum\limits_{i=1}^{n} \lvert w_i\rvert^2}$.
$w * v$	convolution of the vector w and the vector v.
I	Identity matrix.
$M > 0$	The matrix M is positive definite.
M^T	Transpose of the matrix M.
$\mathbf{trace}(M)$	Trace of the square matrix M.
$\mathbf{rank}(M)$	Rank of the matrix M.
$\lVert M\rVert_0$	0-norm of the matrix M: number of nonzero entries of M.
$\lVert M\rVert_1$	1-norm of the matrix M: $\lVert M\rVert_1 = \sum_{i,j} \lvert M_{ij}\rvert$.
$\lVert M\rVert_\infty$	l_∞-norm of M seen as a long vector: $\lVert M\rVert_\infty = \max_{i,j} \lvert M_{ij}\rvert$.
$\lVert M\rVert_F$	Frobenious norm of the matrix M: $\lVert M\rVert_F = \sqrt{\mathbf{trace}(M^T M)}$.
$\lVert M\rVert_*$	Nuclear norm of the matrix M: $\lVert M\rVert_* = \sum_{i=1} \sigma_i$.

$M \circ N$	Hadamard product of matrices M and N: $(M \circ N)_{ij} = M_{ij}N_{ij}$.
diag(w)	**diag**$(w) = diag(w_1, w_2, \ldots, w_n)$ for $w = [w_1, w_2, \ldots, w_n]^T$.
vec(M)	**vec**$(M) = [m_{11}, \ldots, m_{p1}, m_{12}, \ldots, m_{p2}, \ldots, m_{1q}, \ldots, m_{pq}]^T$ for $M = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1q} \\ m_{21} & m_{22} & \cdots & m_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ m_{p1} & m_{p2} & \cdots & m_{pq} \end{bmatrix}$.
mat(M)	**mat**$(M) = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1q} \\ m_{21} & m_{22} & \cdots & m_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ m_{p1} & m_{p2} & \cdots & m_{pq} \end{bmatrix}$ for $M = diag(m_{11}, m_{21}, \ldots, m_{p1}, m_{12}, m_{22}, \ldots, m_{p2}, \ldots, m_{1q}, m_{2q}, \ldots, m_{pq})$.

4.2 Problem Formulation

Outliers refer to the data points in a data set that are inconsistent with expected normal behavior based on most of the available data [34, 56]. Outliers may occur in the measured data due to human mistakes, instrument errors, faults in systems and so on. Processing or analyzing data with the contaminated data can lead to serious consequences. Outlier detection is receiving increasing attention in a wild range applications in science and engineering such as bioinformatics, computer vision, image processing and system modelling. Our goal is to detect outliers from the measured data and recover the clean data.

If the clean data is used for data analysis or signal processing, better results could be expected.

In context of system identification, consider a discrete-time system described by

$$y_t + a_1 y_{t-1} + \ldots + a_n y_{t-n} = b_0 u_t + b_1 u_{t-1} + \ldots + b_m u_{t-m}, \quad t = 1, 2, \ldots, N, \tag{4.1}$$

where u_t is the input and y_t the true output. Suppose the observed output is given by

$$\bar{y}_t = y_t + z_t, \tag{4.2}$$

where z_t is zero or an outlier. Let

$$\theta^T = \left[\begin{array}{c} a_1, \ldots, a_n, b_0, b_1, \ldots, b_m \end{array} \right], \tag{4.3}$$

and

$$\varphi_t^T = \left[\begin{array}{c} \bar{y}_{t-1}, \ldots, \bar{y}_{t-n}, u_t, u_{t-1}, \ldots, u_{t-m} \end{array} \right].$$

With the data of $\{u_t, \bar{y}_t, t = 1, 2, \ldots, N\}$, the least squares estimate for θ which minimizes the loss function,

$$J = \frac{1}{2} \sum_{t=1}^{N} \left(\bar{y}_t - \varphi_t^T \theta \right)^2,$$

is given by

$$\hat{\theta} = \left[\sum_{t=1}^{N} \varphi_t \varphi_t^T \right]^{-1} \left[\sum_{t=1}^{N} \varphi_t \bar{y}_t \right]. \tag{4.4}$$

However, the ordinary least squares (OLS) estimate from (4.4) could be biased even if z_t is a white noise [10, 125]. The instrumental variable (IV) method [129, 130] is to reduce the bias of the OLS estimate and yields the following estimate:

$$\hat{\theta} = \left[\sum_{t=1}^{N} \Omega_t \varphi_t^T \right]^{-1} \left[\sum_{t=1}^{N} \Omega_t \bar{y}_t \right], \tag{4.5}$$

where the elements in matrix Ω_t are called instruments or instrumental variables. It is obvious that the outliers in $\bar{y}$ may cause big errors in parameter estimation and should be eliminated from the observed signal before (4.5) is applied.

It is noted that there are some works on outlier detection and clean data recovery in computer vision. For example, in the domain of video surveillance, people would like to obtain a good model for the background variations of a scene. When the scene is photoed, the image signal of the background variation is corrupted by possible foreground objects. The resulting data matrix D formed by columns of grayscale frames can be expressed as $D = L + S$, where L is from the background variations and S from the foreground objects. The goal for the image processing is to recover the clean data L from the observed corrupted data D. Due to the nature of imaging, it is reasonable to assume that L is low-rank. Further, the foreground objects occupy only a fraction of image pixels, and S is thus sparse. As a result, the processing job is transferred to a mathematical problem of decomposing D into a low-rank L and a sparse S. A number of algorithms [57, 58, 61–64] have been developed to solve such a problem.

For our problem, it is not trivial to form D with desirable features as above. Let $Y = [\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{N_D}]^T$, where $N_D \leq N$. Then, $D = Y$ or $D = [Y, Y, \ldots, Y]$ will be both of rank 1 only and their ranks cannot be reduced anymore. Therefore, it is desirable to rearrange the measured output $\bar{y}$ in a matrix form, which can be decomposed into a low-rank matrix and a sparse matrix. Construct a Hankel form from $\bar{y}_i$ as follows,

$$D = \begin{bmatrix} \bar{y}_1 & \bar{y}_2 & \bar{y}_3 & \cdots & \bar{y}_q \\ \bar{y}_2 & \bar{y}_3 & \cdots & \cdots & \bar{y}_{q+1} \\ \bar{y}_3 & \vdots & \ddots & \cdots & \bar{y}_{q+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \bar{y}_p & \bar{y}_{p+1} & \bar{y}_{p+2} & \cdots & \bar{y}_{N_D} \end{bmatrix}, \tag{4.6}$$

where $p + q - 1 = N_D$ and $N_D \leq N$. Without loss of generality, we assume that $p \geq q$, otherwise we can use D^T. It follows from (4.2) that

$$D = L + S, \tag{4.7}$$

where

$$L = [L_p, L_{p+1}, L_{p+2}, \ldots, L_{N_D}] = \begin{bmatrix} y_1 & y_2 & y_3 & \cdots & y_q \\ y_2 & y_3 & \cdots & \cdots & y_{q+1} \\ y_3 & \vdots & \ddots & \cdots & y_{q+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_p & y_{p+1} & y_{p+2} & \cdots & y_{N_D} \end{bmatrix} \tag{4.8}$$

and

$$S = \begin{bmatrix} z_1 & z_2 & z_3 & \cdots & z_q \\ z_2 & z_3 & \cdots & \cdots & z_{q+1} \\ z_3 & \vdots & \ddots & \cdots & z_{q+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ z_p & z_{p+1} & z_{p+2} & \cdots & z_{N_D} \end{bmatrix}. \tag{4.9}$$

Usually, outliers occur rarely in z, which makes S a sparse matrix. The matrix L is intrinsically low-rank if we choose $p, q \gg n$, which is always possible for system identification since the data points are much more than the system order. Without loss of generality, we show the low-rank property of L for the following second order system,

$$y_t + a_1 y_{t-1} + a_2 y_{t-2} = bu_t.$$

Let $U_p = [u_1, u_2, u_3, \ldots, u_p]^T$. One sees that

$$\begin{aligned}
L_{p+2} &= -a_1 L_{p+1} - a_2 L_p + bU_{p+2}, \\
L_{p+3} &= -a_1 L_{p+2} - a_2 L_{p+1} + bU_{p+3} \\
&= \left(a_1^2 - a_2\right) L_{p+1} + a_1 a_2 L_p - a_1 bU_{p+2} + bU_{p+3} \\
L_{p+4} &= -a_1 L_{p+3} - a_2 L_{p+2} + bU_{p+4}, \\
&= \left(a_1^2 + a_1 a_2 + a_2^2\right) L_{p+1} + \left(a_2^2 - a_1^2 a_2\right) L_p + \left(a_1^2 - a_2\right) bU_{p+2} - a_1 bU_{p+3} + bU_{p+4}, \\
&\qquad \ldots
\end{aligned}$$

and

$$\begin{aligned}
L &= [L_p, L_{p+1}, U_{p+2}, U_{p+3}, U_{p+4}, \ldots, U_{N_D}] \begin{bmatrix}
1 & 0 & -a_2 & a_1 a_2 & a_2^2 - a_1^2 a_2 & \cdots & \cdots \\
0 & 1 & -a_1 & a_1^2 - a_2 & a_1^2 + a_1 a_2 + a_2^2 & \cdots & \cdots \\
0 & 0 & b & -a_1 b & a_1^2 b - a_2 b & \cdots & \cdots \\
0 & 0 & 0 & b & -a_1 b & \cdots & \cdots \\
0 & 0 & 0 & 0 & b & \cdots & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & 0 & 0 & \cdots & b
\end{bmatrix} \\
&\triangleq AB.
\end{aligned}$$

The matrix L has the same rank as A since B is an upper triangular non-singular matrix. The most popular input signals for identification test are step and pseudo random binary sequence (PRBS). If the input is a step function, $U_{p+2} = U_{p+3} = U_{p+4} = \ldots = U_{N_D}$ and

rank$(A) = 3$, which is very small compared with q or p. For a PRBS with $u_t = 0$ or 1, we form its complement series defined by $\bar{u}_t = 0$ if $u_t = 1$; $\bar{u}_t = 1$ if $u_t = 0$. Conduct the test of u_t and collect its response y_t, followed by the test of $\bar{u}_t$ with its response $\bar{y}_t$. Then $(y_t + \bar{y}_t)$ is the response to $(u_t + \bar{u}_t)$, the step test.

Note that S in (4.9) can be re-written as a linear mapping:

$$S = \mathcal{A}(z) = z_1 A_1 + z_2 A_2 + \cdots + z_{N_D} A_{N_D}, \tag{4.10}$$

where $A_1, A_2, \ldots, A_{N_D} \in \mathbb{R}^{p \times q}$, are linear independent coefficient matrices given by

$$A_1 = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}, A_2 = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & \cdots & \cdots & 0 \\ 0 & \vdots & \ddots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, \cdots, \tag{4.11}$$

and $N_D \leq pq$. Substituting (4.10) into (4.7) yields

$$D = L + \mathcal{A}(z). \tag{4.12}$$

Our problem is stated as follows.

Problem 4.1. *Recover L in (4.12), where D and A_i ($i = 1, 2, \ldots, N_D$) are given, L is low-rank and $\mathcal{A}(z)$ is sparse.*

One may formulate Problem 4.1 as the rank minimization problem for L conditional on sparse $\mathcal{A}(z)$ as follows:

$$\begin{aligned} \min_{z} \quad & \textbf{rank}(L) + \gamma \left\| \mathcal{A}(z) \right\|_0 \\ \text{subject to} \quad & L + \mathcal{A}(z) = D, \end{aligned} \tag{4.13}$$

where γ is a chosen parameter. To see why above optimization can find exact decomposition, consider a simple example as follows. Let $y = 1, 1, 1, 1, 1$ and $z = 0, 0, 0, 5, 0$. Then we have $\bar{y} = y + z = 1, 1, 1, 6, 1$ and

$$D = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 6 \\ 1 & 6 & 1 \end{bmatrix}.$$

With $\gamma = \frac{1}{\sqrt{3}}$, the optimization problem becomes

$$\begin{aligned} \min_{z} \quad & f = \textbf{rank}(L) + \tfrac{1}{\sqrt{3}} \left\| \mathcal{A}(z) \right\|_0 \\ \text{subject to} \quad & L + \mathcal{A}(z) = D, \end{aligned}$$

We present three cases of matrix decompositions as follows.

Case 1: Outlier is correctly detected and separated accurately from the clean data:

$$L_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad S_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 5 \\ 0 & 5 & 0 \end{bmatrix},$$

and we have $f = 2.15$.

Case 2: Outlier is correctly detected but not fully separated from the clean data:

$$L_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 4 \\ 1 & 4 & 1 \end{bmatrix}, \quad S_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 2 \\ 0 & 2 & 0 \end{bmatrix},$$

and we have $f = 4.15$.

Case 3: Outlier is wrongly detected:

$$L_3 = \begin{bmatrix} 0.5 & 1 & 1 \\ 1 & 1 & 6 \\ 1 & 6 & 1 \end{bmatrix}, \quad S_3 = \begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

and we have $f = 3.58$. It is easy to see that only Case 1 achieves the minimum value of the cost function f, which is the case by simulation with any number of cases.

4.3 The Solution

Problem (4.13) is highly non-convex. Fortunately, it is shown [58] that a relaxed tractable substitute is obtained by replacing the rank with the nuclear norm, and 0-norm with 1-norm, so that we have the convex surrogate of (4.13) as

$$\begin{aligned} \min_{z} \quad & \|\mathcal{A}(z) - D\|_* + \gamma \|A(z)\|_1 \\ \text{subject to} \quad & L + \mathcal{A}(z) = D. \end{aligned} \tag{4.14}$$

The problem in (4.14) can be recast [131] as the following semidefinite programming (SDP) problem:

$$\begin{aligned} \min_{z,W,U,V} \quad & \gamma \mathbf{1}_p^T W \mathbf{1}_q + \frac{1}{2}\left(\textbf{trace}\,(U) + \textbf{trace}\,(V)\right) \\ \text{subject to} \quad & \begin{bmatrix} \frac{1}{2}U & \frac{1}{2}\left(\mathcal{A}(z) - D\right)^T \\ \frac{1}{2}\left(\mathcal{A}(z) - D\right) & \frac{1}{2}V \end{bmatrix} \geq 0, \\ & -W_{ij} \leq \mathcal{A}(z)_{ij} \leq W_{ij}, \\ & L + \mathcal{A}(z) = D, \end{aligned} \tag{4.15}$$

where $W \in \mathbb{R}^{p\times q}$, $U \in \mathbb{R}_s^{q\times q}$ and $V \in \mathbb{R}_s^{p\times p}$. Problem (4.15) can be solved by the primal-dual interior-point method [132, 133], which is outlined as follows.

Consider an SDP problem:

$$\begin{aligned} \min_{Z} \quad & \textbf{trace}(CZ) \\ \text{subject to} \quad & \mathcal{F}(Z) = h, \\ & Z \geq 0, \end{aligned} \tag{4.16}$$

where $Z \in \mathbb{R}_s^{l\times l}$, the coefficients $C \in \mathbb{R}_s^{l\times l}$ and $h \in \mathbb{R}^m$ are known, and $\mathcal{F}(\cdot)$ is a linear mapping from $\mathbb{R}_s^{l\times l}$ to $\mathbb{R}^m$. Its Lagrange dual function $J_d(x, X)$ is given by

$$\begin{aligned} J_d\left(x, X\right) &= \inf\left\{\textbf{trace}\left(CZ\right) - x^T\left(\mathcal{F}\left(Z\right) - h\right) - \textbf{trace}(XZ)\right\} \\ &= \inf\left\{\textbf{trace}\left(CZ\right) - x^T\mathcal{F}\left(Z\right) - \textbf{trace}(XZ) + x^T h\right\} \\ &= \begin{cases} h^T x, \text{ if } \mathcal{F}_{adj}(x) + X = C, \\ -\infty, \text{ otherwise,} \end{cases} \end{aligned}$$

where $x \in \mathbb{R}^m$ and $X \in \mathbb{R}_s^{l\times l}$ are the Lagrange multipliers, and $\mathcal{F}_{adj}(\cdot)$ is the adjoint mapping of $\mathcal{F}(\cdot)$, for which there holds

$$\textbf{trace}(\mathcal{F}_{adj}(w)Y) = w^T\mathcal{F}(Y)$$

for all $w \in \mathbb{R}^m$ and $Y \in \mathbb{R}_s^{l\times l}$. The dual problem of (4.16) is to maximize $J_d\left(x, X\right)$, that is,

$$\begin{aligned} \max_{x,X} \quad & h^T x \\ \text{subject to} \quad & \mathcal{F}_{adj}(x) + X = C, \\ & X \geq 0. \end{aligned} \tag{4.17}$$

The SDPs in (4.16) and (4.17) are a Lagrange dual. Let Z^* and (x^*, X^*) be the optimal solutions of (4.16) and (4.17), respectively. The optimal duality gap $\left(\textbf{trace}(CZ^*) - h^T x^*\right)$ is zero if and only if the complementary slackness condition, **trace**(X^*Z^*)=0, holds[134–136], which is equivalent [137] to $X^*Z^* = 0$. Now the optimal conditions for (4.16) and

(4.17) become

$$\begin{aligned}
&\mathcal{F}(Z^*) = h,\ Z^* \geq 0, \\
&\mathcal{F}_{adj}(x^*) + X^* = C,\ X^* \geq 0, \\
&X^* Z^* = 0.
\end{aligned} \tag{4.18}$$

The primal-dual interior-point method [135] perturbs the complementary slackness condition to

$$XZ = \mu I,\ \mu > 0.$$

Then compute the solution following the central path equations:

$$\begin{aligned}
&\mathcal{F}(Z) = h,\ Z \geq 0, \\
&\mathcal{F}_{adj}(x) + X = C,\ X \geq 0, \\
&XZ = \mu I.
\end{aligned} \tag{4.19}$$

An infeasible primal-dual interior-point algorithm is described in details in [138] and [139] to solve (4.19). For completeness, it is summarized in Algorithm 4.1. Other primal-dual interior-point algorithms[140] are similar to this one.

It is pointed out [140] that the number of iterations of the interior-point method is usually small and the most significant cost in each iteration is solving (4.20) and (4.21) in Algorithm 4.1, which are obtained by linearization of the nonlinear equations of (4.19). The general way to solve (4.20) and (4.21) is described as follows. The equations (4.20) and (4.21) have the following form:

$$\mathcal{H}(\Delta Z X + Z \Delta X) = D_1, \tag{4.22}$$

$$\mathcal{F}(\Delta Z) = r, \tag{4.23}$$

$$\mathcal{F}_{adj}(\Delta x) + \Delta X = D_2, \tag{4.24}$$

Algorithm 4.1 Infeasible primal-dual interior-point method

```
1: Initialization. Given the initial values of $Z$, $x$ and $X$ with $Z > 0$ and $X > 0$. Choose the positive tolerances $\epsilon_1$, $\epsilon_2$ and $\epsilon_3$.
2: Compute residuals and evaluate stopping criteria. Terminate if the specified maximum number of iterations is reached or the following three conditions are satisfied:
       $\|\mathcal{F}(Z) - h\|_2 \leq \epsilon_1,$
       $\|\mathcal{F}_{adj}(x) + X - C\|_F \leq \epsilon_2,$
       $\textbf{trace}(XZ) \leq \epsilon_3.$
3: Compute the Nesterov-Todd scaling matrix [138]. The matrix $\Phi$ defines a congruence which jointly diagonalizes $Z$ and $X^{-1}$:
       $\Phi^T Z \Phi = \textbf{diag}(\lambda), \; \Phi^T X^{-1} \Phi = \textbf{diag}(\lambda)^{-1},$
   where $\Phi \in \mathbb{R}^{l \times l}$ and $\lambda \in \mathbb{R}^l$ with $\lambda_i > 0$. The computations of $\Phi$ and $\lambda$ are given in details in [139] and [140].
4: Compute the affine scaling directions $\Delta Z_a$, $\Delta x_a$ and $\Delta X_a$. Solve the linear equations:
       $\mathcal{H}(\Delta Z_a X + Z \Delta X_a) = -diag(\lambda)^2,$
       $\mathcal{F}(\Delta Z_a) = -(\mathcal{F}(Z) - h),$                                   (4.20)
       $\mathcal{F}_{adj}(\Delta x_a) + \Delta X_a = -(\mathcal{F}_{adj}(x) + X - C),$
   where $\mathcal{H}(\cdot)$ is defined [141] as
       $\mathcal{H}(M) = \frac{1}{2}(\Phi^T M \Phi^{-T} + \Phi^{-1} M^T \Phi).$
5: Select $\mu$. The parameter $\mu$ is selected as
       $\mu = \frac{\textbf{trace}(ZX)}{l} \left( \frac{\textbf{trace}\left( (Z + \alpha_p \Delta Z_a)(X + \alpha_d \Delta X_a) \right)}{\textbf{trace}(ZX)} \right)^{\delta},$
   where
       $\alpha_p = \min\left\{1, \sup\left\{\alpha_p | Z + \alpha_p \Delta Z_a \geq 0\right\}\right\},$
       $\alpha_d = \min\left\{1, \sup\left\{\alpha_d | X + \alpha_d \Delta X_a \geq 0\right\}\right\},$
   and $\delta$ is an algorithm parameter (a typical value is $\delta = 3$).
6: Compute the centering-corrector steps, $\Delta Z_c$, $\Delta x_c$ and $\Delta X_c$. Solve the linear equations:
       $\mathcal{H}(\Delta Z_c X + Z \Delta X_c) = \mu I - \mathcal{H}(\Delta Z_a \Delta X_a),$
       $\mathcal{F}(\Delta Z_c) = 0,$                                   (4.21)
       $\mathcal{F}_{adj}(\Delta x_c) + \Delta X_c = 0.$
7: Update iterates.
       $Z := Z + \beta_p (\Delta Z_a + \Delta Z_c), \; x := x + \beta_d (\Delta x_a + \Delta x_c), \; X := X + \beta_d (\Delta X_a + \Delta X_c),$
   where
       $\beta_p = \min\left\{1, \sup\left\{\beta_p | Z + \beta_p (\Delta Z_a + \Delta Z_c) \geq 0\right\}\right\},$
       $\beta_d = \min\left\{1, \sup\left\{\beta_d | X + \beta_d (\Delta X_a + \Delta X_c) \geq 0\right\}\right\}.$
   Go to step 2.
```

where D_1, r and D_2 are known. The solution of (4.22) is given [139] by

$$\Delta X = -\Phi\Phi^T \Delta Z \Phi\Phi^T + 2\Phi(D_1 \circ \Lambda)\Phi^T, \tag{4.25}$$

where $\Lambda_{ij} = 1/(\lambda_i + \lambda_j)$. Substituting (4.25) into (4.24) gives an equivalent set of linear equations to (4.22)-(4.24):

$$-T^{-1}\Delta Z T^{-1} + \mathcal{F}_{adj}(\Delta x) = R, \tag{4.26}$$

$$\mathcal{F}(\Delta Z) = r, \tag{4.27}$$

where $T = (\Phi\Phi^T)^{-1} > 0$ and $R = D_2 - 2\Phi(D_1 \circ \Lambda)\Phi^T$. It follows from (4.26) that

$$\Delta Z = T\left(\mathcal{F}_{adj}(\Delta x) - R\right)T. \tag{4.28}$$

Substituting (4.28) into (4.27) gives

$$H\Delta x = r + \mathcal{F}(TRT), \tag{4.29}$$

where

$$H\Delta x = \mathcal{F}\left(T\mathcal{F}_{adj}(\Delta x)T\right).$$

Eq. (4.29) is a positive definite set of linear equations of order m (the number of variables). The overall cost of solving (4.29) includes two parts: the cost of forming H and computing the solution of (4.29). The later part is $O(m^3)$ if the equations are solved by Cholesky decomposition. The former part depends on the structure of $\mathcal{F}(\cdot)$. General implementations of the interior-point method require that $\mathcal{F}(\cdot)$ is expressed [66] as

$$\mathcal{F}(Z) = \left[\textbf{trace}(F_1 Z), \textbf{trace}(F_2 Z), \ldots, \textbf{trace}(F_m Z)\right]^T,$$

where $F_1, F_2, \ldots, F_m$ are coefficient matrices. In this case, the entries of H are given [66] by

$$H_{ij} = \textbf{trace}\,(F_i T F_j T)\,, \quad i,j = 1,\ldots,m. \tag{4.30}$$

Computing (4.30) requires $O\left(\max\left\{ml^3, m^2 l^2\right\}\right)$ since each of the m matrix products (i.e., $F_i T$) needs $O(l^3)$ and each of the m^2 traces of matrix products (i.e., **trace** $(F_i T F_j T)$) requires $O(l^2)$ [66]. Therefore, the overall cost of solving (4.29) is $O\left(\max\left\{m^3, ml^3, m^2 l^2\right\}\right)$. Now, come back to (4.15). Suppose that $p = O(N_D)$ and $q = O(N_D)$. Then, the computational cost per iteration is at least $O(N_D^6)$ since (4.15) has $N_D + pq + p(p+1)/2 + q(q+1)/2$ variables.

4.4 Fast Algorithm

In this section, we present the fast algorithm for solving (4.15) by exploiting its structure. Let

$$Z = \begin{bmatrix} \frac{1}{2}U & Z_{21}^T & 0 & 0 \\ Z_{21} & \frac{1}{2}V & 0 & 0 \\ 0 & 0 & Z_{33} & 0 \\ 0 & 0 & 0 & Z_{44} \end{bmatrix},$$

where $Z_{21} \in \mathbb{R}^{p\times q}$, $Z_{33} \in \mathbb{R}_d^{pq\times pq}$ and $Z_{44} \in \mathbb{R}_d^{pq\times pq}$. Then, (4.15) is rewritten as

$$\begin{aligned} \min \quad & \textbf{trace}(Z) \\ \text{subject to} \quad & Z \geq 0, \\ & Z_{21} = \tfrac{\mathcal{A}(z)-D}{2}, \\ & Z_{33} = \frac{\gamma}{2}\left(\textbf{diag}\left(\textbf{vec}\left(W - \mathcal{A}(z)\right)\right)\right), \\ & Z_{44} = \frac{\gamma}{2}\left(\textbf{diag}\left(\textbf{vec}\left(W + \mathcal{A}(z)\right)\right)\right). \end{aligned} \tag{4.31}$$

The Lagrange dual function $J_d(X)$ is given by

$$J_d(X) = \inf\left(\textbf{trace}(Z) - \textbf{trace}(XZ)\right),$$

where

$$X = \begin{bmatrix} X_{11} & X_{21}^T & 0 & 0 \\ X_{21} & X_{22} & 0 & 0 \\ 0 & 0 & X_{33} & 0 \\ 0 & 0 & 0 & X_{44} \end{bmatrix}$$

is the Lagrange multiplier with $X_{11} \in \mathbb{R}_s^{q\times q}$, $X_{22} \in \mathbb{R}_s^{p\times p}$, $X_{33} \in \mathbb{R}_d^{pq\times pq}$ and $X_{44} \in \mathbb{R}_d^{pq\times pq}$. One sees that

$$
\begin{aligned}
J_d(X) &= \inf\left(\textbf{trace}(Z) - \textbf{trace}(XZ)\right) \\
&= \inf\left\{\frac{1}{2}\textbf{trace}(U) + \frac{1}{2}\textbf{trace}(V) + \gamma\sum_{i,j=1} W_{ij} - \frac{1}{2}\textbf{trace}(X_{11}U) - \frac{1}{2}\textbf{trace}(X_{22}V)\right. \\
&\quad -\textbf{trace}\left(X_{21}\left(\mathcal{A}(z) - D\right)^T\right) - \frac{\gamma}{2}\sum_{i,j=1}\left(\textbf{mat}(X_{33})_{ij}\left(W_{ij} - \mathcal{A}(z)_{ij}\right)\right) \\
&\quad \left. -\frac{\gamma}{2}\sum_{i,j=1}\left(\textbf{mat}(X_{44})_{ij}\left(W_{ij} + \mathcal{A}(z)_{ij}\right)\right)\right\} \\
&= \inf\left\{\textbf{trace}(D^T X_{21})\right. \\
&\quad +\frac{1}{2}\left(\textbf{trace}(U) - \textbf{trace}(X_{11}U)\right) + \frac{1}{2}\left(\textbf{trace}(V) - \textbf{trace}(X_{22}V)\right) \\
&\quad +\gamma\sum_{i,j=1} W_{ij} - \frac{\gamma}{2}\sum_{i,j=1}\left(\textbf{mat}(X_{33})_{ij}W_{ij}\right) - \frac{\gamma}{2}\sum_{i,j=1}\left(\textbf{mat}(X_{44})_{ij}W_{ij}\right) \\
&\quad \left. -\textbf{trace}\left(\mathcal{A}(z)X_{21}^T\right) + \frac{\gamma}{2}\textbf{trace}\left(\mathcal{A}(z)\textbf{mat}(X_{33})^T\right) - \frac{\gamma}{2}\textbf{trace}\left(\mathcal{A}(z)\textbf{mat}(X_{44})^T\right)\right\} \\
&= \begin{cases}
\textbf{trace}(D^T X_{21}), \text{ if} \begin{cases}
X_{11} = X_{22} = I, \\
\frac{\gamma}{2}X_{33} + \frac{\gamma}{2}X_{44} = \gamma I, \\
\mathcal{A}_{adj}(X_{21}) - \frac{\gamma}{2}\mathcal{A}_{adj}(\textbf{mat}(X_{33})) + \frac{\gamma}{2}\mathcal{A}_{adj}(\textbf{mat}(X_{44})) = 0,
\end{cases} \\
-\infty, \text{ otherwise.}
\end{cases}
\end{aligned}
$$

where $\mathcal{A}_{adj}(\cdot)$ is the adjoint mapping of $\mathcal{A}(\cdot)$ and defined as

$$
\mathcal{A}_{adj}(M) = \left[\textbf{trace}\left(A_1^T M\right), \textbf{trace}\left(A_2^T M\right), \ldots, \textbf{trace}\left(A_{N_D}^T M\right)\right]^T,
$$

for all $M \in \mathbb{R}^{p\times q}$. The dual problem of (4.31) is then given by

$$
\begin{array}{ll}
\max & \textbf{trace}(D^T X_{21}) \\
\text{subject to} & X_{33} \geq 0, \\
& X_{44} \geq 0, \\
& \begin{bmatrix} I & X_{21}^T \\ X_{21} & I \end{bmatrix} \geq 0, \\
& \frac{\gamma}{2} X_{33} + \frac{\gamma}{2} X_{44} = \gamma I, \\
& \mathcal{A}_{adj}(X_{21}) - \frac{\gamma}{2}\mathcal{A}_{adj}(\textbf{mat}(X_{33})) + \frac{\gamma}{2}\mathcal{A}_{adj}(\textbf{mat}(X_{44})) = 0.
\end{array} \tag{4.32}
$$

Let Z^* and X^* be the optimal solutions of (4.31) and (4.32), respectively. Under zero optimal duality gap, (4.18) and (4.19) are applied to (4.31) and (4.32) to get

$$
\begin{aligned}
& 2Z_{21}^* = \mathcal{A}(x^*) - D,\ Z_{33}^* = \frac{\gamma}{2}\left(\textbf{diag}\left(\textbf{vec}\left(W^* - \mathcal{A}(x^*)\right)\right)\right), \\
& Z_{44}^* = \frac{\gamma}{2}\left(\textbf{diag}\left(\textbf{vec}\left(W^* + \mathcal{A}(x^*)\right)\right)\right),\ Z^* \geq 0, \\
& \frac{\gamma}{2}X_{33}^* + \frac{\gamma}{2}X_{44}^* = \gamma I,\ \mathcal{A}_{adj}(X_{21}^*) - \frac{\gamma}{2}\mathcal{A}_{adj}(\textbf{mat}(X_{33}^*)) + \frac{\gamma}{2}\mathcal{A}_{adj}(\textbf{mat}(X_{44}^*)) = 0,\ X^* \geq 0, \\
& Z^* X^* = 0,
\end{aligned}
$$

and the central path equations become

$$
\begin{aligned}
& 2Z_{21} = \mathcal{A}(z) - D,\ Z_{33} = \frac{\gamma}{2}\left(\textbf{diag}\left(\textbf{vec}\left(W - \mathcal{A}(z)\right)\right)\right), \\
& Z_{44} = \frac{\gamma}{2}\left(\textbf{diag}\left(\textbf{vec}\left(W + \mathcal{A}(z)\right)\right)\right),\ Z \geq 0, \\
& \frac{\gamma}{2}X_{33} + \frac{\gamma}{2}X_{44} = \gamma I,\ \mathcal{A}_{adj}(X_{21}) - \frac{\gamma}{2}\mathcal{A}_{adj}(\textbf{mat}(X_{33})) + \frac{\gamma}{2}\mathcal{A}_{adj}(\textbf{mat}(X_{44})) = 0,\ X \geq 0, \\
& ZX = \mu I,
\end{aligned} \tag{4.33}
$$

where $\mu > 0$. Algorithm 4.1 in Section 4.3 can be invoked to solve (4.33) but with a high computational cost. In this regard, we develop a more efficient procedure for solving the linearized equations of the nonlinear equations of (4.33).

It follows from (4.26) and (4.27) that a set of linearized equations for (4.33) are given by

$$\Delta Z_{33} - \frac{\gamma}{2}\left(\textbf{diag}\left(\textbf{vec}\left(\Delta W - \mathcal{A}(\Delta z)\right)\right)\right) = B_1, \tag{4.34}$$

$$\Delta Z_{44} - \frac{\gamma}{2}\left(\textbf{diag}\left(\textbf{vec}\left(\Delta W + \mathcal{A}(\Delta z)\right)\right)\right) = B_2, \tag{4.35}$$

$$2\Delta Z_{21} - \mathcal{A}(\Delta z) = B_3, \tag{4.36}$$

$$\frac{\gamma}{2}\Delta X_{33} + \frac{\gamma}{2}\Delta X_{44} = B_4, \tag{4.37}$$

$$-\mathcal{A}_{adj}(\Delta X_{21}) + \frac{\gamma}{2}\mathcal{A}_{adj}(\textbf{mat}(\Delta X_{33})) - \frac{\gamma}{2}\mathcal{A}_{adj}(\textbf{mat}(\Delta X_{44})) = b, \tag{4.38}$$

$$T\begin{bmatrix} 0 & \Delta X_{21}^T & 0 & 0 \\ \Delta X_{21} & 0 & 0 & 0 \\ 0 & 0 & \Delta X_{33} & 0 \\ 0 & 0 & 0 & \Delta X_{44} \end{bmatrix} T + \begin{bmatrix} \frac{1}{2}\Delta U & \Delta Z_{21}^T & 0 & 0 \\ \Delta Z_{21} & \frac{1}{2}\Delta V & 0 & 0 \\ 0 & 0 & \Delta Z_{33} & 0 \\ 0 & 0 & 0 & \Delta Z_{44} \end{bmatrix} = R, \tag{4.39}$$

where $B_1 \in \mathbb{R}_d^{pq\times pq}$, $B_2 \in \mathbb{R}_d^{pq\times pq}$, $B_3 \in \mathbb{R}^{p\times q}$, $B_4 \in \mathbb{R}_d^{pq\times pq}$, $b \in \mathbb{R}^{N_D}$,

$$T = \begin{bmatrix} T_{11} & T_{21}^T & 0 & 0 \\ T_{21} & T_{22} & 0 & 0 \\ 0 & 0 & T_{33} & 0 \\ 0 & 0 & 0 & T_{44} \end{bmatrix} \in \mathbb{R}_s^{(2pq+p+q)\times(2pq+p+q)},$$

$$R = \begin{bmatrix} R_{11} & R_{21}^T & 0 & 0 \\ R_{21} & R_{22} & 0 & 0 \\ 0 & 0 & R_{33} & 0 \\ 0 & 0 & 0 & R_{44} \end{bmatrix} \in \mathbb{R}_s^{(2pq+p+q)\times(2pq+p+q)},$$

are known, with $T_{11} \in \mathbb{R}_s^{q\times q}$, $T_{33} \in \mathbb{R}_d^{pq\times pq}$, $T_{44} \in \mathbb{R}_d^{pq\times pq}$, $R_{11} \in \mathbb{R}_s^{q\times q}$, $R_{33} \in \mathbb{R}_d^{pq\times pq}$, $R_{44} \in \mathbb{R}_d^{pq\times pq}$ and $T > 0$. It follows from (4.36) that $\Delta Z_{21} = \frac{1}{2}(\mathcal{A}(\Delta z) + B_3)$. If ΔX_{21} in (4.39) is known, we have

$$\begin{aligned} \Delta U &= 2(R_{11} - T_{11}\Delta X_{21}^T T_{21} - T_{21}^T \Delta X_{21} T_{11}), \\ \Delta V &= 2(R_{22} - T_{21}\Delta X_{21}^T T_{22} - T_{22}\Delta X_{21} T_{21}^T). \end{aligned} \tag{4.40}$$

Therefore, (4.39) is reduced to

$$\frac{1}{2}A(\Delta z) + T_{22}\Delta X_{21} T_{11} + T_{21}\Delta X_{21}^T T_{21} = R_{21} - \frac{1}{2}B_3, \tag{4.41}$$

$$T_{33}\Delta X_{33} T_{33} + \Delta Z_{33} = R_{33}, \tag{4.42}$$

$$T_{44}\Delta X_{44} T_{44} + \Delta Z_{44} = R_{44}. \tag{4.43}$$

Our strategy to solve the simultaneous equations (4.34)-(4.38) and (4.40)-(4.43) is to find ΔX_{21} in terms of Δz, then obtain Δz, and lastly compute other variables. To simplify (4.41), we compute a block diagonal congruence transformation such that

$$\begin{bmatrix} G_1 & 0 \\ 0 & G_2 \end{bmatrix} \begin{bmatrix} T_{11} & T_{21}^T \\ T_{21} & T_{22} \end{bmatrix} \begin{bmatrix} G_1^T & 0 \\ 0 & G_2^T \end{bmatrix} = \begin{bmatrix} I & \Sigma \\ \Sigma & I \end{bmatrix},$$

where

$$\Sigma = \begin{bmatrix} \textbf{diag}(\sigma_1 \ldots \sigma_q) \\ 0 \end{bmatrix},$$

with $0 \le \sigma_k < 1$, $k = 1, \ldots, q$ (since T is positive definite). The matrices G_1 and G_2 are computed by

$$G_1 = Q^T L_1^{-1}, \quad G_2 = P^T L_2^{-1},$$

where L_1 and L_2 are obtained from the Cholesky decompositions of $T_{11} = L_1 L_1^T$ and $T_{22} = L_2 L_2^T$, and the diagonal matrix $\Sigma \in \mathbb{R}^{p\times q}$ and the orthogonal matrices $P \in \mathbb{R}^{p\times p}$,

$Q \in \mathbb{R}^{q\times q}$ are obtained from the singular value decomposition (SVD):

$$L_2^{-1}T_{21}L_1^{-T} = P\Sigma Q^T. \tag{4.44}$$

Define $\Delta\tilde{X}_{21} = G_2^{-T}\Delta X_{21}G_1^{-1}$, $\tilde{R}_{21} = G_2(R_{21} - \frac{1}{2}B_3)G_1^T$, and $\tilde{\mathcal{A}}(\cdot) = G_2\mathcal{A}(\cdot)G_1^T$, i.e.,

$$\tilde{\mathcal{A}}(z) = z_1\tilde{A}_1 + z_2\tilde{A}_2 + \cdots + z_{N_D}\tilde{A}_{N_D},$$

where $\tilde{A}_i = G_2A_iG_1^T$. Then, (4.41) is reduced to

$$\frac{1}{2}\tilde{\mathcal{A}}(\Delta z) + \Delta\tilde{X}_{21} + \Sigma\Delta\tilde{X}_{21}^T\Sigma = \tilde{R}_{21}, \tag{4.45}$$

and we also have

$$\begin{aligned}
\tilde{\mathcal{A}}_{adj}(\Delta\tilde{X}_{21}) &= \left[\textbf{trace}\left(\tilde{A_1}^T\Delta\tilde{X}_{21}\right), \textbf{trace}\left(\tilde{A_2}^T\Delta\tilde{X}_{21}\right), \ldots, \textbf{trace}\left(\tilde{A_{N_D}}^T\Delta\tilde{X}_{21}\right)\right]^T \\
&= \left[\textbf{trace}\left(G_1{A_1}^TG_2^TG_2^{-T}\Delta X_{21}G_1^{-1}\right), \textbf{trace}\left(G_1{A_2}^TG_2^TG_2^{-T}\Delta X_{21}G_1^{-1}\right), \ldots,\right. \\
&\quad \left.\textbf{trace}\left(G_1{A_{N_D}}^TG_2^TG_2^{-T}\Delta X_{21}G_1^{-1}\right)\right]^T \\
&= \left[\textbf{trace}({A_1}^T\Delta X_{21}), \textbf{trace}({A_2}^T\Delta X_{21}), \ldots, \textbf{trace}({A_{N_D}}^T\Delta X_{21})\right]^T \\
&= \mathcal{A}_{adj}(\Delta X_{21}).
\end{aligned} \tag{4.46}$$

Now we find $\Delta\tilde{X}_{21}$ in (4.45) by exploiting the techniques in [66] as follows. Suppose a mapping

$$\mathcal{S} : \mathbb{R}^{p\times q} \to \mathbb{R}^{p\times q}, \quad \mathcal{S}(M) = M + \Sigma M^T\Sigma,$$

where $\mathcal{S}(\cdot)$ is self-adjoint and can be factored as

$$\mathcal{S}(M) = \mathcal{L}(\mathcal{L}_{adj}(M)),$$

with the mapping $\mathcal{L}(\cdot) : \mathbb{R}^{p\times q} \to \mathbb{R}^{p\times q}$, defined by

$$\mathcal{L}(M)_{ij} = \begin{cases} \sqrt{1-\sigma_i^2\sigma_j^2}M_{ij} + \sigma_i\sigma_j M_{ji}, & i<j, \\ \sqrt{1+\sigma_i^2}M_{ii}, & i=j, \\ M_{ij}, & i>j, \end{cases}$$

and its adjoint $\mathcal{L}_{adj}(\cdot) : \mathbb{R}^{p\times q} \to \mathbb{R}^{p\times q}$, by

$$\mathcal{L}_{adj}(M)_{ij} = \begin{cases} M_{ij}, & i>q, \\ M_{ij} + \sigma_i\sigma_j M_{ji}, & q \geq i > j, \\ \sqrt{1+\sigma_i^2}M_{ii}, & i=j, \\ \sqrt{1-\sigma_i^2\sigma_j^2}M_{ij}, & i<j. \end{cases}$$

The inverse mappings of $\mathcal{L}(\cdot)$ and $\mathcal{L}_{adj}(\cdot)$ are

$$\mathcal{L}^{-1}(M)_{ij} = \begin{cases} (M_{ij} - \sigma_i\sigma_j M_{ji})/\sqrt{1-\sigma_i^2\sigma_j^2}, & i<j, \\ M_{ii}/\sqrt{1+\sigma_i^2}, & i=j, \\ M_{ij}, & i>j, \end{cases}$$

and

$$\mathcal{L}_{adj}^{-1}(M)_{ij} = \begin{cases} M_{ij}, & i>q, \\ M_{ij} - \sigma_i\sigma_j M_{ji}/\sqrt{1-\sigma_i^2\sigma_j^2}, & q \geq i > j, \\ M_{ii}/\sqrt{1+\sigma_i^2}, & i=j, \\ M_{ii}/\sqrt{1-\sigma_i^2\sigma_j^2}, & i<j. \end{cases}$$

Then, it follows from (4.45) that

$$\Delta\tilde{X}_{21} = \mathcal{S}^{-1}(\tilde{R}_{21} - \frac{1}{2}\tilde{A}(\Delta z)). \tag{4.47}$$

To find Δz, substituting (4.46) and (4.47) into (4.34)-(4.38), (4.42) and (4.43), and eliminating all variables except Δz give

$$H\Delta z = g, \tag{4.48}$$

where g is known and H satisfies

$$H\Delta z = \tilde{\mathcal{A}}_{adj}\left(S^{-1}\left(\tilde{\mathcal{A}}\left(\Delta z\right)\right)\right) + 2\gamma^2\mathcal{A}_{adj}\left(\mathbf{mat}\left(\left(T_{33}^2 + T_{44}^2\right)^{-1}\right)\circ\mathcal{A}\left(\Delta z\right)\right), \quad (4.49)$$

for all Δz, i.e.,

$$\begin{aligned} H_{ij} &= \mathbf{trace}\left(\mathcal{L}^{-1}(\tilde{A}_i^T)\mathcal{L}^{-1}(\tilde{A}_j)\right) + 2\gamma^2\mathbf{trace}\left(A_i^T\left(\mathbf{mat}\left(\left(T_{33}^2 + T_{44}^2\right)^{-1}\right)\circ A_j\right)\right), \\ &\quad i, j = 1, \ldots, N_D. \end{aligned} \quad (4.50)$$

One way to solve (4.48) for Δz is to compute each element of H (i.e., (4.50)) and then factor H using Cholesky decomposition. Liu et al. [66] suggest another method to solve (4.48) which is numerically more stable. Note that $\mathcal{S}(\cdot)$ is positive definite, so is H. Factoring H with Cholesky decomposition yields

$$H = H_L^T H_L = \left[H_1^T, H_2^T\right]\begin{bmatrix} H_1 \\ H_2 \end{bmatrix}, \quad (4.51)$$

where

$$H_1 = \left[\mathbf{vec}\left(\mathcal{L}^{-1}(\tilde{A}_1)\right), \mathbf{vec}\left(\mathcal{L}^{-1}(\tilde{A}_2)\right), \ldots, \mathbf{vec}\left(\mathcal{L}^{-1}(\tilde{A}_{N_D})\right)\right],$$

and

$$H_2 = \sqrt{2}\gamma\left(T_{33}^2 + T_{44}^2\right)^{-\frac{1}{2}}\left[\mathbf{vec}\left(A_1\right), \mathbf{vec}\left(A_2\right), \ldots, \mathbf{vec}\left(A_{N_D}\right)\right].$$

It is suggested [66] to use a QR decomposition of H_L to factor H and then to solve (4.48). After Δz is found from (4.48), $\Delta\tilde{X}_{21}$ is computed from (4.47) and ΔX_{21} from $\Delta X_{21} = G_2^T\Delta\tilde{X}_{21}G_1$. Substituting these into (4.34)-(4.39) yields ΔU, ΔV and ΔW.

For later reference, we name Algorithm 4.1 with (4.20) and (4.21) solved by the general way (4.22)-(4.30) as the standard method, while Algorithm 4.1 with (4.20) and (4.21) solved by the fast implementation (4.34)-(4.51) as the proposed method. In addition, if

D, L and S have certain linear structure (Hankel, Toeplitz, etc.), such a structure is preserved in the resulting matrices since it is determined by the mapping $\mathcal{A}$ and $\mathcal{A}$ does not change throughout the proposed method.

Let us now analyze the complexity for each iteration of the proposed method. Remember the assumptions that $p \geq q$ and $N_D \leq pq$. The cost of computing G_1 and G_2 is $O(p^3)$. The cost of computing the SVD (4.44) is $O(pq^2)$ [142]. Computing the mapping

$$\tilde{\mathcal{A}}(\cdot) = G_2 \mathcal{A}\left(\cdot\right) G_1^T = \sum_{i=1}^{N_D} \left(G_2 A_i G_1^T\right)$$

requires $O(N_D p^2 q)$. It costs $O(q^2)$ [66] to evaluate $\mathcal{S}(\cdot)$ and its inverse. Forming H_2 costs $O(pqN_D)$ since T_{33} and T_{44} are diagonal. Equation (4.48) is solved by QR decomposition and the computational cost is $O(2N_D^2 pq)$. Therefore, if $N_D \geq p$, the overall cost is $O(N_D^2 pq)$, otherwise, $O(N_D p^2 q)$. Especially, if we assume $p = O(N_D)$ and $q = O(N_D)$, the overall computational cost per iteration is $O(N_D^4)$, which has been reduced greatly from $O(N_D^6)$ of the standard method.

4.5 Analysis and Implementation

When can the outliers be found with the proposed algorithm? This is an important theoretical issue. Cand‘es et al. [57] solve the following convex optimization problem,

$$\begin{aligned} \min_{S} \quad & \|L\|_* + \gamma \|S\|_1 \\ \text{subject to} \quad & L + S = D. \end{aligned} \tag{4.52}$$

Let the singular value decomposition of $L \in \mathbb{R}^{p \times q}$ be

$$L = \bar{U}\Sigma\bar{V}^T = \sum\nolimits_{i=1}^{\bar{r}} \sigma_i \bar{u}_i \bar{v}_i^T,$$

where $\bar{r}$ is the rank. The incoherence conditions with parameter μ are

$$\max_i \left\|\bar{U}^T e_i\right\|_2^2 \leq \frac{\mu\bar{r}}{p}, \quad \max_i \left\|\bar{V}^T e_i\right\|_2^2 \leq \frac{\mu\bar{r}}{q}, \quad \left\|\bar{U}\bar{V}^T\right\|_\infty \leq \sqrt{\frac{\mu\bar{r}}{pq}}, \tag{4.53}$$

where e_i's are canonical basis vectors.

Theorem 4.1. *[57] Suppose that $L \in \mathbb{R}^{p\times q}$ obeys (4.53) and the support set of S is uniformly distributed among all sets of cardinality $\bar{m}$. Then, there is a numerical constant c such that with probability at least $1 - cp^{-10}$ (over the choice of support of S), (4.52) exactly recovers L and S with $\gamma = \frac{1}{\sqrt{\max(p,q)}}$, provided that*

$$\textbf{\textit{rank}}(L) \leq \rho_{\bar{r}} p \mu^{-1} (\log(p))^{-2} \quad \textit{and} \quad \bar{m} \leq \rho_s pq,$$

where $\rho_{\bar{r}}$ and ρ_s are some positive numerical constants.

In view of the above theorem, the separation of the low-rank and sparse matrices works and thus the outliers are detected in our context if the low-rank component is of reasonably low rank and the sparse component reasonably sparse [57], indeed.

It follows from the Theorem 4.1 that the general choice for γ is $\gamma = \frac{1}{\sqrt{\max(p,q)}}$ [57]. For our application, we usually use a square Hankel matrix, $p = q$. It follows that

$$\gamma = \frac{1}{\sqrt{p}}. \tag{4.54}$$

This choice works well in all our simulation studies to be presented in Section 4.7, though our S may not meet the distribution assumption in Theorem 4.1 exactly.

Consider next computational burden of the proposed method. As analyzed in Section 4.4, the complexity of each iteration in the algorithm is dominated by size of D matrix. It is possible to apply the method to a small subset of the output series so as to reduce complexity greatly provided that the conditions of Theorem 4.1 are still satisfied, which mainly means that the L matrix from this subset has low rank and S is sparse. Furthermore, the method can work by formally setting the size of z being equal to the total number of the data points used, that is, every data point is possibly with outlier, provided that actually there are few outliers, that is, the found z vector has most elements being zero. Such a universal setting for z/A_i gives the maximum amount of complexity. Obviously, it is impossible to have outlier for each data point in practice, otherwise they are no longer outliers. The universal setting above can be avoided and a great deal of computation can be saved if some screening procedure is used to locate possible outlier points and assign z/A_i corresponding to these few points only.

One possible screening procedure to find suspected outliers (may not be exact) is the following three-sigma rule which we adapt from the static case in the literature to our dynamic system case. Initially, we apply the estimation in (4.4) to the entire measurement data set to find $\hat{\theta} = [\hat{a}_1, \ldots, \hat{a}_n, \hat{b}_0, \hat{b}_1, \ldots, \hat{b}_m]^T$. We then calculate the predicted output from

$$\hat{y}_t + \hat{a}_1\hat{y}_{t-1} + \ldots + \hat{a}_n\hat{y}_{t-n} = \hat{b}_0 u_t + \hat{b}_1 u_{t-1} + \ldots + \hat{b}_m u_{t-m},$$

and form the estimation error series or the residual: $\delta y_t = \bar{y}_t - \hat{y}_t$. Let σ be the standard derivation of δy_t. The samples whose δy_t are outside the interval $[-3\sigma, +3\sigma]$ are viewed as suspected outliers. This three-sigma rule in general picks up outliers as well as some other points with big noise in noisy environment, but the size of z vector has been substantially reduced already. Formally, suppose that the suspected outliers using the above rule are at $i = k_1, k_2, \ldots, k_d$, $d \ll N$. Then assign the size of z as d and A_i accordingly for $i = k_1, k_2, \ldots, k_d$. Finally invoke the proposed method for exact outlier detection.

To reduce the size of Hankel matrix, we can split the original data set of $\bar{y}_t$ to subsets as follows.

- Take the surrounding data of one or few suspected outliers to form a new set $\bar{\mathcal{Y}}_{k_j}$ of N_D points. It is suggested that N_D is in the range of $20 \sim 30$ in order to make $N_D/2$ larger than the order of the system. For example, let k_1 be a single suspected outlier and $N_D = 21$. Form a subset of data as $\bar{\mathcal{Y}}_{k_1} = \{\bar{y}_{k_1-\alpha}, \ldots, \bar{y}_{k_1}, \ldots, \bar{y}_{k_1+\beta}\}$ where $\beta - \alpha = 20$.

- Take under-sampling of the data to form a subset, i.e., take one from every n_s points of $\bar{y}_t$ to form a subset $\bar{\mathcal{Y}}_{k_j}$ of N_D points around $\bar{y}_{k_j}$. Again, we suggest that N_D is $20 \sim 30$. For example, let k_1 be a single suspected outlier, $n_s = 10$ and $N_D = 21$, then set

$$\bar{\mathcal{Y}}_{k_1} = \{\bar{y}_{k_1-\alpha n_s}, \ldots, \bar{y}_{k_1-n_s}, \bar{y}_{k_1}, \bar{y}_{k_1+n_s}, \ldots, \bar{y}_{k_1+\beta n_s}\},$$

  where $\beta - \alpha = 20$. Let the corresponding outlier-free set be

$$\mathcal{Y}_{k_1} = \{y_{k_1-\alpha n_s}, \ldots, y_{k_1-n_s}, y_{k_1}, y_{k_1+n_s}, \ldots, y_{k_1+\beta n_s}\}.$$

  It follows from the discrete system theory that the series in $\mathcal{Y}_{k_1}$ satisfies (4.1) with a different set of coefficients, which implies that the Hankel matrix in form of (4.8) formed from $\mathcal{Y}_{k_1}$ is still of low-rank.

The proposed method is then applied to the subsets $\bar{\mathcal{Y}}_{k_j}$ and the computational burden would be reduced.

To know outlier detection performance of our method, one may wish to know if the outliers are correctly detected. To measure it, introduce the detection rate as

$$\text{Detection rate} = \frac{\text{number of correctly detected outliers}}{\text{number of true outliers}} \times 100\%. \tag{4.55}$$

In many applications, the recovered $\hat{L}$ will be used for actual data processing. It would be useful to measure how accurate the recovered data are in comparison with the actual data. Thus, define the detection accuracy [131] as

$$\text{Accuracy} = \frac{\|\mathcal{A}(\hat{z}) - \mathcal{A}(z)\|_F}{\|\mathcal{A}(z)\|_F} + \frac{\left\|\hat{L} - L\right\|_F}{\|L\|_F}, \tag{4.56}$$

where $(\mathcal{A}(\hat{z}), \hat{L})$ is the solution of (4.14).

Once the proposed method is applied, the resulting data may be used to construct a new and "clean" output series for use in system identification. Recall that the proposed method produces the vector $\hat{z}$. The first way to get the "clean" set is to exclude those points of $\bar{y}_t$ corresponding to outliers (the elements of $\hat{z}$ being nonzero) and use the rest which can meet the system equation (4.1). For later reference, we name this as removing outliers (RO). The second way is to subtract $\hat{z}$ from $\bar{y}_t$ to form the recovered series $\hat{y}$ (RS). The third way is to use $\hat{y}$ after the outlier points are removed (RO $\&$ RS).

Finally, the performance of the system identification is evaluated by the following parameter estimation error,

$$\|\Delta\theta\|_2 = \left\|\hat{\theta} - \theta\right\|_2. \tag{4.57}$$

4.6 In Presence of Both Noise and Outliers

In the preceding sections, we detect outliers by the decomposition of data matrix into low-rank and sparse matrices. The proposed solution can work perfectly, that is, full and exact detection of all outliers if there are strictly low-rank and exactly sparse matrices. However, the observations $\bar{y}_t$ are often corrupted by measurement noise in practice as well, that is,

$$\bar{y}_t = y_t + e_t + z_t, \tag{4.58}$$

where e_t is a noise and z_t is zero or an outlier. The existence of noise may reduce the accuracy of outlier detection. To our best knowledge, no noise is considered in the literature on outlier detection. One may attempt to filter out the noise before detecting outliers. One obvious way to do so is a low-pass filter since the noise has high frequencies than the system spectrum in most system identification applications. To be specific, choose a low-pass Butterworth filter of order 10 for simulation studies in the next section.

In general, what is needed to use the proposed method effectively is denoising, which means the reconstruction of a signal from a mixture of signal and noise [143]. The existing denoising methods include Kernel estimators, Spline estimators, Fourier transform and wavelet transform [143]. Among these methods, wavelet based denoising has proven quite effective in a wide range of applications such as signal processing [143], image processing [144], microscopy [145] and geophysics [146].

Consider a finite length signal with additive noise,

$$\bar{y} = y + \sigma_N e,$$

where $\bar{y}$ is the noisy signal, y is the clean signal, e is a Gaussian noise with $e \sim N(0, 1)$

and σ_N is the noise level. Wavelet denoising is to recover y from $\bar{y}$, and performed in the following three steps. The discrete wavelet transform (DWT) is first performed on $\bar{y}$ by passing it through a series of low-pass (g) and high-pass (h) filters. Let c_j and d_j be the approximation and detail coefficients at jth decomposition level, respectively, and $\downarrow 2$ be the down-sampling operation. Then c_j and d_j are given [147] by

$$\begin{aligned} c_1 &= (\bar{y} * g) \downarrow 2, \\ d_1 &= (\bar{y} * h) \downarrow 2, \\ c_j &= (c_{j-1} * g) \downarrow 2, \quad \text{for } j > 1, \\ d_j &= (c_{j-1} * h) \downarrow 2, \quad \text{for } j > 1, \end{aligned}$$

where the down-sampling is in accordance with the Nyquist sampling theorem. Denote $c_{j,k}$ and $d_{j,k}$ as the kth value of approximation coefficients and detail coefficients at jth decomposition level, respectively. The second step of wavelet denoising is shrinking the coefficients $d_{j,k}$ by thresholding to obtain $\hat{d}_{j,k}$. Thresholding is applied to $d_{j,k}$ rather than $c_{j,k}$, since the approximation coefficients represent 'low-frequency' terms which usually contain important components of the signal and are less affected by the noise. There are two general thresholding rules: hard thresholding and soft thresholding. The function of hard thresholding is defined [143] as

$$\hat{d}_{j,k} = \begin{cases} 0, & |d_{j,k}| \le \lambda, \\ d_{j,k}, & |d_{j,k}| > \lambda, \end{cases}$$

where λ is the threshold limit. While the function of soft thresholding is defined [143] as

$$\hat{d}_{j,k} = \begin{cases} 0, & |d_{j,k}| \le \lambda, \\ sign\left(d_{j,k}\right)\left(|d_{j,k}| - \lambda\right), & |d_{j,k}| > \lambda. \end{cases}$$

The threshold limit λ can be chosen based on the SureShrink [148, 149]. Lastly, the denoised signal is obtained by the inverse DWT of the processed coefficients. The processed approximation coefficients $\hat{c}_j$ and the recovered signal $\hat{y}$ are calculated [147] by

$$\begin{aligned} \hat{c}_{j-1} &= (\hat{c}_j \uparrow 2) * g' + (\hat{d}_j \uparrow 2) * h', \quad \text{for } j > 1, \\ \hat{y} &= (\hat{c}_1 \uparrow 2) * g' + (\hat{d}_1 \uparrow 2) * h', \end{aligned}$$

where g' is a low-pass reconstruction filter, h' is a high-pass reconstruction filter and the $\uparrow 2$ operation adds one zero between adjacent samples.

The denoising methods could reduce the noise. However, they may also weaken the saliency of outliers, which makes outlier detection more difficult. Therefore, it is highly desirable to design a technique that reduces the noise effect while keeping the salient behaviours of outliers. We present such a technique as follows.

Consider the system (4.1) with the output observations given by (4.58). Use our three-sigma rule to find suspected outliers. Let the suspected outliers be at $i = k_1, k_2, \ldots, k_d$, $d \ll N$. Adopt the under-sampling, that is, take one from every n_s points of $\bar{y}_t$ to form a smaller data set $\bar{\mathcal{Y}}_{k_j}$ of N_D points around $\bar{y}_{k_j}$ for each k_j, $j = 1, 2, \ldots, d$. For example, let $j = 1$, $n_s = 10$ and $N_D = 21$, so that we form a new set of more time separated 21 points as follows:

$$\bar{\mathcal{Y}}_{k_1} = \{\bar{y}_{k_1-\alpha n_s}, \ldots, \bar{y}_{k_1-n_s}, \bar{y}_{k_1}, \bar{y}_{k_1+n_s}, \ldots, \bar{y}_{k_1+\beta n_s}\}, \tag{4.59}$$

where $\beta - \alpha = 20$. To reduce noise in $\bar{\mathcal{Y}}_{k_1}$ as much as possible while keeping the outlier point as intact as possible, we keep $\bar{y}_{k_1}$ from filtering. But for other points in $\bar{\mathcal{Y}}_{k_1}$, we resort to the fine data points around them for filtering. Say for $\bar{y}_{k_1-\alpha n_s}$, take

$$\tilde{y}_{k_1-\alpha n_s} = \frac{\sum_{i=1}^{9} (\bar{y}_{k_1-\alpha n_s+i-5})}{9}$$

to replace $\bar{y}_{k_1-\alpha n_s}$. The filtered data set for $\tilde{\mathcal{Y}}_{k_1}$ is formed as

$$\tilde{\mathcal{Y}}_{k_1} = \{\tilde{y}_{k_1-\alpha n_s}, \dots, \tilde{y}_{k_1-n_s}, \bar{y}_{k_1}, \tilde{y}_{k_1+n_s}, \dots, \tilde{y}_{k_1+\beta n_s}\} \tag{4.60}$$

and is used in the proposed method to detect the true outlier. It is seen that the averaging is applied to non-outlier points so that noise can be reduced, where as it is not applied to the suspected outlier points so that their salient feature remains. For easy reference, the above technique to form the filter set (4.60) is called under-sampling with averaging.

Unlike the noise free case where most elements of the solution $\hat{z}$ to (4.14) are exactly zero with few nonzero elements viewed as outliers, $\hat{z}$ in the noise case is non-zero for all its elements. These elements differ only in their magnitudes, and have to be differentiated between noise and outliers. To this end, use the under-sampling to find a subset $\bar{\mathcal{Y}}_0$ like (4.59) with the same parameters n_s and N_D but exclude any suspected outliers. Then, obtain the solution $\hat{z}_0$ for this set $\bar{\mathcal{Y}}_0$ with the proposed method, and compute the standard derivation of $\hat{z}_0$ as σ_0. Now for the solution $\hat{z}$ to each filter set $\tilde{\mathcal{Y}}_{k_j}$, its element $\hat{z}_t$ is regarded as an outlier if $|\hat{z}_t| > 3 \sim 5\sigma_0$.

4.7 Simulation

Case studies are presented for illustration of the proposed method in this section. We consider the following discrete-time system,

$$y_t - 1.9y_{t-1} + 0.95y_{t-2} = 0.05u_t, \quad t = 1, 2, \dots, N. \tag{4.61}$$

The coefficient matrices $A_i \in \mathbb{R}^{p\times q}$ are chosen as in (4.11). Let $p = q = (N_D+1)/2$ and $\gamma = \frac{1}{\sqrt{p}}$. For example, if $N_D = 19$, then $p = q = 10$ and $\gamma = 0.32$. The proposed method was implemented by CVXOPT (version 1.1.6) [150] in Python 2.7, and simulated on a 3.40 GHz processor with 8 GB of memory.

To see detection performance of our method, we compare the proposed method with the Hampel filter method implemented in Matlab R2013a. Consider (4.61) with the measurement given by $\bar{y} = y + z$, where 10% entries of z are disturbed uniformly at random by outliers and the remaining entries of z are zero. For different N_D, results are shown in Table 4.1. As can be seen, the proposed method has achieved precise detection of outliers while the Hampel filter has not.

Table 4.1: Detection Rate

N_D	Proposed method	Hampel filter
19	100%	100%
49	100%	40%
99	100%	75%
149	100%	64.71%
199	100%	66.67%
299	100%	51.61%
399	100%	52.63%

To see the speed of our method, we compare the proposed method with the standard method which is implemented by YALMIP [151] and SDPT3-4.0 [133] in Matlab R2013a. Consider the the previous case. For different N_D, results are shown in Table 4.2. Blank entries in Table 4.2 indicate that the simulation ceased because of memory limitations. As can be seen, the detection rate and the accuracy of the proposed method are as good as those of the standard method, but the time per iteration has been reduced significantly. The standard method cannot handle this problem when $N_D > 99$. Our method performs well on large scale data. Note that the standard method and the proposed method are implemented in different kinds of softwares, Matlab and Python, re-

Table 4.2: Computational time

	Standard method			Proposed method		
N_D	Time per iteration	Accuracy	Detection rate	Time per iteration	Accuracy	Detection rate
19	0.11s	7.07×10^{-7}	100%	0.010s	2.97×10^{-6}	100%
49	0.09s	1.32×10^{-7}	100%	0.022s	3.70×10^{-7}	100%
99	1.09s	2.12×10^{-7}	100%	0.14s	1.25×10^{-6}	100%
149				0.57s	9.86×10^{-6}	100%
199				1.38s	8.62×10^{-7}	100%
299				5.71s	1.00×10^{-6}	100%
399				16.44s	1.01×10^{-6}	100%

spectively, since the related toolboxes are separately programmed in Matlab and Python. For a fair comparison of the running speeds of two methods, the common practice is to count time of running a same standard test program [152] for each case and compare. The running time is 0.31s for Matlab R2013a while 0.50s for Python 2.7, and Matlab R2013a is faster than Python 2.7 for the same program. In other word, if the proposed method had been implemented in Matlab R2013a, the running time would have been further reduced. Hence, the comparisons in Table 4.2 is justified but not in favor of our method.

The proposed method can improve system identification. For example, when $N_D = 49$, the singular values of D and resulting low-rank matrix $\hat{L}$, which are sorted descendingly, are depicted in Figure 4.1, where there are obviously three dominating singular values of $\hat{L}$, indicating that the model is of second-order. Figure 4.2 shows the step re-

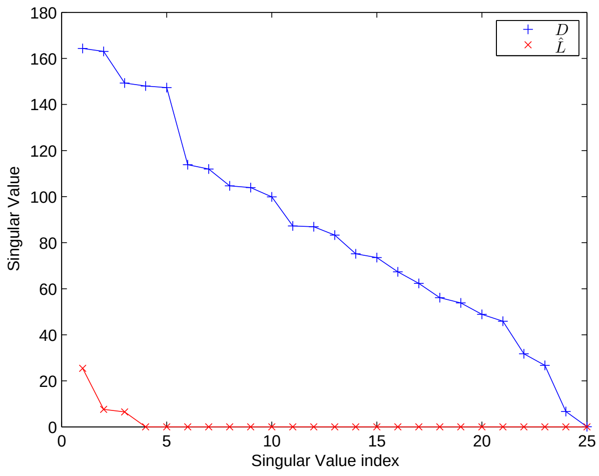


Figure 4.1: Singular value index when $N_D = 49$.

sponses of two models, one of which is identified with the raw signal of order 20 and the other with the recovered signal of order 2. The results clearly show that the identification outcome is much more satisfactory with the recovered signal than that with the raw

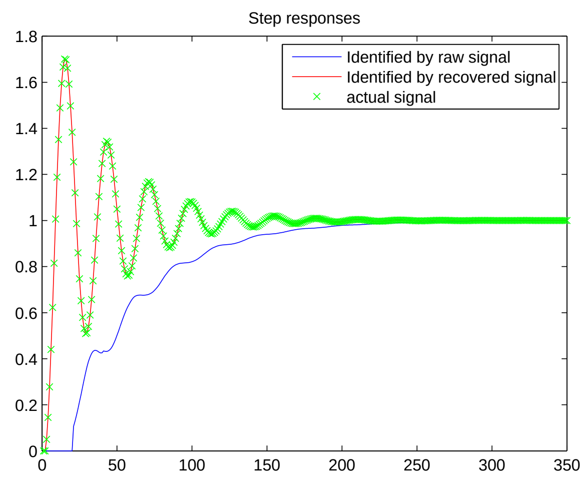


Figure 4.2: Step responses of two models.

signal.

To see computational saving from subset construction in Section 4.5, suppose that $N_D = 199$ and $\bar{y}$ is corrupted by 2 outliers at $i = 40, 120$. Finding the suspected outliers based on the three-sigma rule yields 9 suspected points at $i = 12 \sim 18, 40, 120$, as shown in Figure 4.3. Performing our method on the different subsets formed as described

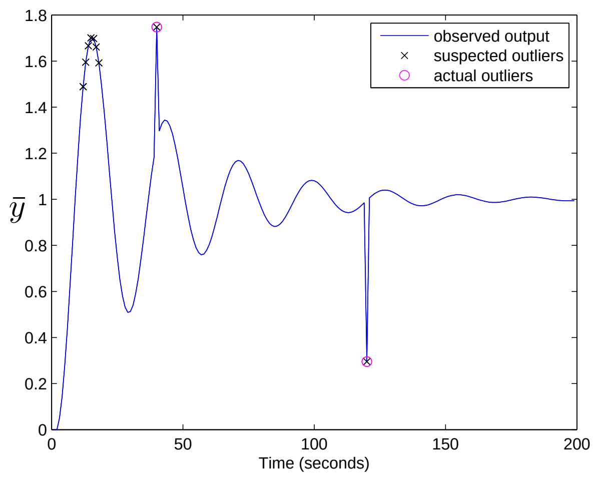


Figure 4.3: Observed output.

in Section 4.5 yields the results, which are shown in Table 4.3. For the surrounding data case, we take 3 subsets of data with $N_D = 29$: $\bar{\mathcal{Y}}_1 = \{\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_{29}\}$, $\bar{\mathcal{Y}}_2 = \{\bar{y}_{26}, \bar{y}_{27}, \ldots, \bar{y}_{54}\}$ and $\bar{\mathcal{Y}}_3 = \{\bar{y}_{106}, \bar{y}_{107}, \ldots, \bar{y}_{134}\}$, which cover all suspected samples. For the under-sampling case, we produce three subsets of samples with $n_s = 2$ and $N_D = 29$: $\bar{\mathcal{Y}}_1 = \{\bar{y}_1, \bar{y}_3, \ldots, \bar{y}_{57}\}$, $\bar{\mathcal{Y}}_2 = \{\bar{y}_2, \bar{y}_4, \ldots, \bar{y}_{58}\}$ and $\bar{\mathcal{Y}}_3 = \{\bar{y}_{90}, \bar{y}_{92}, \ldots, \bar{y}_{146}\}$. As can

Table 4.3: Computational reduction

	Total Time	Detection rate
All the data	31.09s	100%
All the data with reduced z	3.57s	100%
Surrounding data	0.31s	100%
Under-sampling	0.33s	100%

be seen, for the cases in Row 3-5, outliers are detected correctly and the computational burden is reduced compared with that of “All the data”.

To see performance of the under-sampling with averaging technique in presence of both outliers and noise, consider the following measurement,

$$\bar{y}_t = y_t + 0.05e_t + z_t,$$

where e_t is a white noise with $e_t \sim N(0, 1)$. Let the outliers be at $i = 40, 120$. The measurement is shown in Figure 4.4. Using the three-sigma rule yields 8 suspected outliers at $i = 12 \sim 17, 40, 120$. Let $n_s = 2$ and $N_D = 29$. We apply the proposed method to the following subset, $\bar{\mathcal{Y}}_0 = \{\bar{y}_{61}, \bar{y}_{63}, \ldots, \bar{y}_{117}\}$, and yield $\hat{z}_0$ with its standard derivation $\sigma_0 = 0.038$. Then, use $3\sigma_0$ as the threshold to pick up outliers. With different data preprocessing approaches, the detection results are shown in Table 4.4, where CD and WD represent the number of correct detections and wrong detections, respectively. A low-pass Butterworth filter of order 10 with the cutoff frequency of 5Hz is chosen as the low-pass filter. For wavelet denoising, we choose a 3-level wavelet decomposition

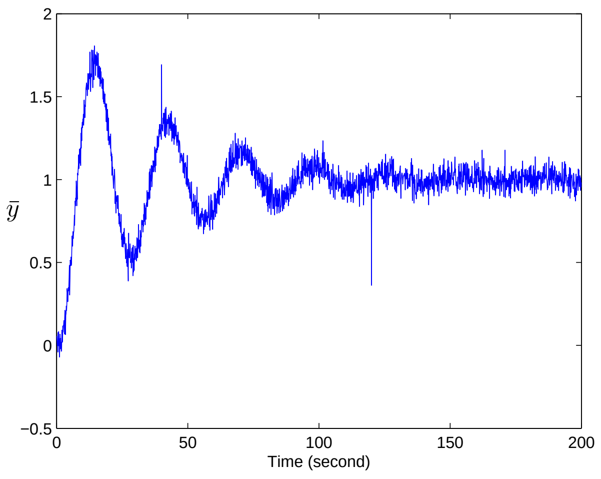


Figure 4.4: Measurement.

with db8 as the mother wavelet and adopt the soft thresholding. As can be observed, for the “Three-sigma” and the “PM” cases, outliers are correctly detected, while some normal points are also detected as outliers. For the “Low-pass+PM” and the “Wavelet+PM” cases, the detection results are poor, since no true outliers are found. For the “Undersampling averaging+PM” case, the true outliers are correctly detected and no normal points are detected as outliers.

To see the performance of system identification, we perform the OLS method on the raw data $\bar{y}_t$ as the benchmark and obtain $\|\Delta\theta\|_2 = 1.76$, which is very poor because of measurement noise and outliers. Note that though the measurement noise is white in the output equation (4.2), the equivalent noise in the system equation (4.1) is colored [10]. Therefore, the IV method should be used for parameter estimation and $\Omega_t = [1/t^2, 1, t]$ is chosen as the instrumental variable, which yields the error of 0.23. Better results are obtained with our outlier detection method followed by IV estimation with “clean” data sets and shown in Table 4.5.

Table 4.4: Outlier detection

		Three-sigma		Proposed Method (PM)		Low-pass+PM		Wavelet+PM		Under-sampling averaging+PM	
		CD	WD	CD	WD	CD	WD	CD	WD	CD	WD
Outliers	2	2	5	2	2	0	3	0	1	2	0
Normal points	197	192	0	195	0	194	2	196	2	197	0

Table 4.5: Parameter Estimation errors

		Three-sigma	Low-pass+PM	Wavelet+PM	Under-sampling average+PM
IV	RO	0.19	0.29	0.23	0.23
	RS	-	0.076	0.052	0.024
	RO & RS	-	0.17	0.048	0.0066
	Raw signal	0.23			

One sees that for different "clean" data sets, use of "RS" and "RO & RS" produces less error than that of "RO", and for different processing techniques, the under-sampling with averaging one achieves best estimation and remarkable improvement over the original IV one.

4.8 Conclusions

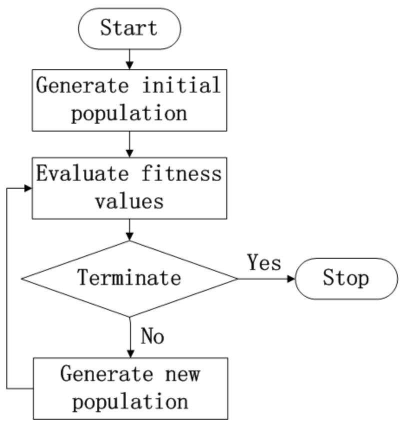
In this chapter, we formally solve an outlier detection problem in the context of system identification. The problem is formulated as the matrix decomposition problem with a low-rank matrix and a sparse matrix. The matrix decomposition problem is recast as an SDP problem. A fast algorithm is proposed to solve such an SDP problem with significant computational saving compared with the standard method and it can preserve Hankel matrix structure in the resulting matrices. It should be pointed out that our method is also applicable for other linear structures such as Toeplitz and moment matrices since different structures can be accommodated by setting relevant mapping A_i matrices. The proposed method has achieved satisfactory detection rate and accuracy. Furthermore, the techniques for constructing subsets while retaining the matrix property are presented for additional reduction of computational burden. In case of significant noise with outliers, an under-sampling with averaging technique for data preprocessing is devised to attenuate the noise effect while keeping the salient behaviour of outliers and enables application of the proposed method for correct outlier detection while other filtering techniques fail. Significant improvement of parameter estimation is achieved from the recovered “clean” data with the proposed method over the one based on the raw data.

Chapter 5

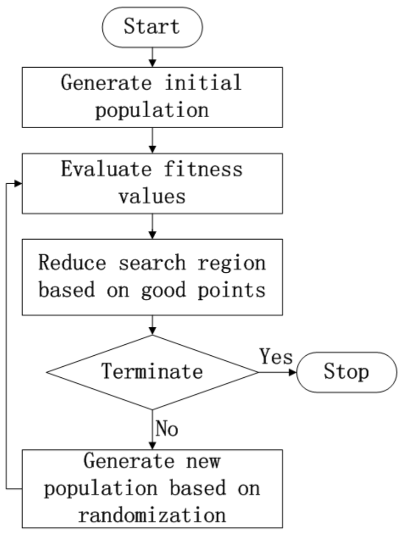
Global Optimization Method Based on Randomized Group Search in Contracting Regions

5.1 Introduction

Chapter 3 and Chapter 4 present new methods helping with getting good models. Once a model is obtained, researchers and engineers may carry out further designs based on it to meet the needs of production and life such as saving cost and improving efficiency. Optimization techniques are vitally important for model based designs in modern engineering and planning. The techniques are classified as either local or global algorithms, where global optimization algorithms may be classified as either metaheuristics or deterministic algorithms [67]. Metaheuristics for global optimization, which have become very popular since last century, are typically inspired by some phenomena from nature. Metaheuristics are usually single-solution based or population-based, where the later deals with a set of solutions rather than a single one. There are three main steps in all the population-based metaheuristics as shown in Figure 5.1a.



(a) General population-based metaheuristics.



Start
Generate initial population
Evaluate fitness values
Reduce search region based on good points
Terminate
Yes
Stop
No
Generate new population based on randomization

(b) Proposed method.

Figure 5.1: Flowcharts.

This chapter aims to present a brand-new population-based method for global optimization problems. The proposed method is stand-alone and not related to any of the existing population-based methods. It has two key novelties as shown in Figure 5.1b. Firstly, the region in which each population lies changes and contracts exponentially, which guarantees convergence of the proposed algorithm. Secondly, each population is generated with randomization, where the size of random samples, is chosen [96] to ensure that the empirical minimum is an estimate of the true minimum within a predefined accuracy with a certain confidence. Its main ideas and contributions are highlighted in comparison with the existing population-based methods such as GA, DE and PSO as follows.

- Generation of search region. A search region is the region in which the samples are drawn to form a population. We believe that there is a higher chance for better points of the next population to appear near the good points of this population than the bad points. Hence, the samples in the current population are ranked ascendingly based on their fitness values and a small subset of top-ranking ones are selected as good points. The group search is then constrained around these good points to generate the next population. Therefore, in our context of group search, the search region is chosen as the intersection of the neighborhoods of the good points and the feasible region. The size of the neighborhoods is set smaller and smaller over iterations to reflect the fact that these good points generally get better and better over the populations as well as closer and closer to the optimum. The contracting neighbors imply that the search region also contracts over iterations, which ensures convergence. The existing methods such as GA, DE and PSO do not have regional concept and operate only on individual samples.

- Generation of populations. In each iteration, a population is randomly generated within the search region. The randomized sampling is adopted since it is most general with no restrictions on the problem and least likely to trap in a local optima, so as to find a global optima for the general applicability. And the good points of the current population carry over to the next population and the optimal fitness value will not deteriorate over iterations. It is worth noting that in some application [96], the performance by randomized algorithms can match or even surpass that of an analytical one. Besides, generation of new population by random sampling is simple in concept and easy to implement, while GA and DE produce new population by “selection”, “crossover” and “mutation” operators, and PSO updates every individual by some mathematical formulas. It is however noted that the randomization alone is neither efficient nor effective to find the global minimum if the population is generated only once, or repeatedly on the same original feasible region.

- Features. The features of our method are shown in comparison with the existing popular methods in Table 5.1. Our method requires ranking of fitness values and its computational time obviously increases linearly with the population size. The optimal fitness value is associated only with the position of one neighborhood, and its influence on next population is not as strong as PSO. In addition, since our method adopts the elitist selection, the optimal fitness value does not get worse over iterations.

- Applications. Our method has no restrictions on the properties of objective functions such as modality, separability and differentiability. It works on both constrained and unconstrained problems. Also, our method applies to both continuous and combinatorial optimization problems, while DE and PSO favor continuous optimization problems. The implementation of the proposed method uses sampling and does not need other complicated techniques. Moreover, our method only has three design parameters, which are not sensitive on simulation results. Hence, it is not necessary to customize the parameters for different problems.

- Performance. The memory requirement of our method is small. We only need to record the positions of the retained points and the size of search region for each iteration. Hence, the simplicity and rapidity are to be expected. In addition, simulation results in Section 5.7 show that our method has a very good accuracy on the global optima of low-dimensional examples and works reasonably well on high-dimensional problems.

The rest of this chapter is organized as follows. Section 5.2 presents the proposed method. Section 5.3 clarifies how to choose the sample size and Section 5.4 details the sampling process. Section 5.5 discusses tuning parameters. The convergence analysis is given in Section 5.6. Section 5.7 presents simulation studies and Section 5.8 concludes the chapter.

5.2 The Proposed Method

To motivate our method, consider a geographer example. Suppose that a geographer wants to measure the highest altitude of a certain district with infinite points. It is obviously impossible for him to measure the altitude of every point in the district. One may adopt the iterative search in the spirit of hill climbing. If he starts with a single point and move away from it over iterations, he will easily get stuck at some local optimum. An advisable way is to start with a group of points instead of a single one and adopt the group search within the search region iteratively. With no information about the terrain, he may take a set of random samples and measure their altitudes. He finds a few samples with highest altitudes. He naturally narrows his search around these points. He then repeats his random sampling and altitude evaluation until he finds an approximate highest point. Our method generalizes the geographer example.

Table 5.1: Comparisons of GA, DE, PSO, and the proposed method

Features	GA	DE	PSO	Proposed method
Ranking of solutions	Yes	No	No	Yes
Influence of population size on computational time	Exponential	Linear	Linear	Linear
Influence of optimal fitness value on population for next iteration	Medium	Less	Most	Medium
No deterioration of optimal fitness value over iterations	No	Yes	No	Yes

Suppose that we want to solve a complex minimization problem given in (1.1) by iterative search. To avoid premature convergence to a local optimum, we search by a group of points instead of a single one for each iteration. Without restrictive assumptions on properties of the given problem, we opt to take random samples within the search region for each iteration. With such a group of samples, we naturally view those with small fitness values as good samples, and believe that there is a higher chance for better points to appear in the neighbors of these good points than bad ones. Therefore, we narrow our next search region in the former. The above process continues till some stopping rule is hit. A formal frame of our method is given in Algorithm 5.1. It is described in details

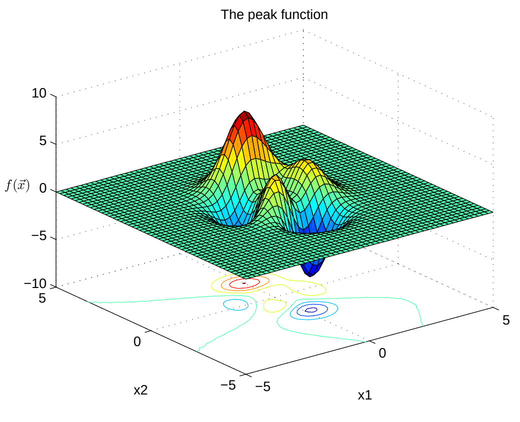
Algorithm 5.1

```
1: Initialization of population.
2: Evaluation of fitness values.
3: Generation of new search region based on good points.
4: Verification of terminate conditions.
       if any terminate condition is met
          Output.
       end do
5: Generation of new population.
6: Repetition from step 2.
```

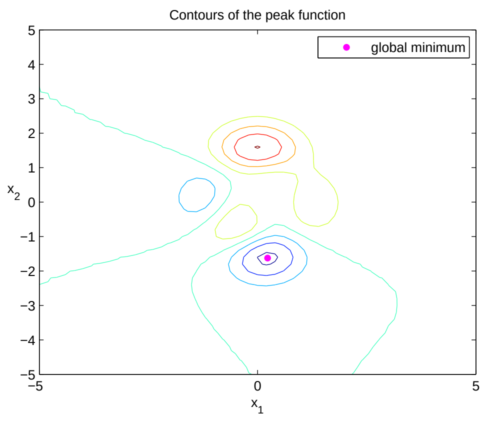
and illustrated as follows with the Peak example [153], which ships with MATLAB and has been taken as a standard example to examine whether an optimization algorithm is able to find the global minimum. The Peak function is given by

$$f(x) = 3(1-x_1)^2 e^{(-x_1^2-(x_2+1)^2)} - 10(\frac{x_1}{5} - x_1^3 - x_2^5)e^{(-x_1^2-x_2^2)} - \frac{1}{3}e^{(-(x_1+1)^2-x_2^2)}, \quad (5.1)$$

where $x = [x_1, x_2]^T$. Its feasible region is $-5 \leq x_1, x_2 \leq 5$. The surface plot and contour lines are respectively shown in Figure 5.2a and Figure 5.2b, where the global optimum is at $(0.2283, -1.6255)$ with the global minimum $f_{min}(x) = -6.5511$.

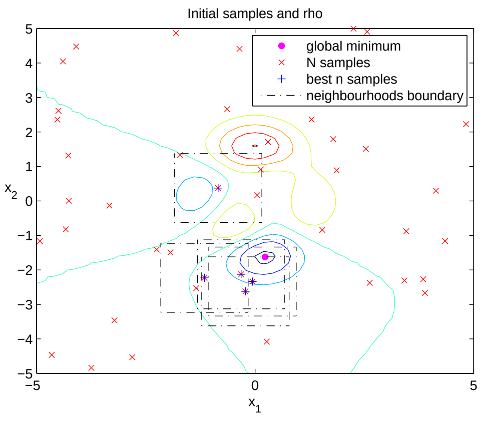


(a) Surface plot.

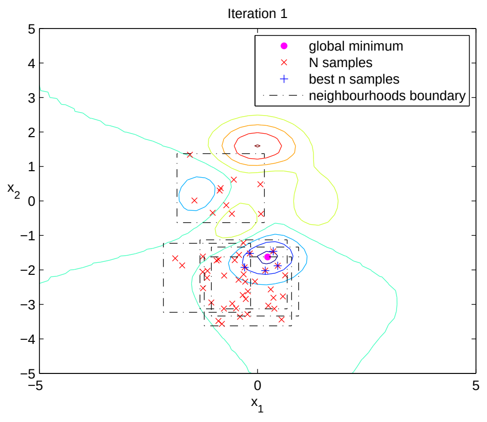


(b) Contour lines.

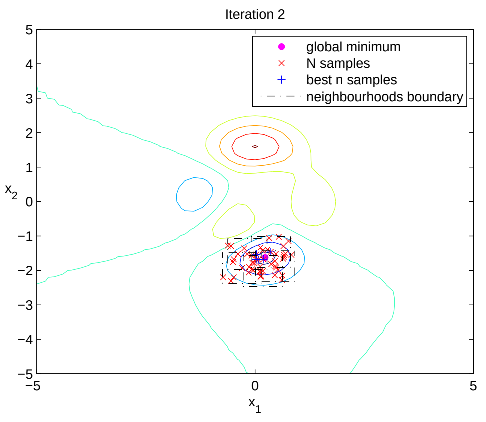
Figure 5.2: Peak function.



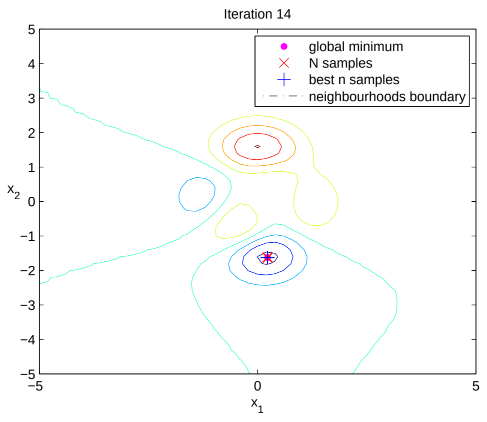
(a) Iteration 1.



(b) Iteration 2.



(c) Iteration 3.



(d) Iteration 15.

Figure 5.3: Iterations of illustrative example.

1. Generate randomly an initial population of N samples within the initial search region, which is the feasible region. The choice of population size N will be discussed in Section 5.3. For the Peak example, in order to show our method clearly by graphics, we let $N = 50$. The initial population is randomly generated and shown in Figure 5.3a as red crosses.

2. Evaluate the fitness values at the N samples and rank them ascendingly based on their fitness values, where the ranked samples are denoted by $\{x_{(k,1)}, x_{(k,2)}, \ldots, x_{(k,N)}\}$, where $x_{(k,i)}$ is the ith best sample of kth iteration. A small subset of top-ranking M points are selected, where $M \ll N$. For the Peak example, evaluate the fitness values at the 50 samples and rank them to obtain $\{x_{(1,1)}, x_{(1,2)}, \ldots, x_{(1,50)}\}$. Let $M = 5$. We then have the selected points as $\{x_{(1,1)}, x_{(1,2)}, \ldots, x_{(1,5)}\}$, which are shown in Figure 5.3a as blue plus signs.

3. Let a new search region be the intersection of the feasible region and the neighborhoods of the selected points. The neighborhood of a point is a geometric unit of any shape with the center being the point itself. For convenience of sampling, we choose an n-dimensional hypercube as a neighborhood for an n-dimensional case. Let $\rho_{(k)}$ be the half side length of the neighboring hypercube in kth iteration. The total volume of these hypercubes is then given by

$$V_k = M(2\rho_{(k)})^n. \tag{5.2}$$

Define the volume shrinkage ratio ν over iterations as

$$\nu = \frac{V_{k+1}}{V_k} \times 100\%. \tag{5.3}$$

It is a turning parameter. Its selection will be discussed in Section 5.5. Once ν is chosen, it is fixed throughout iterations. It follows from (5.2) and (5.3) that $\rho_{(k)}$ is updated as

$$\rho_{(k)} = \rho_{(k-1)}\nu^{\frac{1}{n}}. \tag{5.4}$$

The volume of the hypercube is gradually reduced over iterations, which implies the contraction of the search region. For the Peak example, choose a square as the neighborhood. At the initialization stage, the search region is the feasible region with $V_1 = 100$. Let $\nu = 20\%$. To generate five hypercubes for the next iteration, we have $V_2 = V_1 \times \nu = 20$ from (5.3) and $\rho_2 = 1$ from (5.2) with $n = 2$. For all future iterations, $\rho_{(k)}$ is updated according to (5.4). The neighboring squares of $x_{(1,1)} \sim x_{(1,5)}$ are shown in Figure 5.3a as black squares with dash dot line.

4. Terminate if any of the following conditions is met:

   (i) $k = K_0$, where K_0 is the predefined maximum number of iterations;

   (ii) $\rho_{(k)} < \varepsilon_1$, where ε_1 is a positive tolerance on optimum;

   (iii) $0 < \sum_{i=1}^{M} \left| f(x_{(k-1,i)}) - f(x_{(k,i)}) \right| < \varepsilon_2$, where ε_2 is a positive tolerance on minimum.

   Then, output the optimal solution found by the proposed algorithm. Otherwise, go to step 5. For the Peak example, choose $K_0 = 100$ and $\varepsilon_1 = \varepsilon_2 = 10^{-8}$.

5. Generate randomly $(N - M)$ points within the search region, which form a new population together with the M selected points. For the Peak example, $N = 50$

and $M = 5$. A new population of 45 samples is produced and shown in Figure 5.3b as red crosses.

6. Go to step 2.

For the Peak example, the algorithm stops after 15 iterations because the condition (iii) is hit. To see better on how the process works, Figure 5.3 also shows the second, third and last iterations. As can be seen from Figure 5.3d, our algorithm finds the approximate global minimum.

Algorithm 5.1 has a global consciousness of search. But like other ones in the literature, it cannot always guarantee convergence to the global optimum. It may be modified so as to traverse all the local optima and find the global optimum with arbitrary accuracy if the number of local optima in the feasible region is limited. This inspires us to adopt an repeated use of Algorithm 5.1. In order for it not to converge to the same local optimum with different iterations, a small neighborhood of the current optimum is removed from the feasible region for the next use of Algorithm 5.1. These ideas form Algorithm 5.2, which is described as follows.

Algorithm 5.2

```
1: Execution of Algorithm 5.1.
2: Reduction of feasible region.
3: Repetition from step 1.
```

1. Run Algorithm 5.1 until the termination condition (ii) is met. Let x^* be the resultant optimal solution and ρ^* be the half side length of the current neighboring hypercubes.

2. Remove the neighborhood of x^* from the feasible region.

3. Go to step 1.

Suppose that the minimum distance of any two local optima is d. Set $\varepsilon_1 < \frac{d}{\sqrt{n}}$ so that each removed neighborhood has no other optima than the found one.

5.3 Sample Size

For one iteration of our method, N samples are randomly generated within the search region. Define the empirical minimum as

$$f_{emp}(x) = \min_{i=1,2,\ldots,N} f(x_{(i)}). \tag{5.5}$$

One needs to determine the sample size N to make $f_{emp}(x)$ a good estimate of the true minimum in the search region. The randomized algorithms [154–157] have been applied to solve hard problems arising in control synthesis and verify performance of complex systems where the deterministic algorithms fail. In their context, randomized algorithms are used to estimate the probability that a system with uncertain parameters restricted to a box attains a given level of performance by sampling a certain number of points.

Theorem 5.1 ([154])**.** *Let $\epsilon, \delta \in (0,1)$. If*

$$N \geq \left\lceil \frac{\ln \frac{1}{\delta}}{\ln \frac{1}{1-\epsilon}} \right\rceil, \tag{5.6}$$

then, with probability no less than $(1-\delta)$, the empirical minimum satisfies the following inequality

$$P\{f(x) < f_{emp}(x)\} \leq \epsilon, \tag{5.7}$$

that is,

$$P\{P\{f(x) < f_{emp}(x)\} \leq \epsilon\} \geq 1-\delta. \tag{5.8}$$

Theorem 5.1 states that if N satisfies (5.6), the empirical minimum is an estimate of the true minimum within a predefined accuracy of ϵ with confidence of δ [155]. Note that in Theorem 5.1, the choice of N is independent of problem scale. In our method, Theorem 5.1 is iteratively used and a same N is chosen for every iteration. Intuitively, if a large N is chosen, our method tends to converge with a small number of iterations, whereas a small N may lead to a large number of iterations. Consider two extreme cases for discrete feasible region. If N is infinite large, we may only need to sample once (all points) to obtain the global optimum. On the other hand, if $N = 1$, it may take infinite iterations to get the global optimum. Hence, there is trade-off between N and the number of iterations. Generally, we let both ϵ and δ vary between [0.001, 0.02], giving N in the range of [200, 5000].

5.4 Sampling Process

When N is fixed, samples are randomly generated based on the distribution of x in the search region. The distribution depends on the features of the given problem. In our context, we treat an optimization problem as a kind of black-box problem with no prior information on the distribution, or each point in the search region is equally likely to be optimum. Hence, we adopt the uniform distribution, that is, we take samples uniformly at random within the search region. The implementation of uniform sampling is discussed as follows.

The realization of uniform sampling depends on the shape of a search region. We first consider a simple case that the search region is one hyperrectangle. In this case, sampling uniformly within the whole region is equivalent to sampling uniformly on each dimension of the hyperrectangle and then making a combination, since each dimension is taken as a random variable and all the random variables are independent to each other [158]. Consider a simple 2-dimensional example shown in Figure 5.4. Suppose that we would like to take 20 samples uniformly at random within a square bounded by $[1, 2]$. We first respectively take 20 points uniformly and randomly within $x_1 \in [1, 2]$ and $x_2 \in [1, 2]$ to get $\{x_{1,(\cdot,1)}, x_{1,(\cdot,2)}, \dots, x_{1,(\cdot,20)}\}$ and $\{x_{2,(\cdot,1)}, x_{2,(\cdot,2)}, \dots, x_{2,(\cdot,20)}\}$, which are shown as blue crosses and green circles in Figure 5.4. Then, combining these points yields 20 uniform samples within the square, say, $\{(x_{1,(\cdot,1)}, x_{2,(\cdot,1)}), (x_{1,(\cdot,2)}, x_{2,(\cdot,2)}), \dots, (x_{1,(\cdot,20)}, x_{2,(\cdot,20)})\}$, which are shown as red plus signs. For easy reference, we name this procedure as one-

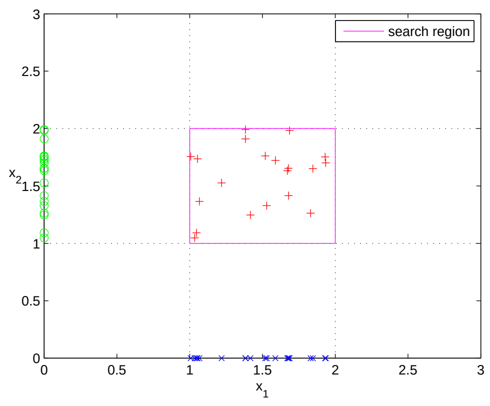


Figure 5.4: Uniform sampling within a square.

hyperrectangle sampling.

Next, suppose that the search region contains multiple hyperrectangles which are of the same size but mutually exclusive. We propose a possible solution and explain it through a 2-dimensional example as shown in Figure 5.5. Let the search region consist of two same-sized rectangles, say, A and B. Move B to B_1, and arrange B_1 and A next to each other along their long sides. Take samples within the union rectangle of A and B_1, and then map the points from B_1 to B.

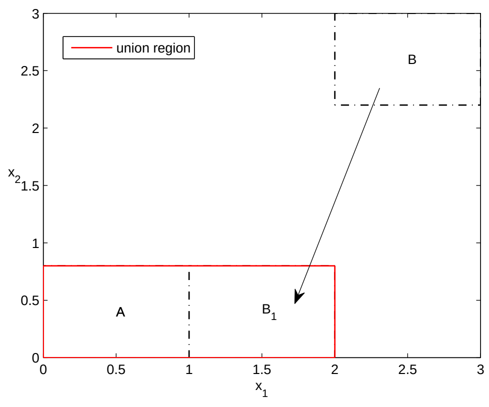


Figure 5.5: Uniform sampling within A and B.

Consider now a more general case of several hyperrectangles with arbitrary sizes and intersections. We consider a 2-dimensional example shown in Figure 5.6, where the magenta line encompasses the search region. No technique is found in the literature on

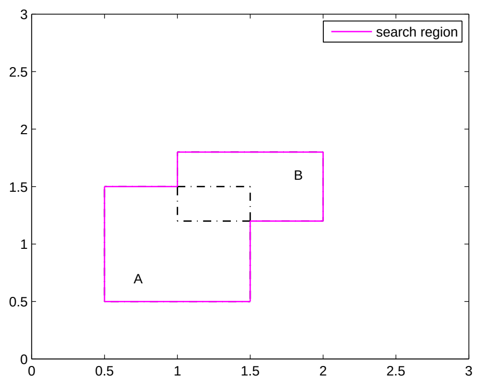


Figure 5.6: The rectangles A and B are interconnected.

the realization of uniform sampling within an irregular region. In our simulation, we use a simple substitute to approximate the uniform sampling within such a search region. We take $N^{(j)}$ samples uniformly at random within the jth hyperrectangle, where

$$N^{(j)} = \frac{V^{(j)}}{\sum V^{(j)}} N,$$

and $V^{(j)}$ is the volume of the jth hyperrectangle.

When the feasible region is irregular (not hyperrectangular), the search region being the intersection of it with hyperrectangles may also be irregular. Uniform sampling in such a region is even harder than the previous case. We adopt the rejection sampling technique [96] to approximate the uniform sampling within an irregular shaped region. We take samples uniformly at random within the hyperrectangles and discard samples outside the feasible region. This is illustrated in Figure 5.7 by a 2-dimensional case, where the blue dot line encompasses the feasible region, the black dash-dot line circumscribes a neighboring square, the magenta line encloses the search region, and the reserved and rejected samples are represented by green plus signs and red crosses, respectively.

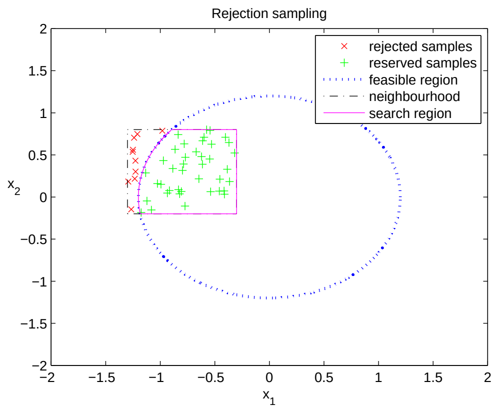


Figure 5.7: Rejection sampling.

5.5 Tuning Parameters

The proposed method has three parameters: the population size N, the volume shrinkage ratio ν and the number of retained samples M. Usually, a large N is preferred for large-scale problems. A large ν favors exploration and thus a slow convergence speed while a small ν endorses exploitation and thus a fast convergence speed. A large M is favored for complex multimodal functions (containing many local optima) and a small M is sufficient for unimodal functions or simple multimodal functions.

To compare the influence of different sets of parameters on the results of optimization, we enumerated and tested all parameter combinations from $N = \{200, 500, 1000, 2000, 5000\}$, $\nu = \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$ and $M = \{1, 2, 5, 10, 20, 50, 100\}$ on the popular benchmark problems, which are widely used in literature [159–161]. Our extensive simulation shows that the influence on the results with different values of parameters does not make great difference, but the set $\{N = 500, \nu = 0.1, M = 5\}$ has a good overall performance for low-dimensional examples ($n < 10$) and $\{N = 500, \nu = 0.5, M = 50\}$ is generally good for high-dimensional problems ($n \geq 10$). Hence, these parameter settings are suggested for use.

5.6 Convergence Analysis

The convergence of the proposed method is analyzed for the case of closed and bounded feasible region in this section. Consider Algorithm 5.1, when K_0 is infinite large, ε_1 and ε_2 are infinite small. Under these conditions, Algorithm 5.1 keeps running and its convergence analysis makes sense.

Recall that in the kth iteration, N points are generated in the search region and M top-ranking points are selected and denoted by $x_{(k,i)}$, $i = 1, 2, \dots, M$. Let $S_{(k+1,i)}$, $i = 1, 2, \dots, M$, be their corresponding neighborhoods, within which a new population is generated: $x_{(k+1,i)}$, $i =, 1, 2, \dots, M$. Note that the size of $S_{(k,i)}$,

$$\rho_{(k)} = \rho_{(2)} \nu^{\frac{k-2}{n}}, \quad 0 < \nu < 1, \tag{5.9}$$

reduces gradually and approaches to 0 when $k \to \infty$. Hence, after a large number of iterations, say, $k > K$, $S_{(k,i)}$, $i = 1, 2, \dots, M$, become very small. The samples within it have nearly same fitness values and their value set is denoted by $f_{S_{(k,i)}}$. For one or several particular iterations, the neighboring squares $S_{(k,i)}$, $i = 1, 2, \dots, M$, may be mutually exclusive. However, they in general cannot be always mutually exclusive for all $k \geq K$. For illustration, consider $M = 2$. Suppose that $S_{(k,1)}$ and $S_{(k,2)}$ are mutually exclusive onwards for a number of iterations. The size of $S_{(k+j,i)}$ decreases over $j = 1, 2, \dots$, it becomes much smaller relative to their distance, and $f_{S_{(k+j,1)}}$ and $f_{S_{(k+j,2)}}$ tend to have a less overlapping set. It is almost certain to have some $J > 0$ such that $f_{S_{(k+J,1)}}$ and $f_{S_{(k+J,2)}}$ differentiate from each other, that is, $\max f_{S_{(k+J,1)}} < \min f_{S_{(k+J,2)}}$. Then $x_{(k+J,1)}$ and $x_{(k+J,2)}$ will both be in $S_{(k+J,1)}$. The next search region is formed by $S_{(k+J+1,1)}$ and $S_{(k+J+1,2)}$ around or inside $S_{(k+J,1)}$. In other words, $S_{(k+J+1,1)}$ and $S_{(k+J+1,2)}$ move away from $S_{(k+J,2)}$ but together to $S_{(k+J,1)}$, causing their overlapping, that is, these two neighbors are no longer mutual exclusive for this particular iteration. We thus make the following reasonable assumption.

Assumption 5.1. *For the iteration index sequence $\{k\}$, there exist $K > 0$ and a subse-*

quence $\{k_i\}$ of $\{k\}$ with $k_1 \geq K$ such that the following probability is zero:

$$Prob\left\{S_{(k_1,1)} \cap S_{(k_1,2)} = \Phi, S_{(k_2,1)} \cap S_{(k_2,2)} = \Phi, \ldots\right\} = 0.$$

Assumption 5.1 almost always holds in practice. One seeming counter-example is shown in Figure 5.8. It looks possible that $S_{(k,1)}$ and $S_{(k,2)}$ keep on two arms of the

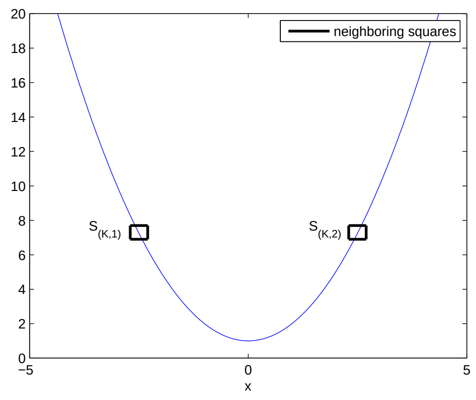


Figure 5.8: Counter example.

parabola for all $k \geq K$. This requires the best two samples of N ones to be always separate in the above two arms, which is unlikely to hold forever, since two best samples are likely to be biased to one arm once over a sufficient number of iterations under the randomized sampling at each iteration. As verification, we run 10000 times of simulation of the above case with our algorithm under $K_0 = 2\ million$. No run gave $S_{(k,1)} \cap S_{(k,2)} = \Phi$ for the 2nd million of iterations.

The only real exception to Assumption 5.1, which we can think of is that the objective function is constant locally, where the fitness values of all the samples from one population onwards are same and best. In this case, $x_{(k,i)}$, $i = 1, 2, \ldots, M$, are arbitrarily assigned and so are $S_{(k+1,i)}$, $i = 1, 2, \ldots, M$. Then, $x_{(k,1)}$ can keep jumping over M areas without convergence while $S_{(k+1,1)}$ and $S_{(k+1,2)}$ can keep mutually exclusive. This will fail Assumption 5.1. Other optimization algorithms in the literature will encounter the same problem for such a case.

Theorem 5.2. *Under Assumption 5.1, the sequence* $\left\{x_{(k,1)}\right\}$ *generated by Algorithm 5.1 is convergent with probability 1.*

Proof. Note that the feasible region is closed and bounded. According to the Bolzano-Weierstrass theorem [162], the feasible region is sequentially compact. Then, there exists a convergent subsequence $\{x_{(k_i,1)}\}$ of $\{x_{(k,1)}\}$, $i = 1, 2, \ldots$. Let $x_{(k_i,1)}$ converge to $\bar{x}_{(1)}$, i.e., $x_{(k_i,1)} \to \bar{x}_{(1)}$ as $i \to \infty$. Take a subsequence of $\{x_{(k,2)}\}$ by letting $k = k_i$: $\{x_{(k_i,2)}\}$. For this sequence, $\{x_{(k_i,2)}\}$, there similarly exists a convergent subsequence $\{x_{(k_{i_j},2)}\}$, that is, $x_{(k_{i_j},2)} \to \bar{x}_{(2)}$ as $j \to \infty$. Note that k_{i_j} is an infinite integer subset of the infinite integer set of k_i. If we take a subsequence of the convergent sequence $\{x_{(k_i,1)}\}$ as $\{x_{(k_{i_j},1)}\}$, then it must converge to the same limit as its mother sequence, that is, $x_{(k_{i_j},1)} \to \bar{x}_{(1)}$. Thus, we have

$$x_{(k_{i_j},1)} \to \bar{x}_{(1)}, \quad \text{as} \quad j \to \infty,$$

$$x_{(k_{i_j},2)} \to \bar{x}_{(2)}, \quad \text{as} \quad j \to \infty.$$

For ease of exposition, use l in place of k_{i_j}, giving

$$x_{(l,1)} \to \bar{x}_{(1)}, \quad \text{as} \quad l \to \infty,$$

$$x_{(l,2)} \to \bar{x}_{(2)}, \quad \text{as} \quad l \to \infty.$$

We now show $\bar{x}_{(1)} = \bar{x}_{(2)}$ with probability 1 by contradiction, that is, the probability of $\bar{x}_{(1)} \neq \bar{x}_{(2)}$ is zero under Assumption 5.1. Suppose $\bar{x}_{(1)} \neq \bar{x}_{(2)}$. It follows that $\left\|\bar{x}_{(1)} - \bar{x}_{(2)}\right\| = \alpha > 0$, where α is a constant. Since $x_{(l,1)}$ and $x_{(l,2)}$ converge to $\bar{x}_{(1)}$ and

$\bar{x}_{(2)}$, respectively, there are K_1 and K_2 for $\varepsilon_1 = \frac{3\alpha}{8}$ such that

$$\begin{aligned} ||x_{(l,1)} - \bar{x}_{(1)}|| &< \varepsilon_1 = \tfrac{3\alpha}{8}, \quad l > K_1 \\ ||x_{(l,2)} - \bar{x}_{(2)}|| &< \varepsilon_1 = \tfrac{3\alpha}{8}, \quad l > K_2. \end{aligned} \tag{5.10}$$

Let $K = \max\{K_1, K_2\}$. It follows from (5.9) that $\rho_{(l)} \to 0$ as $l \to \infty$, there is $K' > K$ for $\varepsilon_2 = \frac{\alpha}{8\sqrt{n}}$ such that

$$\rho_{(l+1)} < \varepsilon_2 = \frac{\alpha}{8\sqrt{n}}, \quad l > K'. \tag{5.11}$$

The equations (5.10) and (5.11) together implies

$$S_{(l+1,1)} \cap S_{(l+1,2)} = \Phi, \quad l > K', \tag{5.12}$$

and see Figure 5.9 for illustration, where the regions A and B are disconnected. Invoking

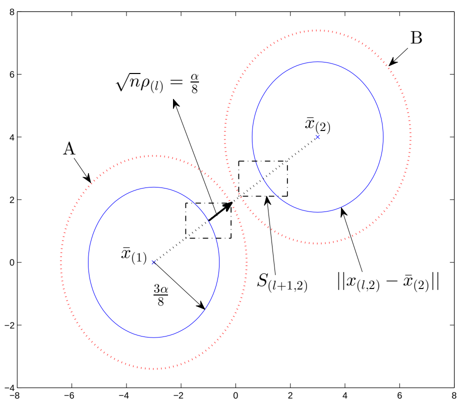

B
$\sqrt{n}\rho_{(l)} = \frac{\alpha}{8}$
$\bar{x}_{(2)}$
A
$\bar{x}_{(1)}$
$\frac{3\alpha}{8}$
$S_{(l+1,2)}$
$||x_{(l,2)} - \bar{x}_{(2)}||$

Figure 5.9: Disconnectivity of A and B.

Assumption 5.1, the probability for (5.12) is zero.

Apply the above argument to $\{x_{(k,1)}\}, \{x_{(k,2)}\}, \ldots, \{x_{(k,i)}\}$ for each i, $i = 3, 4, \ldots, M$, one by one. More specifically, use the above $\{l\}$ in $\{x_{(k,3)}\}$ to create its subsequence, $\{x_{(l,3)}\}$, and then produce a common index series $\{l'\}$ such that subsequences of $\{x_{(k,i)}\}$:

$\{x_{(l',i)}\}$, $i = 1, 2, 3$, converge to a common limit. By induction on i, we can have a common index series $\{l\}$ such that with probability 1, there holds

$$x_{(l,i)} \to \bar{x}, \quad \text{as} \quad l \to \infty, \quad i = 1, 2, \ldots, M. \tag{5.13}$$

We now show that the original sequence $\{x_{(k,1)}\}$ converges to $\bar{x}$ as well, that is, for any $\varepsilon > 0$, there is an integer K such that

$$\left\|x_{(k,1)} - \bar{x}\right\| < \varepsilon, \quad k > K, \tag{5.14}$$

provided that there are convergent subsequences $\{x_{(k_u,i)}\}$, $i = 1, 2, \ldots, M$, with a common index $k_u = l$ satisfying (5.13). It follows from (5.13) for $k_u = l$ that for $\varepsilon_1 = \frac{\varepsilon}{2}$, there is $K_1 > 0$ such that for any $k_u > K_1$

$$\left\|x_{(k_u,i)} - \bar{x}\right\| < \varepsilon_1, \quad i = 1, 2, \ldots, M. \tag{5.15}$$

In addition, it follows from (5.9) that there is $K_2 > 0$ such that for any $k_u > K_2$, $\rho_{(k_u)} < \frac{\varepsilon}{2\sqrt{n}}(1 - \nu^{\frac{1}{n}})$. Then, we have

$$\begin{aligned}
\sum_{j=k_u+1}^{k_{u+1}-1} \rho_{(j)} &= \rho_{(k_u)}(\nu^{\frac{1}{n}} + \nu^{\frac{2}{n}} + \ldots + \nu^{\frac{k_{u+1}-k_u-1}{n}}) \\
&= \rho_{(k_u)} \frac{1-\nu^{\frac{k_{u+1}-k_u-1}{n}}}{1-\nu^{\frac{1}{n}}} \\
&\leq \rho_{(k_u)} \frac{1}{1-\nu^{\frac{1}{n}}} \\
&< \frac{\varepsilon}{2\sqrt{n}}.
\end{aligned}$$

Let $K = \max\{K_1, K_2\}$. There is $K' > K$ such that

$$R_{k_u+1} = \bigcup_{i=1}^{M} S_{(k_u+1,i)} \subset B\left(\bar{x}, \varepsilon_2\right), \quad k_u > K',$$

where $B(\bar{x}, \delta) = \{x : \|x - \bar{x}\| < \delta\}$ and $\varepsilon_2 = \varepsilon_1 + \sqrt{n}\rho_{(k_u+1)}$, and see Figure 5.10 for illustration. Our construction of $S_{(k,i)}$ with exponential contracting size over k, $k =$

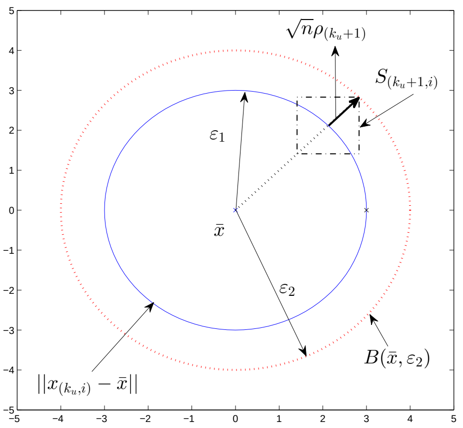


Figure 5.10: $R_{k_u+1} \subset B(\bar{x}, \varepsilon_2)$.

$k_u + 1, k_u + 2, \ldots, k_{u+1} - 1$, yields

$$R_k \subset B(\bar{x}, \varepsilon_3), \quad k = k_u + 1, k_u + 2, \ldots, k_{u+1} - 1, \tag{5.16}$$

where $\varepsilon_3 = \varepsilon_1 + \sum\limits_{j=k_u+1}^{k_{u+1}-1} \sqrt{n}\rho_{(j)} = \varepsilon_1 + \sqrt{n} \sum\limits_{j=k_u+1}^{k_{u+1}-1} \rho_{(j)} < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon$. It implies that for each population, $k = k_u + 1, k_u + 2, \ldots, k_{u+1} - 1$, its N samples $x_{(k,i)}$ satisfy

$$\left\| x_{(k,i)} - \bar{x} \right\| < \varepsilon, \quad \text{for} \quad k = k_u + 1, k_u + 2, \ldots, k_{u+1} - 1, \quad i = 1, 2, \ldots, M. \tag{5.17}$$

Then, (5.15) and (5.17) together shows that (5.14) holds for $k = k_u, k_u + 1, \ldots, k_{u+1} - 1$. For k_{u+1}, it follows from (5.13) again that (5.15) holds for $k_u \Rightarrow k_{u+1}$. By the above argument, (5.17) holds for $k = k_{u+1} + 1, k_{u+1} + 2, \ldots, k_{u+2} - 1$. Then, (5.14) holds for $k_{u+1}, k_{u+1} + 1, k_{u+1} + 2, \ldots, k_{u+2} - 1$. By the induction, (5.15) and (5.17) hold for all k_{u+v}, $v = 0, 1, 2, \ldots$, implying (5.14) with $K = k_u$. $\square$

Remark 5.1. In all our simulations, $\{x_{(k,1)}\}$ showed convergence.

Assumption 5.2. *The distance between any two local optima is strictly positive.*

Assumption 5.3. *There is $K > 0$ such that $p_k > 0$ for all $k \geq K$, where $p_k = Prob\{x^* \in S_{(k,1)}\}$ and x^* is a local optimum.*

Theorem 5.3. *Under Assumptions 5.1 - 5.3, the sequence $\{x_{(k,1)}\}$ generated by Algorithm 5.1 converges to a local optimum x^* with probability 1.*

Proof. Let α be the minimum of distances of any two local optima. It follows from (5.16) in the proof of Theorem 5.2 that the size (ε_3) of R_k converges to zero under Assumption 5.1. Thus, given $\varepsilon < \frac{\alpha}{2}$, there is $K > 0$ such that the size of R_k is less than ε, and it can contain at most one optimum. By Theorem 5.2, $\{x_{(k,1)}\}$ converges to $\bar{x}$ with probability 1. Let x^* be a local optimum. We prove $\bar{x} = x^*$ by contradiction. If there is no optimum in R_k, then $p_k = 0$ for $k = K, K+1, \ldots$, which contradicts Assumption 5.3. Otherwise, there is one optimum in R_k. Let $\beta = \|\bar{x} - x^*\|$. For $\varepsilon < \frac{\beta}{2}$, there is $K > 0$ such that $R_k \in B(\bar{x}, \varepsilon)$, which is mutually exclusive with $B(x^*, \varepsilon)$. Then, $p_k = 0$ for $k = K, K+1, \ldots$, which contradicts Assumption 5.3. □

The probability $p_k = 1$ is a special case of Assumption 5.3. It means that x^* is in every $S_{(k,1)}$, $k \geq K$. Obviously, $\{x_{(k,1)}\}$ converges to x^*.

Corollary 5.1. *Under Assumptions 5.1 - 5.2, the sequence $\{x_{(k,1)}\}$ generated by Algorithm 5.1 converges to a local optimum x^* with probability 1 if $p_k = 1$.*

Remark 5.2. If x^* is sampled at Kth iteration, where R_k contains one optimum for any $k \geq K$, x^* will be forever retained, which is equivalent to $p_k = 1$ for $k \geq K$, and $\{x_{(k,1)}\}$ converges to x^*.

Finally, consider briefly the convergence of Algorithm 5.2. Algorithm 5.2 calls Algorithm 5.1 infinitely. We consider the case that K_0 is finite large, ε_1 and ε_2 are finite small for Algorithm 5.1. Under Assumptions 5.1 - 5.3, Algorithm 5.1 terminates when it finds a solution very near some local optimum. A small neighborhood of the found solution, which contains this local optimum, is then removed from the feasible region. Suppose that the number of local optima in the feasible region is limited, which is likely to hold for most applications. Then, Algorithm 5.2 will traverse all local optima and find the global one.

5.7 Simulations

In order to test the performance such as applicability, accuracy and speed of the proposed method, Algorithm 5.1 is applied to both low-dimensional and high-dimensional benchmark problems (Algorithm 5.2 cannot be examined by simulation since it takes an infinite long time). It is programmed in MATLAB R2013a and run on a 64-bit Windows 7 system with 3.40GHz Inter Core i5 processor and 8 GB RAM.

5.7.1 Low-Dimensional Examples

The low-dimensional problems are taken from [159, 160], which include five test functions with different properties as shown in Table 5.2. The initial search region for each unconstrained problem is a region bounded by $[L, U]$. For the constrained G8 problem, the initial search region is its feasible region. Jamil et al. [159] and Hedar [160] have taken these problems as benchmarks for comparing global optimization methods. These

Table 5.2: Properties of low-dimensional test functions

Test function	n	$[L, U]$	Modality	Continuity	Differentiability	Constraint
Booth	2	$[-10, 10]$	unimodal	continuous	differentiable	unconstrained
Goldstein	2	$[-2, 2]$	multimodal	continuous	differentiable	unconstrained
Hartman3	3	$[0, 1]$	multimodal	continuous	differentiable	unconstrained
Tripod	2	$[-100, 100]$	multimodal	discontinuous	non-differentiable	unconstrained
G8 problem	2	-	multimodal	continuous	differentiable	constrained

problems are not chosen in favor of our method.

Let $N = 500, \nu = 0.1, M = 5, K_0 = 100$ and $\varepsilon_1 = \varepsilon_2 = 10^{-8}$. We run Algorithm 5.1 on each problem to get an accuracy $|f(x^*) - f_{opt}(x)|$ and a running time, where $f(x^*)$ is the minimum value found by our algorithm and $f_{opt}(x)$ is the global minimum in theory. The result of a single run has some randomness. Therefore, we repeat the process for 51 times to get the average accuracy and average running time. Average results are compared with those with existing popular algorithms and shown in Table 5.3, where "-" means that a certain algorithm cannot be applied to a given problem using off-the-shelf MATLAB toolboxes [153]. These algorithms are selected for comparison algorithms because they are classical, mature and widely adopted to solve global optimization problems [67, 70]. The best results among all methods are highlighted in bold. As can be seen from Table 5.3, the proposed method applies to all these problems. Generally, the proposed method is very effective for all these low-dimensional problems. It gets the most accurate solutions among all methods for Goldstein, Tripod and G8 problem. For Booth and Hartman3, our results are as accurate as the best ones by other methods. Our method has relatively short running times.

Table 5.3: Average accuracy and running time for low-dimensional problems

Test function	Pattern Search		Genetic Algorithm		Simulated Annealing		Particle Swarm Optimization		Proposed Method	
	Accuracy	Time (s)	Accuracy	Time (s)	Accuracy	Time (s)	Accuracy	Time (s)	Accuracy	Time (s)
Booth	1.26e-10	0.0526	7.58e-07	0.0836	7.81e-05	0.573	**3.91e-11**	0.152	3.33e-10	**0.0109**
Goldstein	7.39e-10	0.0931	14.78	0.0903	2.59e-07	0.565	61.94	0.175	**4.62e-10**	**0.0161**
Hartman3	**1.79e-05**	**0.0586**	0.062	0.103	3.69	0.492	12.13	0.112	3.01e-05	0.102
Tripod	2.00	**0.0311**	36.30	0.141	0.35	0.547	0.96	0.232	**0.0392**	0.0418
G8 problem	0.096	**0.0290**	0.016	0.785	-	-	-	-	**2.50e-05**	0.0539

5.7.2 High-Dimensional Examples

We take the CEC2013 testbed [161] as our high-dimensional problems, as it is well recognized as a benchmark set of optimization problems. It includes 28 numerical test functions, where $f1 \sim f5$ are unimodal functions, $f6 \sim f20$ are multimodal functions and $f21 \sim f28$ are composition functions. We test these functions at three problem dimensionality, $n = 10, 30$ and 50, respectively. For each of these functions, the initial search region is a hypercube bounded by $[-100, 100]$. Let $N = 500, \nu = 0.5, M = 50$, $K_0 = 20n$ and $\varepsilon_1 = \varepsilon_2 = 10^{-8}$. We run our method on each problem for 51 times to record the best, worst, median, mean, and standard deviation of the accuracy. Results are shown in Tables 5.4, 5.5 and 5.6 for $n = 10, 30, 50$, respectively. Very accurate solutions are highlighted in bold. As can be observed from Table 5.4 - 5.6, the proposed method is able to find the approximate global minimal values for many of the 28 test functions such as $f1$, $f3$, $f5$, $f6$, $f7$, $f9$, $f10$, $f16$ for $n = 10$ and $f1$, $f5$, $f10$, $f16$ for $n = 30, 50$. It outperforms most of the selected algorithms [163] of CEC2013 on $f7$, $f9$, $f12$, $f13$, $f15$, $f16$, $f18$, $f20$, $f23$ and $f25$. Like other methods [164, 165], our method encounters difficulties for the composition functions which are characterized as difficult problems [161].

In regard to the complexity, we follow its definition in [161]. Let $T0$ be the time of running a standard test program which performs some simple mathematical operations, $T1$ be the time of executing 200,000 evaluations of $f14$ for a certain dimension, $T2$ be the complete computation time for the algorithm with 200,000 evaluations of the same dimensional $f14$ and $\hat{T}2$ is the average of $T2$ (over 5 runs). Then, $(\hat{T}2 - T1)/(T0)$

Table 5.4: Accuracy for $n = 10$

Func.	Best	Worst	Median	Mean	Std.
f1	**2.01e-09**	1.00e-08	4.97e-09	5.08e-09	1.92e-09
f2	338.55	5.51e+05	4.10e+04	6.57e+04	8.86e+04
f3	**2.49e-04**	8.94	0.084	0.29	1.24
f4	3031.19	1.90e+04	8862.60	9111.58	3271.10
f5	**2.73e-05**	1.15	3.36e-04	0.023	0.16
f6	**0.019**	9.82	0.35	1.19	2.62
f7	**1.42e-08**	2.06e-04	2.38e-08	1.33e-05	4.76e-05
f8	20.00	20.46	20.30	20.29	0.12
f9	**1.34e-06**	1.53	3.12e-06	0.26	0.44
f10	**4.59e-09**	0.064	0.032	0.032	0.018
f11	1.99	10.94	4.97	5.01	1.79
f12	0.99	7.96	3.98	4.23	1.53
f13	0.99	17.48	8.09	8.17	3.73
f14	3.66	635.39	150.06	167.74	140.60
f15	3.66	296.64	151.44	145.20	78.32
f16	**3.88e-08**	0.020	3.53e-06	0.0042	0.0068
f17	5.56	18.43	14.93	14.72	2.15
f18	2.84	19.70	14.20	13.91	3.33
f19	0.099	1.02	0.66	0.64	0.18
f20	0.43	3.42	1.57	1.68	0.67
f21	200.00	400.19	400.19	392.34	39.25
f22	27.89	457.66	173.01	178.07	104.63
f23	18.07	399.38	81.35	103.39	82.41
f24	104.88	200.00	200.00	190.75	28.33
f25	103.74	200.00	200.00	194.39	22.67
f26	101.99	200.02	103.98	106.35	13.49
f27	300.00	400.00	300.00	335.29	48.26
f28	100.00	300.00	300.00	237.25	93.72

Table 5.5: Accuracy for $n = 30$

Func.	Best	Worst	Median	Mean	Std.
f1	**1.63e-08**	3.93e-08	2.83e-08	2.82e-08	5.96e-09
f2	3.32e+05	5.69e+06	1.51e+06	1.78e+06	1.07e+06
f3	6.58	5.63e+06	1.74e+05	1.02e+06	1.55e+06
f4	2.10e+04	5.26e+04	3.41e+04	3.36e+04	7946.14
f5	**0.0012**	58.85	12.56	18.13	16.72
f6	14.19	78.75	15.31	24.57	20.92
f7	0.058	8.40	1.26	2.00	2.06
f8	20.80	21.04	20.96	20.95	0.057
f9	2.88	9.48	6.72	6.77	1.48
f10	**1.45e-08**	0.034	0.0074	0.0086	0.0088
f11	16.91	50.74	32.83	32.83	7.15
f12	20.89	47.76	31.84	32.83	6.24
f13	23.31	117.07	66.38	67.94	21.46
f14	762.17	2194.69	1486.47	1493.16	345.97
f15	723.64	2224.89	1437.00	1487.09	332.25
f16	**0.0077**	0.032	0.015	0.016	0.0052
f17	49.07	89.87	64.52	65.30	7.48
f18	49.85	81.76	65.71	66.25	7.61
f19	2.05	4.40	2.80	2.95	0.61
f20	15.00	15.00	15.00	15.00	3.70e-05
f21	200.00	443.54	300.00	336.43	101.55
f22	487.76	2444.88	1557.59	1522.39	426.52
f23	939.93	2804.85	1649.38	1640.93	411.30
f24	200.15	218.96	202.53	203.99	4.26
f25	200.02	256.70	236.35	222.87	22.73
f26	200.01	200.19	200.07	200.08	0.034
f27	306.23	553.05	375.46	380.23	44.70
f28	300.00	300.00	300.00	300.00	6.00e-08

Table 5.6: Accuracy for $n = 50$

Func.	Best	Worst	Median	Mean	Std.
f1	**3.36e-08**	1.19e-07	5.75e-08	6.24e-08	1.79e-08
f2	1.22e+06	7.19e+06	2.79e+06	3.13e+06	1.20e+06
f3	8.13e+04	1.22e+08	2.09e+07	2.99e+07	2.90e+07
f4	3.13e+04	5.75e+04	4.21e+04	4.30e+04	6630.71
f5	**0.0032**	94.94	45.27	49.43	23.38
f6	43.45	48.22	45.09	44.99	1.05
f7	2.86	23.34	10.51	11.27	4.69
f8	21.00	21.19	21.13	21.13	0.041
f9	10.15	22.24	15.89	16.11	2.79
f10	**5.28e-08**	0.10	0.027	0.032	0.021
f11	48.75	110.44	76.61	77.00	11.72
f12	55.72	110.44	78.60	79.34	12.63
f13	95.96	262.36	169.10	171.45	34.99
f14	1700.11	5082.22	3354.87	3349.36	697.86
f15	2532.21	5720.08	3779.75	3821.00	508.17
f16	**0.0096**	0.044	0.023	0.022	0.0075
f17	105.08	161.21	135.58	134.93	14.18
f18	103.83	158.24	134.07	132.52	11.61
f19	3.19	8.23	5.34	5.46	1.10
f20	16.80	25.00	25.00	24.57	1.76
f21	200.00	1122.19	200.00	573.61	397.66
f22	2368.39	5531.07	3598.38	3580.49	682.76
f23	2787.25	5218.98	4127.58	4138.58	548.57
f24	210.10	246.93	226.47	227.33	8.89
f25	273.48	310.18	289.49	288.82	8.86
f26	200.26	356.66	330.32	294.30	64.58
f27	484.54	909.06	726.01	713.64	90.85
f28	400.00	400.00	400.00	400.00	5.17e-07

reflects the complexity of the algorithm. Results calculated on $n = 10, 30, 50$ are shown in Table 5.7. According to Table 5.7, the complexity of the proposed method grows

Table 5.7: Complexity

n	$T0$ (s)	$T1$ (s)	$\hat{T}2$ (s)	$(\hat{T}2 - T1)/(T0)$
10		1.33	5.25	35.64
30	0.11	1.89	13.14	102.27
50		2.52	20.89	167.00

nearly linearly with n as expected.

5.8 Conclusions

In this chapter, we have proposed a new global optimization method based on randomized group search in contracting regions. The samples in each population are always randomly generated in the search region. A group of good samples are retained for elitism instead of the best one. which avoids premature convergence to a local optimum. The search region is limited around the retained samples and reduced over iterations as better solutions are obtained. The reduction of search region over iterations guarantees the convergence, and differentiates our method from existing methods which all operate on the fixed region. The proposed method is fast and easy to implement. Moreover, the proposed method is shown to be very effective for low-dimensional benchmark problems and work reasonably well on high-dimensional ones. In short, this is a very different method for global optimization with simplicity, clarity, efficiency and effectiveness.

There could be a few directions for future research. Efficient methods on uniform sampling in irregular regions are in high demand. Adaptive schemes which adjust three tuning parameters instead of constant settings may be developed to improve the performance of the proposed method. Our method may be combined with other methods to complement each other.

Chapter 6

Determining Stabilizing Parameter Regions for General Delay Control Systems

6.1 Introduction

The previous chapter presents a new global optimization method which helps with model based designs. Model based designs play a vitally important role in different fields. In control engineering, information retrieved from a plant is used to design control systems to meet the needs of real applications. Finding stabilizing regions for control systems in parameter space is important for controller tuning or controller optimization. The existing approaches such as [83, 84, 89, 92, 93] in the literature seek the solutions for the stabilizing parameter regions for limited classes of plants or controllers.

In this chapter, we design a general algorithm for determining stabilizing parameter regions for delay control systems based on randomized sampling. Each unknown parameter is assumed to follow the uniform distribution in a given range and a certain number of random sample points are generated in the parameter space. Next, given a delay control system, we convert it into a unified state-space form. Efficient LMI stability criterion is developed for a control system with multiple delays in both input and state. Then each point in the parameter space is checked by the developed stability criterion. After that, these points are separated into stable and unstable regions by the decision function obtained from some learning method. The effectiveness of the proposed method is illustrated by simulation examples.

The rest of this chapter is organized as follows. Section 6.2 presents the idea of proposed method. Section 6.3 develops the stability criterion. Determining stabilizing parameter regions is discussed in Section 6.4. Section 6.5 gives simulation examples and Section 6.6 concludes the chapter.

6.2 The Proposed Method

We consider a unity feedback control system as shown in Figure 6.1. The plant may have

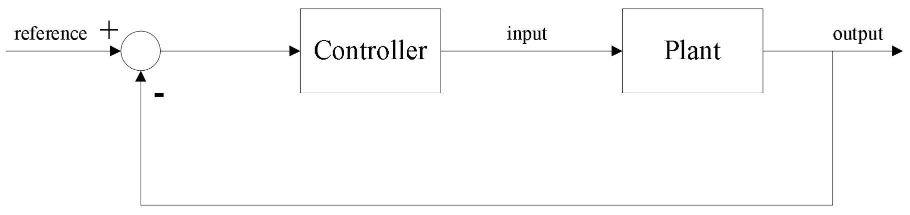


Figure 6.1: Unity feedback control system.

some unknown parameters that may affect the system stability and the parameters of the controller are also needed to be designed. Hence, knowing stabilizing parameter regions is instructive for robustness analysis and design. Some methods [84–86] can give analytical solutions for stabilizing parameter regions, but these methods usually have many constraints and could only be applied to limited plants or controllers. Some numerical methods [93, 94] also have some restrictions on system structures and their algorithms might be difficult to be implemented. The objective of this chapter is to provide stabilizing parameter regions with a new approach which is totally different from the existing methods in this specific area. We illustrate the idea of our method with a simple example.

We consider the model in [92] as follows,

$$G(s) = \frac{s^3 + 4s^2 - s + 1}{s^5 + 2s^4 + 32s^3 + 14s^2 - 4s + 50},$$

with a PI controller

$$C(s) = K_p + \frac{K_i}{s},$$

where K_p and K_i are unknown parameters. With the method in [92], the stabilizing parameter region is shown in Figure 6.2a.

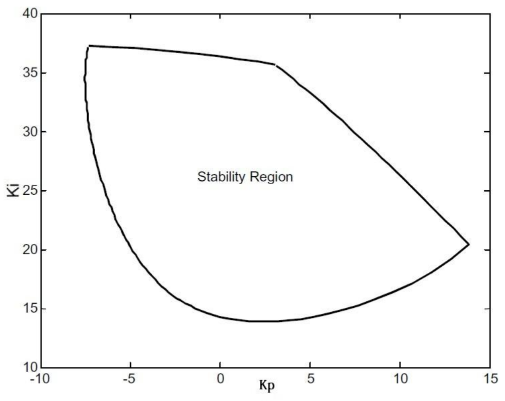
We employ the idea of randomized sampling. Suppose that each unknown parameter follows the uniform distribution in a given range, that is, $K_p \in [-10, 15]$, $K_i \in [10, 40]$ and they distribute uniformly in their respective range. Then a certain number of random points are sampled in the parameter space. Throughout this chapter, $N = 5000$ is used for all simulation cases.

Next, we check whether each of these points could stabilize the system by some stability criterion. The characteristic equation of the closed-loop system is

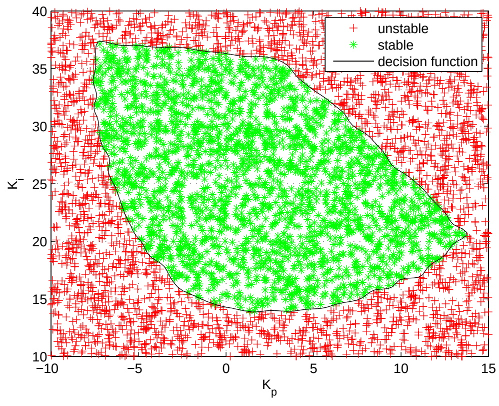
$$s^6 + 2s^5 + (K_p + 3)s^4 + (4K_p + K_i + 14)s^3$$
$$+(4K_i - K_p - 4)s^2 + (K_p - K_i + 50)s + K_i = 0.$$

We can simply calculate the closed-loop poles for stability testing. If a point of $[K_p, K_i]$ could stabilize the system, it is labeled as 'stable'. Otherwise, if a point could not stabilize the system, it is labeled as 'unstable'. However, calculating the closed-loop poles is not possible for systems with time delays. In this case, we present a Linear Matrix Inequality (LMI) stability criterion which will be discussed in next section.

Lastly, the points in the parameter space are divided into stable and unstable regions by the decision function obtained from some learning method, such as the Neural Networks and the Support Vector Machines (SVM) [2]. We choose SVM as the classification tool and employ the LibSVM [112] kit with its arguments '-t'=2 (Radial Basis Function (RBF) as kernel) and '-c'=1000000 (penalty parameter) to solve the problem. The resulting stabilizing parameter region is shown in Figure 6.2b. It is seen from Figure 6.2a



(a) Result in [92].



(b) Result with the proposed method.

Figure 6.2: Stabilizing parameter region for the illustrative example.

and Figure 6.2b that the stable region from the proposed method is almost same as that in [92]. Hence, our method is effective and straightforward.

6.3 Stability Criterion

As stated in previous section, it is impossible to calculate the closed-loop poles for systems with time delays. Therefore, in this section, we present an effective algorithm for stability testing which can be applied to a much wider range of systems. Given a delay system with PI or PID controller, we first convert it into a unified state-space form, which is a generalization of the method in [166] where a delay-free system is considered. Next, we present a conversion of delay systems with general dynamic controllers. Lastly, we present an LMI stability criterion for the unified state-space form.

6.3.1 PI Control for Input-Delay Plant

Consider a plant:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t-d), \\ y(t) = Cx(t), \end{cases} \tag{6.1}$$

with a PI controller:

$$u(t) = F_1 y(t) + F_2 \int_0^t y(\tau)d\tau.$$

Let

$$z(t) = \begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix} = \begin{bmatrix} x(t) \\ \int_0^t y(\tau)d\tau \end{bmatrix},$$

so that

$$z(t-d) = \begin{bmatrix} z_1(t-d) \\ z_2(t-d) \end{bmatrix} = \begin{bmatrix} x(t-d) \\ \int_0^{t-d} y(\tau)d\tau \end{bmatrix}.$$

The vector $z(t)$ can be viewed as a new state variable of the system, whose dynamics is governed by

$$\dot{z}(t) = \begin{bmatrix} A & 0 \\ C & 0 \end{bmatrix} z(t) + \begin{bmatrix} B \\ 0 \end{bmatrix} u(t-d), \tag{6.2}$$

where

$$\begin{aligned} u(t) &= F_1 C z_1(t) + F_2 z_2(t) \\ &= F_1 [C \quad 0] \begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix} + F_2 [0 \quad I] \begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix}. \end{aligned} \tag{6.3}$$

Let $\bar{C}_1 = [C \quad 0]$ and $\bar{C}_2 = [0 \quad I]$. Equation (6.3) can be rewritten as

$$u(t) = (F_1 \bar{C}_1 + F_2 \bar{C}_2) z(t),$$

or

$$u(t-d) = (F_1 \bar{C}_1 + F_2 \bar{C}_2) z(t-d). \tag{6.4}$$

Substituting (6.4) into (6.2) yields

$$\dot{z}(t) = \tilde{A} z(t) + \tilde{A}_1 z(t-d), \tag{6.5}$$

where

$$\tilde{A} = \begin{bmatrix} A & 0 \\ C & 0 \end{bmatrix}, \text{ and } \tilde{A}_1 = \begin{bmatrix} B \\ 0 \end{bmatrix} (F_1 \bar{C}_1 + F_2 \bar{C}_2).$$

When (6.1) is with a PID controller $u(t) = F_1 y(t) + F_2 \int_0^t y(\tau) d\tau + F_3 \frac{dy(t)}{dt}$, the conversion could not be proceeded. This is because $u(t)$ depends on $u(t-d)$ since $\frac{dy(t)}{dt} = CAx(t) + CBu(t-d)$. Then the control signal cannot be expressed only by state vectors as (6.3) or (6.4). In such a case, we could use a practical D controller:

$$\frac{s}{1 + \dfrac{s}{N_d}},$$

where N_d is chosen by users to limit derivative gain on higher frequencies. Then, the practical PID controller falls in a format of general dynamic controller, which is handled in Section 6.3.3 below.

6.3.2 PID Control for State-Delay Plant

Consider a plant:

$$\begin{cases} \dot{x}(t) = Ax(t) + A_1x(t-d) + Bu(t), \\ y(t) = Cx(t), \end{cases} \tag{6.6}$$

with a PID controller:

$$u(t) = F_1y(t) + F_2\int_0^t y(\tau)d\tau + F_3\frac{dy(t)}{dt}.$$

Let $z_1(t) = x(t)$ and $z_2(t) = \int_0^t y(\tau)d\tau$. We have

$$\dot{z}_1(t) = \dot{x}(t) = Az_1(t) + A_1z_1(t-d) + Bu(t),$$

and

$$\dot{z}_2(t) = y(t) = Cz_1(t).$$

Denoting $z(t) = [z_1^T(t), z_2^T(t)]^T$, we have

$$\dot{z}(t) = \bar{A}z(t) + \bar{A}_1z(t-d) + \bar{B}u(t),$$

where

$$\bar{A} = \begin{bmatrix} A & 0 \\ C & 0 \end{bmatrix},\ \bar{A}_1 = \begin{bmatrix} A_1 & 0 \\ 0 & 0 \end{bmatrix},\ \bar{B} = \begin{bmatrix} B \\ 0 \end{bmatrix}.$$

Combining (6.6) and the definition of z yields

$$y(t) = Cz_1(t) = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix},$$

$$\int_0^t y(\tau)d\tau = z_2(t) = \begin{bmatrix} 0 & I \end{bmatrix} \begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix},$$

and

$$\begin{aligned} \frac{dy(t)}{dt} &= C\dot{x}(t) \\ &= CAx(t) + CA_1x(t-d) + CBu(t) \\ &= [CA, 0]z(t) + [CA_1, 0]z(t-d) + CBu(t). \end{aligned}$$

Denoting $\bar{C}_1 = [C \quad 0]$, $\bar{C}_2 = [0 \quad I]$, $\bar{C}_3 = [CA \quad 0]$, $C_d = [CA_1 \quad 0]$, $\bar{y}_1(t) = \bar{C}_1z(t)$, $\bar{y}_2(t) = \bar{C}_2z(t)$, and $\bar{y}_3(t) = \bar{C}_3z(t) + C_dz(t-d)$, we have

$$u(t) = F_1\bar{y}_1(t) + F_2\bar{y}_2(t) + F_3\bar{y}_3(t) + F_3CBu(t).$$

Suppose that $(I - F_3CB)$ is invertible. Let $\bar{y}(t) = [\bar{y}_1^T(t), \bar{y}_2^T(t), \bar{y}_3^T(t)]^T$, $\bar{C} = [\bar{C}_1^T, \bar{C}_2^T, \bar{C}_3^T]^T$, $\bar{C}_d = [0, 0, C_d^T]^T$, and $\bar{F} = [\bar{F}_1, \bar{F}_2, \bar{F}_3]$, where

$$\begin{aligned} \bar{F}_1 &= (I - F_3CB)^{-1}F_1, \\ \bar{F}_2 &= (I - F_3CB)^{-1}F_2, \\ \bar{F}_3 &= (I - F_3CB)^{-1}F_3. \end{aligned}$$

Then (6.6) is equivalent to

$$\begin{cases} \dot{z}(t) = \bar{A}z(t) + \bar{A}_1z(t-d) + \bar{B}u(t), \\ \bar{y}(t) = \bar{C}z(t) + \bar{C}_dz(t-d), \end{cases}$$

with

$$u(t) = \bar{F}\,\bar{y}(t),$$

i.e.,

$$\begin{aligned}\dot{z}(t) &= \bar{A}z(t) + \bar{A}_1 z(t-d) + \bar{B}\bar{F}\bar{C}z(t) + \bar{B}\bar{F}\bar{C}_d z(t-d) \\ &= (\bar{A} + \bar{B}\bar{F}\bar{C})z(t) + (\bar{A}_1 + \bar{B}\bar{F}\bar{C}_d)z(t-d),\end{aligned} \tag{6.7}$$

which is also in the form of (6.5) with $\tilde{A} = (\bar{A} + \bar{B}\bar{F}\bar{C})$ and $\tilde{A}_1 = (\bar{A}_1 + \bar{B}\bar{F}\bar{C}_d)$.

Remark 6.1. The systems (6.1) and (6.6) only contain one time delay. However, it would not be difficult to make conversion for systems with multiple time delays, which is omitted here for brevity.

The previous two cases only tackle delay systems with PI or PID controller whose parameters appear in a linear form. In practical control systems, the controllers may be of higher orders and the parameters of controllers may also appear in a nonlinear form, such as the lead-lag compensators [167]. Thus, we consider the conversion for delay systems with general dynamic controller as follows.

6.3.3 General Dynamic Controller for a Plant with Multiple Delays in Input and State

Consider a plant (6.8)

$$\begin{cases}\dot{x}(t) &= Ax(t) + A_1x(t-d_1) + A_2x(t-d_2) + \ldots + A_hx(t-d_h) \\ & \qquad +Bu(t) + B_1u(t-d_{h+1}) + B_2u(t-d_{h+2}) + \ldots + B_lu(t-d_{h+l}), \\ y(t) &= Cx(t).\end{cases} \tag{6.8}$$

under the following dynamic controller:

$$C(s) = \frac{b_0 s^m + b_1 s^{m-1} + \cdots + b_{m-1}s + b_m}{s^n + a_1 s^{n-1} + \cdots + a_{n-1}s + a_n},$$

whose minimal state-space realization can be expressed by

$$\begin{cases} \dot{v}(t) = A_c v(t) + B_c y(t), \\ u(t) = C_c v(t) + D_c y(t). \end{cases}$$

Let $z_1(t) = x(t)$ and $z_2(t) = v(t)$. Denoting $z(t) = [z_1^T(t), z_2^T(t)]^T$, we have

$$z(t) = \begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix} = \begin{bmatrix} x(t) \\ v(t) \end{bmatrix},$$

and

$$z(t-d_i) = \begin{bmatrix} z_1(t-d_i) \\ z_2(t-d_i) \end{bmatrix} = \begin{bmatrix} x(t-d_i) \\ v(t-d_i) \end{bmatrix}.$$

Combining the above expressions gives (6.9)

$$\begin{aligned} \dot{z}_1(t) \ &= Az_1(t) + A_1 z_1(t-d_1) + \ldots + A_h z_1(t-d_h) + BD_cCz_1(t) + BC_cz_2(t) \\ &+B_1D_cCz_1(t-d_{h+1}) + B_1C_cz_2(t-d_{h+1}) + \ldots \\ &+B_lD_cCz_1(t-d_{h+l}) + B_lC_cz_2(t-d_{h+l}). \end{aligned} \tag{6.9}$$

and

$$\dot{z}_2(t) = B_cCz_1(t) + A_cz_2(t),$$

i.e.,

$$\dot{z}(t) = \tilde{A}z(t) + \sum_{i=1}^{k} \tilde{A}_i z(t-d_i), \tag{6.10}$$

where

$$\tilde{A} = \begin{bmatrix} A + BD_cC & BC_c \\ B_cC & A_c \end{bmatrix},$$

$$\tilde{A}_i = \begin{cases} \begin{bmatrix} A_i & 0 \\ 0 & 0 \end{bmatrix}, \text{ for } \quad 0 < i \leq h, \\ \begin{bmatrix} B_{i-h}D_cC & B_{i-h}C_c \\ 0 & 0 \end{bmatrix}, \text{ for } \quad h < i \leq k, \end{cases}$$

and $\quad k = h + l.$

Remark 6.2. The system (6.5) is a special case of (6.10).

6.3.4 The LMI Stability Criterion for a System with Multiple Delays in Input and State

Theorem 6.1. *The system (6.10) is asymptotically stable if there exist symmetric positive definite matrices P, Q_1, ..., Q_k and W_1, ..., W_k, such that*

$$\begin{bmatrix} \Omega & \Psi \\ * & \Lambda \end{bmatrix} < 0, \tag{6.11}$$

where (6.12) holds,

$$\Omega = \begin{bmatrix} \tilde{A}^TP + P\tilde{A} + \sum\limits_{i=1}^{k} Q_i - \sum\limits_{i=1}^{k} W_i & P\tilde{A}_1 + W_1 & P\tilde{A}_2 + W_2 & \dots & P\tilde{A}_k + W_k \\ * & -Q_1 - W_1 & 0 & \dots & 0 \\ * & * & -Q_2 - W_2 & \ddots & \vdots \\ * & * & * & \ddots & 0 \\ * & * & * & * & -Q_k - W_k \end{bmatrix}. \tag{6.12}$$

$$\Psi = \begin{bmatrix} d_1\tilde{A}^T W_1 & \dots & d_k\tilde{A}^T W_k \\ d_1\tilde{A}_1^T W_1 & \dots & d_k\tilde{A}_1^T W_k \\ \vdots & \ddots & \vdots \\ d_1\tilde{A}_k^T W_1 & \dots & d_k\tilde{A}_k^T W_k \end{bmatrix},$$

and

$$\Lambda = \begin{bmatrix} -W_1 & 0 & \dots & 0 \\ * & -W_2 & \ddots & \vdots \\ * & * & \ddots & 0 \\ * & * & * & -W_k \end{bmatrix}.$$

Here and in the sequel, a block induced by symmetry is denoted by an ellipsis $*$.

Proof. Define the Lyapunov functional as

$$\begin{aligned} V(z(t)) = \;& z^T(t)Pz(t) + \sum_{i=1}^{k}\left(\int_{t-d_i}^{t} z^T(s)Q_i z(s)ds\right) \\ & + \sum_{i=1}^{k}\left(d_i\int_{-d_i}^{0}\int_{t+\alpha}^{t} \dot{z}^T(s)W_i\dot{z}(s)dsd\alpha\right). \end{aligned}$$

The derivative of $V(z(t))$ is

$$\begin{aligned} \dot{V}(z(t)) \; &= z^T(t)P\dot{z}(t) + \dot{z}^T(t)Pz(t) \\ &\quad + \sum_{i=1}^{k}\left(z^T(t)Q_i z(t)\right) \\ &\quad - \sum_{i=1}^{k}\left(z^T(t-d_i)Q_i z(t-d_i)\right) \\ &\quad + \sum_{i=1}^{k}\left(d_i^2\dot{z}^T(t)W_i\dot{z}(t)\right) \\ &\quad - \sum_{i=1}^{k}\left(d_i\int_{t-d_i}^{t}\dot{z}^T(s)W_i\dot{z}(s)ds\right). \end{aligned}$$

It follows from Jensen's inequality [168] that

$$-d_i\int_{t-d_i}^{t}\dot{z}^T(s)W_i\dot{z}(s)ds \le -\left[z(t)-z(t-d_i)\right]^T W_i\left[z(t)-z(t-d_i)\right].$$

Then we have (6.13).

$$\begin{aligned}\dot{V}(z(t)) &\leq z^T(t)P\left[\tilde{A}z(t)+\sum_{i=1}^{k}\left(\tilde{A}_i z(t-d_i)\right)\right]+\left[\tilde{A}z(t)+\sum_{i=1}^{k}\left(\tilde{A}_i z(t-d_i)\right)\right]^T Pz(t)\\ &+\sum_{i=1}^{k}\left(z^T(t)Q_i z(t)\right)-\sum_{i=1}^{k}\left(z^T(t-d_i)Q_i z(t-d_i)\right)\\ &+\sum_{i=1}^{k}\left\{d_i^2\left[\tilde{A}z(t)+\sum_{i=1}^{k}\left(\tilde{A}_i z(t-d_i)\right)\right]^T W_i\left[\tilde{A}z(t)+\sum_{i=1}^{k}\left(\tilde{A}_i z(t-d_i)\right)\right]\right\}\\ &-\sum_{i=1}^{k}\left\{\left[z(t)-z(t-d_i)\right]^T W_i\left[z(t)-z(t-d_i)\right]\right\}.\end{aligned} \tag{6.13}$$

Let

$$w(t)=\begin{bmatrix} z^T(t) & z^T(t-d_1) & \cdots & z^T(t-d_k)\end{bmatrix}^T,$$

and

$$\Gamma=\begin{bmatrix}\tilde{A} & \tilde{A}_1 & \cdots & \tilde{A}_k\end{bmatrix}.$$

One sees

$$\dot{V}(z(t))\leq w^T(t)\left[\Omega+\sum_{i=1}^{k}\left(d_i^2\Gamma^T W_i\Gamma\right)\right]w(t).$$

By Schur complement, (6.11) guarantees

$$\left[\Omega+\sum_{i=1}^{k}\left(d_i^2\Gamma^T W_i\Gamma\right)\right]<0.$$

Therefore, the system (6.10) is asymptotically stable. □

6.4 Stabilizing Parameter Regions

Each point in the parameter space corresponds to a sample of the parameter vector p, which is denoted by $p_i,\ i=1\ldots N$. We check whether each of these points could stabilize the system by the developed LMI stability criterion. If a point p_i could stabilize the system, it is labeled as 'stable'. Otherwise, if p_i could not stabilize the system, it is labeled as 'unstable'.

The points in the parameter space can be separated into stable and unstable regions by the decision function obtained from some learning method. In this chapter, we choose SVM as the learning method due to its superior performance in a wide range of applications. Support Vector Machines (SVM), which was first introduced by Vapnik [169], has shown many attractive features in the fields of small sample, non-linear and high dimensional pattern recognition [106]. It can be promoted to classification and regression problems. It employs the Structural Risk Minimization principle [106]. The goal of SVM is to find a decision function that minimizes the structural risk, which could be converted into a quadratic programming problem. In addition, the solution of an SVM problem is a globally optimal solution [170].

In this chapter, SVM is employed to solve a binary classification problem. Given the data set $\mathbf{S} = \{S_1, S_2, ..., S_N\}$ with $S_i = (p_i, y_i), i = 1, 2, ..., N$, where p_i is a point in the parameter space and $y_i = 1$ (stable) or -1 (unstable) is the label of the point, SVM is to solve the following problem:

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \phi(p_i)^T \phi(p_j), \\ \text{subject to} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0, \\ & 0 \leq \alpha_i \leq C, \end{aligned}$$

where α is the Lagrange multiplier, $C > 0$ is the penalty parameter which can be set by users and $\phi(\cdot)$ is a mapping from p_i to a higher dimensional space.

There have already been many SVM tool kits that can be used to solve the classification problems. LIBSVM [112] is a simple and effective one developed by Chih-Jen Lin's research group. Throughout this chapter, the LibSVM kit is employed to do simulation with proper arguments.

6.5 Simulation Examples

In this section, four examples are presented to illustrate the effectiveness of the proposed method.

Example 6.1. The analytical method in [84] cannot deal with a process containing multiple zeros, while our method does not have this constraint. Consider the plant:

$$G(s) = \frac{(0.4s+1)(0.2s+1)}{(s-1)(0.5s+1)(0.1s+1)} e^{-ds},$$

with a P controller $C(s) = kI_2$. This control system is converted to the form in (6.10) with

$$\tilde{A} = \begin{bmatrix} -11 & -8 & 20 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

$$\tilde{A}_1 = \begin{bmatrix} -1.6k & -12k & -20k \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Let $p = [d, k]$. Performing our method with the LibSVM arguments '-t'=2 and '-c'=100, the stabilizing parameter region is obtained and shown in Figure 6.3.

Example 6.2. The graphical method in [89] cannot deal with a process containing

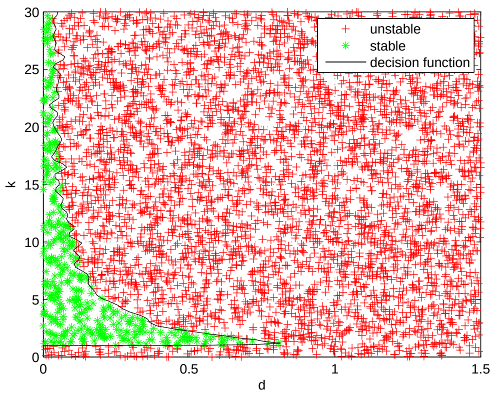


Figure 6.3: Stabilizing parameter region for Example 6.1.

state-delays. However, our method does not have this restriction. Consider the plant:

$$\begin{cases} \dot{x}(t) = \begin{bmatrix} -12.5 & -25 \\ 1 & 0 \end{bmatrix} x(t) \\ + \begin{bmatrix} 0 & 10 \\ 1.5 & 0 \end{bmatrix} x(t-d) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t), \\ y(t) = \begin{bmatrix} 0 & 25 \end{bmatrix} x(t), \end{cases} \tag{6.14}$$

with a P controller $u = -ky$. This control system is converted to the form in (6.10) with

$$\tilde{A} = \begin{bmatrix} -12.5 & -25-25k \\ 1 & 0 \end{bmatrix}, \ \tilde{A}_1 = \begin{bmatrix} 0 & 10 \\ 1.5 & 0 \end{bmatrix}.$$

Let $p = [d, k]$. Performing our method with '-t'=2 and '-c'=1000, the stabilizing parameter region is obtained and shown in Figure 6.4.

Example 6.3. Consider the plant (6.14) with $d = 0.5$ under the controller

$$C(s) = \frac{a}{s+b}. \tag{6.15}$$

Note that b appears in a nonlinear fashion, which is different from parameters of PID

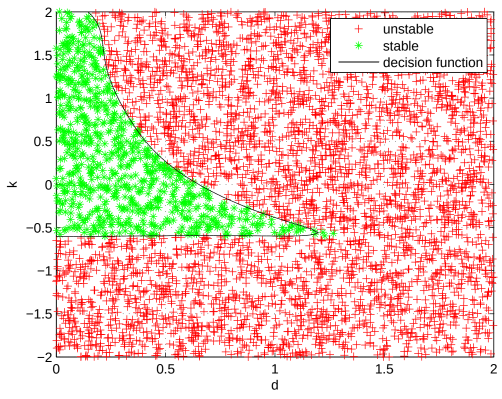


Figure 6.4: Stabilizing parameter region for Example 6.2.

controllers. We can rewrite (6.15) as

$$\begin{cases} \dot{v}(t) = -bv(t) + y(t), \\ u(t) = av(t). \end{cases}$$

This control system is converted to the form in (6.10) with

$$\tilde{A} = \begin{bmatrix} -12.5 & -25 & a \\ 1 & 0 & 0 \\ 0 & 25 & -b \end{bmatrix}, \quad \tilde{A}_1 = \begin{bmatrix} 0 & 10 & 0 \\ 1.5 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Let $p = [b, a]$. Performing our method with '-t'=2 and '-c'=1000, the stabilizing parameter region is obtained and shown in Figure 6.5.

Example 6.4. The proposed method also works well with a high-dimensional param-

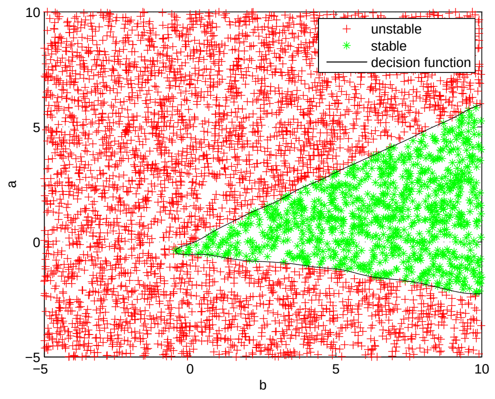


Figure 6.5: Stabilizing parameter region for Example 6.3.

eter space. Consider the plant:

$$\begin{cases} \dot{x}(t) = \begin{bmatrix} -12.5 & -25 \\ 1 & 0 \end{bmatrix} x(t) \\ + \begin{bmatrix} 0 & 10 \\ 1.5 & 0 \end{bmatrix} x(t-d_1) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t-d_2), \\ y(t) = \begin{bmatrix} 0 & 25 \end{bmatrix} x(t), \end{cases}$$

with a controller:

$$\begin{cases} \dot{v}(t) = -bv(t) + y(t), \\ u(t) = v(t). \end{cases}$$

This control system is converted to the form in (6.10) with

$$\tilde{A} = \begin{bmatrix} -12.5 & -25 & 0 \\ 1 & 0 & 0 \\ 0 & 25 & -b \end{bmatrix}, \tilde{A}_1 = \begin{bmatrix} 0 & 10 & 0 \\ 1.5 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\text{and } \tilde{A}_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Let $p = [d_1, d_2, b]$. Performing our method with '-t'=2 and '-c'=1000, the stabilizing parameter region is obtained and shown in Figure 6.6.

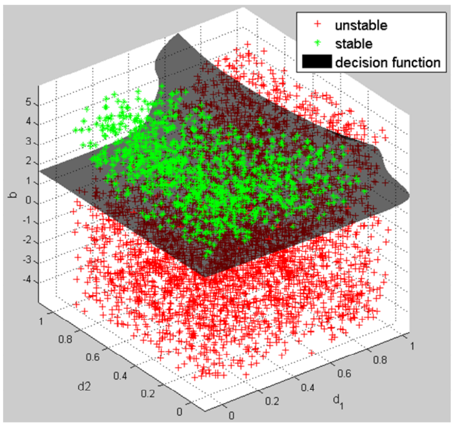


Figure 6.6: Stabilizing parameter region for Example 6.4.

The above examples have well illustrated the effectiveness of the proposed method which can be applied to a much wider range of systems than the existing methods in the literature.

6.6 Conclusions

This chapter proposes a new and general method for determining the stabilizing parameter regions for delay control systems. We first take a certain number of random sample points in the parameter space. Next, we represent a delay control system in a unified state-space form. Then the numerical stability condition is developed and checked for sample points in the parameter space. These points are divided into two classes according to whether they can stabilize the system. The stabilizing parameter regions could be well defined by the decision function obtained from some learning method. The effectiveness of the proposed method is well illustrated with examples. The proposed method does not have essential constraints and has a wide range of applications. Note that our method could be applied to a higher-dimensional parameter space, though the stabilizing parameter regions are difficult to be shown by graphics.

It should be noted that the presented LMI stability criterion is only sufficient since it is based on Lyapunov theory. A sufficient and necessary stability criterion and the additional potential values of the proposed method are to be investigated in future works.

Chapter 7

Conclusions

7.1 Main Findings

This thesis develops some new techniques to help with assessing models in statistical learning, improving the outcome of system identification, solving global optimization problems and finding stabilizing parameter regions for control systems.

In Chapter 2, we propose a new method for model assessment based on Renormalization Group. Renormalization Group is applied to the original data set to obtain the transformed data set with the majority rule to set its labels. The assessment is first performed on the data level without invoking any learning method, and the consistency and non-randomness indices are defined by comparing two data sets to reveal informative content of the data. When the indices indicate informative data, the next assessment is carried out at the model level, and the predictions are compared between two models learnt from the original and transformed data sets, respectively. The model consistency and reliability indices are introduced accordingly. Unlike cross-validation and other standard methods in the literature, the proposed method creates a new data set and data assessment. Besides, it requires only two models and thus less computational burden for model assessment.

In Chapter 3, we propose an improved system identification method with Renormalization Group. Renormalization Group is applied to a fine data set to obtain a coarse data set. The least squares algorithm is performed on the coarse data set. The theoretical analysis under certain conditions shows that the parameter estimation error could be reduced.

In Chapter 4, the outlier detection problem for dynamic systems is formulated as a matrix decomposition problem with low-rank and sparse matrices, and further recast as a semidefinite programming (SDP) problem. A fast algorithm is presented to solve the resulting problem while keeping the solution matrix structure and it can greatly reduce the computational cost over the standard interior-point method. The computational burden is further reduced by proper construction of subsets of the raw data without violating low rank property of the involved matrix. The proposed method can make exact detection of outliers in case of no or little noise in output observations. In case of significant noise, a novel approach based on under-sampling with averaging is developed to denoise while retaining the saliency of outliers, and so-filtered data enables successful outlier detection with the proposed method while the existing filtering methods fail. Use of recovered "clean" data from the proposed method can give much better parameter estimation compared with that based on the raw data.

In Chapter 5, we propose a brand-new method for global optimization through randomized group search in contracting regions. For each iteration, a population is randomly produced within the search region, where the population size is chosen to ensure that the empirical optimum is an estimate of the true optimum within a predefined accuracy with a certain confidence. Fitness values are evaluated at the samples in the population. A very small subset of them with top-ranking fitness values are selected as good points. Neighborhoods of these good points are used to form a new and smaller search region, in which a new population is generated. It is shown that the proposed algorithm always converges and the convergence to local or global optima is analyzed. It is easy to implement the algorithm. Extensive simulation on benchmark problems shows that the proposed method is fast and reasonably accurate.

In Chapter 6, we propose a method for determining the stabilizing parameter regions for general delay control systems based on randomized sampling. A delay control system is converted into a unified state-space form. The numerical stability condition is developed and checked for sample points in the parameter space. These points are separated into stable and unstable regions by the decision function obtained from some learning method. The proposed method is very general and applied to a much wider range of systems than the existing methods in the literature.

7.2 Future Works

The model assessment method proposed in Chapter 2 is illustrated for classification problem. We would like to extend our method to regression problem with trivial modifications. A coarse response, $\hat{y}_j$, of a unit may be obtained simple averaging or a weighted average of responses, y_i, of fine points in the same unit, while the coarse feature can be determined in the same way as in classification. We can use the standard squared errors for regression to make comparisons at both data and model.

In Chapter 3, system identification with Renormalization Group is new and we may advance it with much better results.

In Chapter 4, we would like to investigate more effective methods for system identification when the output is with both noise and outliers.

The global optimization method proposed in Chapter 5 adopts the rejection sampling method for sampling uniformly within an irregular shaped search region. Due to the "curse of dimensionality", the rejection sampling method will not be very efficient in a high-dimensional space. Therefore, we may study some more efficient substitutes on sampling uniformly in a high-dimensional space. In addition, once the tuning parameters are chosen, they are fixed throughout iterations. To further improve the performance of the proposed method, we will develop some adaptive schemes which incessantly adjust the parameters.

The method for determining the stabilizing parameter regions proposed in Chapter 6 uses the LMI stability condition to check the sample points in the parameter space. This condition is only sufficient since it is based on Lyapunov theory. A sufficient and necessary stability criterion is to be investigated.

Bibliography

[1] G. F. Franklin, M. L. Workman, and D. Powell, *Digital control of dynamic systems*. Addison-Wesley Longman Publishing Co., Inc., 1997.

[2] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer, 2009.

[3] O. Bousquet, S. Boucheron, and G. Lugosi, “Introduction to Statistical Learning Theory,” *Lecture Notes in Computer Science.*, no. 3176, pp. 169–207, 2004.

[4] M. R. E. Symonds and A. Moussalli, “A Brief Guide to Model Selection, Multimodel Inference and Model Averaging in Behavioural Ecology Using Akaike’s Information Criterion,” *Behavioral Ecology and Sociobiology*, vol. 65, no. 1, pp. 13–21, 2011.

[5] J. Myung and M. Pitt, “Model Selection Methods,” *Amsterdam Workshop on Model Selection*, 2004.

[6] J. Shao, “Bootstrap Model Selection,” *Journal of the American Statistical Association*, vol. 91, no. 434, pp. 655–665, 1996.

[7] P. Refaeilzadeh, L. Tang, and H. Liu, “Cross-Validation,” *Encyclopedia of Database Systems*, pp. 532–538, 2009.

[8] K. Polat and S. Günes, “An Expert System Approach Based on Principal Component Analysis and Adaptive Neuro-Fuzzy Inference System to Diagnosis of Diabetes Disease,” *Digital Signal Processing*, vol. 17, no. 4, pp. 702–710, 2007.

[9] A. Camstra and A. Boomsma, “Cross-Validation in Regression and Covariance Structure Analysis,” *Sociological Methods & Research*, vol. 21, no. 1, pp. 89–115, 1992.

[10] L. Lennart, “System identification: theory for the user,” *PTR Prentice Hall, Upper Saddle River, NJ*, 1999.

[11] J. Schoukens, *System identification.* Wiley, 2012.

[12] F. Ding, “System identification: New theory and methods,” 2013.

[13] D. Vecchia and J. Splett, “Outlier-resistant methods for estimation and model fitting,” *ISA Transactions*, vol. 33, no. 4, pp. 411–420, 1994.

[14] R. Klopfenstein Jr, “Data smoothing using a least squares fit c++ class,” *ISA transactions*, vol. 37, no. 1, pp. 3–19, 1998.

[15] J. C. Peyton Jones and K. R. Muske, “Identification and adaptation of linear look-up table parameters using an efficient recursive least-squares technique,” *ISA transactions*, vol. 48, no. 4, pp. 476–483, 2009.

[16] K. Natarajan, A. Gilbert, B. Patel, and R. Siddha, “Frequency response adaptation of pi controllers based on recursive least-squares process identification,” *ISA transactions*, vol. 45, no. 4, pp. 517–528, 2006.

[17] Z. Griliches, “A note on serial correlation bias in estimates of distributed lags,” *Econometrica: journal of the Econometric Society*, pp. 65–73, 1961.

[18] P. C. B. Phillips and M. R. Wickens, *Exercises in econometrics.* P. Allan, 1978, vol. 2.

[19] B. A. Inder, “Bias in the ordinary least squares estimator in the dynamic linear regression model with autocorrelated disturbances,” *Communications in Statistics-Simulation and Computation*, vol. 16, no. 3, pp. 791–815, 1987.

[20] T. Stocker, “On the asymptotic bias of ols in dynamic regression models with autocorrelated errors,” *Statistical Papers*, vol. 48, no. 1, pp. 81–93, 2007.

[21] W. X. Zheng, “On a least-squares-based algorithm for identification of stochastic linear systems,” *IEEE Transactions on Signal Processing*, vol. 46, no. 6, pp. 1631–1638, 1998.

[22] ——, “On least-squares identification of armax models,” in *Proceedings of the 15th IFAC Triennial World Congress, Barcelona, Spain*, 2002.

[23] T. Söderström and P. Stoica, *Instrumental variable methods for system identification.* Springer Berlin et al., 1983.

[24] S. Victor, R. Malti, and A. Oustaloup, “Instrumental variable method with optimal fractional differentiation order for continuous-time system identification,” in *Proceedings of the 15th IFAC SYSID Conference*, vol. 15, no. 1, 2009, pp. 904–909.

[25] X. Liu, J. Wang, and W. X. Zheng, “Convergence analysis of refined instrumental variable method for continuous-time system identification,” *Control Theory & Applications, IET*, vol. 5, no. 7, pp. 868–877, 2011.

[26] L. Hurwicz, “Least squares bias in time series,” *Statistical Inference in Dynamic Economic Models*, no. 10, pp. 365–383, 1950.

[27] P. Shaman and R. A. Stine, “The bias of autoregressive coefficient estimators,” *Journal of the American Statistical Association*, vol. 83, no. 403, pp. 842–848, 1988.

[28] R. A. Stine and P. Shaman, “A fixed point characterization for bias of autoregressive estimators,” *The Annals of Statistics*, pp. 1275–1284, 1989.

[29] A. Breton and D. T. Pham, “On the bias of the least squares estimator for the first order autoregressive process,” *Annals of the institute of Statistical Mathematics*, vol. 41, no. 3, pp. 555–563, 1989.

[30] K. Patterson, “Finite sample bias of the least squares estimator in an ar (p) model: estimation, inference, simulation and examples,” *Applied Economics*, vol. 32, no. 15, pp. 1993–2005, 2000.

[31] T. J. Sargent, “Some evidence on the small sample properties of distributed lag estimators in the presence of autocorrelated disturbances,” *The Review of Economics and Statistics*, vol. 50, no. 1, pp. 87–95, 1968.

[32] D. Tjøstheim and J. Paulsen, “Bias of some commonly-used time series estimates,” *Biometrika*, vol. 70, no. 2, pp. 389–399, 1983.

[33] A. Maeshiro, “Teaching regressions with a lagged dependent variable and autocorrelated disturbances,” *Journal of Economic Education*, pp. 72–84, 1996.

[34] R. K. Pearson, “Outliers in process modeling and identification,” *IEEE Transactions on Control Systems Technology*, vol. 10, no. 1, pp. 55–63, 2002.

[35] V. J. Hodge and J. Austin, “A survey of outlier detection methodologies,” *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[36] D. Zhang, L. Yu, and Q.-G. Wang, “Fault detection for a class of network-based nonlinear systems with communication constraints and random packet dropouts,” *International Journal of Adaptive Control and Signal Processing*, vol. 25, no. 10, pp. 876–898, 2011.

[37] W. Liu, Z. Wang, and M. Ni, “Controlled synchronization for chaotic systems via limited information with data packet dropout,” *Automatica*, vol. 49, no. 8, pp. 2576–2579, 2013.

[38] M. Davy and S. Godsill, “Detection of abrupt spectral changes using support vector machines an application to audio signal segmentation,” in *IEEE International*

Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 2. IEEE, 2002, pp. 1313–1316.

[39] G. G. Hazel, “Multivariate gaussian mrf for multispectral scene segmentation and anomaly detection,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 38, no. 3, pp. 1199–1211, 2000.

[40] M. J. Desforges, P. J. Jacob, and J. E. Cooper, “Applications of probability density estimation to the detection of abnormal conditions in engineering,” *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 212, no. 8, pp. 687–703, 1998.

[41] R. J. Bolton, D. J. Hand *et al.*, “Unsupervised profiling methods for fraud detection,” *Credit Scoring and Credit Control VII*, pp. 235–255, 2001.

[42] H. He, J. Wang, W. Graco, and S. Hawkins, “Application of neural networks to detection of medical fraud,” *Expert Systems with Applications*, vol. 13, no. 4, pp. 329–336, 1997.

[43] W. W.-S. Wei, *Time series analysis*. Addison-Wesley Redwood City, California, 1994.

[44] F. Pukelsheim, “The three sigma rule,” *The American Statistician*, vol. 48, no. 2, pp. 88–91, 1994.

[45] R. D. Martin and D. J. Thomson, “Robust-resistant spectrum estimation,” *Proceedings of the IEEE*, vol. 70, no. 9, pp. 1097–1115, 1982.

[46] R. K. Pearson, "Exploring process data," *Journal of Process Control*, vol. 11, no. 2, pp. 179–194, 2001.

[47] P. J. Rousseeuw, "Least median of squares regression," *Journal of the American statistical association*, vol. 79, no. 388, pp. 871–880, 1984.

[48] D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "On the least trimmed squares estimator," *Algorithmica*, pp. 1–36, 2007.

[49] P. Bloomfield and W. L. Steiger, *Least absolute deviations.* Springer, 1984.

[50] P. W. Holland and R. E. Welsch, "Robust regression using iteratively reweighted least-squares," *Communications in Statistics-Theory and Methods*, vol. 6, no. 9, pp. 813–827, 1977.

[51] S. P. Ellis, "Instability of least squares, least absolute deviation and least median of squares linear regression," *Statistical Science*, vol. 13, no. 4, pp. 337–350, 1998.

[52] A. Leontitsis, *LMS Toolbox.* http://www.mathworks.com/matlabcentral/fileexchange/801-lms-toolbox, 2004.

[53] J. Agulló, C. Croux, and S. Van Aelst, "The multivariate least-trimmed squares estimator," *Journal of Multivariate Analysis*, vol. 99, no. 3, pp. 311–338, 2008.

[54] M. A. Branch and A. Grace, *MATLAB: optimization toolbox: user's guide version 1.5.* The MathWorks, 1996.

[55] B. Jones, *MATLAB: Statistics Toolbox; User's Guide.* MathWorks, 1997.

[56] V. Chandola, A. Banerjee, and V. Kumar, "Outlier detection: A survey," *ACM Computing Surveys, to appear*, 2007.

[57] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *Journal of the ACM (JACM)*, vol. 58, no. 3, p. 11, 2011.

[58] J. Wright, A. Ganesh, S. Rao, Y. G. Peng, and Y. Ma, "Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization," *Advances in neural information processing systems*, vol. 22, pp. 2080–2088, 2009.

[59] R. Basri and D. W. Jacobs, "Lambertian reflectance and linear subspaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 218–233, 2003.

[60] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *JASIS*, vol. 41, no. 6, pp. 391–407, 1990.

[61] Z. Lin, M. Chen, and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," *arXiv preprint arXiv:1009.5055*, 2010.

[62] Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, and Y. Ma, "Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix," *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, vol. 61, 2009.

[63] A. Ganesh, Z. Lin, J. Wright, L. Wu, M. Chen, and Y. Ma, "Fast algorithms for recovering a corrupted low-rank matrix," in *3rd IEEE International Work-*

shop on Computational Advances in Multi-Sensor Adaptive Processing (CAM-SAP). IEEE, 2009, pp. 213–216.

[64] M. Ayazoglu, M. Sznaier, and O. I. Camps, "Fast algorithms for structured robust principal component analysis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, 2012, pp. 1704–1711.

[65] M. Fazel, H. Hindi, and S. Boyd, "Rank minimization and applications in system theory," in *Proceedings of the American Control Conference*, vol. 4. IEEE, 2004, pp. 3273–3278.

[66] Z. Liu and L. Vandenberghe, "Interior-point method for nuclear norm approximation with application to system identification," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 3, pp. 1235–1256, 2009.

[67] G. Venter, "Review of optimization techniques," *Encyclopedia of aerospace engineering*, 2010.

[68] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear programming: theory and algorithms.* John Wiley & Sons, 2013.

[69] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, "Lipschitzian optimization without the lipschitz constant," *Journal of Optimization Theory and Applications*, vol. 79, no. 1, pp. 157–181, 1993.

[70] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information Sciences*, vol. 237, pp. 82–117, 2013.

[71] S. Brooks and B. Morgan, “Optimization using simulated annealing,” *The Statistician*, pp. 241–257, 1995.

[72] F. Glover, “Future paths for integer programming and links to artificial intelligence,” *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549, 1986.

[73] V. D. Pinto and W. M. Pottenger, “A survey of optimization techniques being used in the field,” in *The Proceedings of the Third International Meeting on Research in Logistics (IMRL)*. Citeseer, 2000.

[74] R. Battiti and G. Tecchiolli, “Simulated annealing and tabu search in the long run: a comparison on qap tasks,” *Computers & Mathematics with Applications*, vol. 28, no. 6, pp. 1–8, 1994.

[75] J. Paulli, “Information utilization in simulated annealing and tabu search,” *COAL Bulletin*, vol. 22, no. 28-34, 1993.

[76] J. H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence.* U Michigan Press, 1975.

[77] R. Storn and K. Price, “Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces,” *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[78] V. Kachitvichyanukul, “Comparison of three evolutionary algorithms: Ga, pso, and de,” *Industrial Engineering and Management Systems*, vol. 11, pp. 215–223, 2012.

[79] R. Mallipeddi, P. N. Suganthan, Q.-K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing*, vol. 11, no. 2, pp. 1679–1696, 2011.

[80] J. Kennedy, R. Eberhart *et al.*, "Particle swarm optimization," in *Proceedings of IEEE international conference on neural networks*, vol. 4, no. 2. Perth, Australia, 1995, pp. 1942–1948.

[81] M. Moradi and M. Abedini, "A combination of genetic algorithm and particle swarm optimization for optimal dg location and sizing in distribution systems," *International Journal of Electrical Power & Energy Systems*, vol. 34, no. 1, pp. 66–74, 2012.

[82] F. Valdez, P. Melin, and O. Castillo, "An improved evolutionary method with fuzzy logic for combining particle swarm optimization and genetic algorithms," *Applied Soft Computing*, vol. 11, no. 2, pp. 2625–2632, 2011.

[83] Q.-G. Wang, C. Lin, Z. Ye, G. Wen, Y. He, and C. C. Hang, "A quasi-lmi approach to computing stabilizing parameter ranges of multi-loop pid controllers," *Journal of Process Control*, vol. 17, no. 1, pp. 59–72, 2007.

[84] S. C. Lee and Q.-G. Wang, "Stabilization conditions for a class of unstable delay processes of higher order," *Journal of the Taiwan Institute of Chemical Engineers*, vol. 41, no. 4, pp. 440–445, 2010.

[85] S. C. Lee, Q.-G. Wang, and C. Xiang, "Stabilization of all-pole unstable delay

processes by simple controllers," *Journal of Process Control*, vol. 20, no. 2, pp. 235–239, 2010.

[86] S. C. Lee, Q.-G. Wang, and B. N. Le, "Stabilizing control for a class of delay unstable processes," *ISA transactions*, vol. 49, no. 3, pp. 318–325, 2010.

[87] Z. Y. Nie, Q.-G. Wang, M. Wu, and Y. He, "Exact computation of loop gain margins of multivariable feedback systems," *Journal of Process Control*, vol. 20, no. 6, pp. 762–768, 2010.

[88] J. Liu, Y. Xue, and D. Li, "Calculation of pi controller stable region based on d-partition method," in *International Conference on Control Automation and Systems (ICCAS)*. IEEE, 2010, pp. 2185–2189.

[89] Q.-G. Wang, B. N. Le, and T. H. Lee, "Graphical methods for computation of stabilizing gain ranges for tito systems," in *9th IEEE International Conference on Control and Automation (ICCA)*. IEEE, 2011, pp. 82–87.

[90] Q.-G. Wang, Y. He, Z. Ye, C. Lin, and C. C. Hang, "On loop phase margins of multivariable control systems," *Journal of Process Control*, vol. 18, no. 2, pp. 202–211, 2008.

[91] M. Söylemez, N. Munro, and H. Baki, "Fast calculation of stabilizing pid controllers," *Automatica*, vol. 39, no. 1, pp. 121–126, 2003.

[92] N. Tan, I. Kaya, C. Yeroglu, and D. P. Atherton, "Computation of stabilizing pi and pid controllers using the stability boundary locus," *Energy Conversion and Management*, vol. 47, no. 18, pp. 3045–3058, 2006.

[93] B. Fang, “Computation of stabilizing pid gain regions based on the inverse nyquist plot,” *Journal of Process Control*, vol. 20, no. 10, pp. 1183–1187, 2010.

[94] E. N. Gryazina and B. T. Polyak, “Stability regions in the parameter space: D-decomposition revisited,” *Automatica*, vol. 42, no. 1, pp. 13–26, 2006.

[95] K. Saadaoui, S. Testouri, and M. Benrejeb, “Robust stabilizing first-order controllers for a class of time delay systems,” *ISA transactions*, vol. 49, no. 3, pp. 277–282, 2010.

[96] R. Tempo, G. Calafiore, and F. Dabbene, *Randomized Algorithms for Analysis and Control of Uncertain Systems*. Springer, 2004.

[97] K. G. Wilson, “Renormalization Group and Critical Phenomena. I. Renormalization Group and the Kadanoff Scaling Picture,” *Physical Review B*, vol. 4, no. 9, p. 3174, 1971.

[98] A. D. Arulsamy, “Renormalization Group Method Based on the Ionization Energy Theory,” *Annals of Physics*, vol. 326, no. 3, pp. 541–565, 2011.

[99] P. Q. Hung and C. Xiong, “Renormalization Group Fixed Point with a Fourth Generation: Higgs-Induced Bound States and Condensates,” *Nuclear Physics B*, 2011.

[100] B. Hu, “Introduction to Real-Space Renormalization-Group Methods in Critical and Chaotic Phenomena,” *Physics Reports*, vol. 91, no. 5, pp. 233–295, 1982.

[101] W. D. McComb, *Renormalization Methods: A Guide for Beginners*. Oxford University Press, USA, 2007.

[102] J. J. Binney, N. J. Dowrick, A. J. Fisher, and M. Newman, *The Theory of Critical Phenomena: An Introduction to the Renormalization Group*. Oxford University Press, Inc., 1992.

[103] A. Sarkar and J. Bhattacharjee, "Renormalization Group as a Probe for Dynamical Systems," in *Journal of Physics: Conference Series*, vol. 319. IOP Publishing, 2011, p. 012017.

[104] M. Carfora, "Renormalization Group and the Ricci Flow," *Arxiv preprint arXiv:1001.3595*, 2010.

[105] D. Sornette, *Why Stock Markets Crash: Critical Events in Complex Financial Systems*. Princeton Univ Pr, 2004.

[106] S. R. Gunn, "Support Vector Machines for Classification and Regression," *ISIS Technical Report*, vol. 14, 1998.

[107] H. R. Zhang, X. D. Wang, C. J. Zhang, and X. S. Cai, "Robust Identification of Non-linear Dynamic Systems Using Support Vector Machine," in *Science, Measurement and Technology, IEE Proceedings-*, vol. 153, no. 3. IET, 2006, pp. 125–129.

[108] M. Davy, A. Gretton, A. Doucet, and P. J. W. Rayner, "Optimized Support Vector Machines for Nonstationary Signal Classification," *Signal Processing Letters, IEEE*, vol. 9, no. 12, pp. 442–445, 2002.

[109] A. Gretton and F. Desobry, “On-line One-class Support Vector Machines. An Application to Signal Segmentation.” in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2. IEEE, 2003, pp. 709–712.

[110] C. W. Hsu, C. C. Chang, C. J. Lin, and Others, “A Practical Guide to Support Vector Classification,” 2003.

[111] D. Meyer, “Support Vector Machines,” *Porting R to Darwin/X11 and Mac OS X*, p. 23, 2011.

[112] C. C. Chang and C. J. Lin, “LIBSVM: A Library for Support Vector Machines,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[113] J. Yeo and M. A. Moore, “Renormalization Group Analysis of the Mp-spin Glass Model with p= 3 and M= 3,” *Arxiv preprint arXiv:1111.3105*, 2011.

[114] A. K. Jain, “Data Clustering: 50 years beyond K-means,” *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[115] D. K. Roy and L. K. Sharma, “Genetic k-means Clustering Algorithm for Mixed Numeric and Categorical Data Sets,” *International Journal of Artificial Intelligence & Applications*, vol. 1, no. 2, pp. 23–28, 2010.

[116] H. Chen, P. Tino, and X. Yao, “Probabilistic Classification Vector Machines,” *IEEE Transactions on Neural Networks*, vol. 20, no. 6, pp. 901–914, 2009.

[117] A. Uzilov, J. Keegan, and D. Mathews, “Detection of Non-coding RNAs on the Basis of Predicted Secondary Structure Formation Free Energy Change,” *BMC Bioinformatics*, vol. 7, no. 1, p. 173, 2006.

[118] J. Demšar, “Statistical Comparisons of Classifiers over Multiple Data Sets,” *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[119] P. Eykhoff, *System identification: Parameter and statte estimation.* NY: John Wiley & Sons, 1974.

[120] *Regression Analysis Tutorial.* University of California at Berkeley, 1967.

[121] A. Dumitru, J. Jalilian-Marian, T. Lappi, B. Schenke, and R. Venugopalan, “Renormalization group evolution of multi-gluon correlators in high energy qcd,” *Physics Letters B*, vol. 706, no. 2, pp. 219–224, 2011.

[122] J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* New York, NY: Springer-Verlag New York, 2009.

[123] S. Salas, E. Hille, and G. Etgen, *Calculus: One and several variables.* Wiley, 1990.

[124] T. Söderström, “Ergodicity results for sample covariances,” *Problems of Control and Information Theory*, vol. 4, no. 2, pp. 131–138, 1975.

[125] T. Söderström and P. Stoica, *System identification.* Prentice-Hall, Inc., 1988.

[126] O. Reiersøl, “Confluence analysis by means of lag moments and other methods of confluence analysis,” *Econometrica: Journal of the Econometric Society*, pp. 1–24, 1941.

[127] K. L. Chung, *A course in probability theory.* Harcourt, Brace & World (New York), 1968.

[128] Q.-G. Wang, X. Guo, and Y. Zhang, “Direct identification of continuous time delay systems from step responses,” *Journal of Process Control*, vol. 11, no. 5, pp. 531–542, 2001.

[129] T. Söderström and P. Stoica, *Instrumental variable methods for system identification.* Springer-Verlag Berlin, 1983, vol. 161.

[130] T. Söderström, “A generalized instrumental variable estimation method for errors-in-variables identification problems,” *Automatica*, vol. 47, no. 8, pp. 1656–1666, 2011.

[131] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, “Rank-sparsity incoherence for matrix decomposition,” *SIAM Journal on Optimization*, vol. 21, no. 2, pp. 572–596, 2011.

[132] C. Helmberg, F. Rendl, R. J. Vanderbei, and H. Wolkowicz, “An interior-point method for semidefinite programming,” *SIAM Journal on Optimization*, vol. 6, no. 2, pp. 342–361, 1996.

[133] R. H. Tütüncü, K. C. Toh, and M. J. Todd, "Solving semidefinite-quadratic-linear programs using sdpt3," *Mathematical programming*, vol. 95, no. 2, pp. 189–217, 2003.

[134] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM review*, vol. 38, no. 1, pp. 49–95, 1996.

[135] H. Wolkowicz, *Semidefinite programming*. Faculty of Mathematics, University of Waterloo, 2002.

[136] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[137] J. J. Dattorro, *Convex optimization and Euclidean distance geometry*. Meboo Publishing USA, 2005.

[138] M. J. Todd, K. C. Toh, and R. H. Tütüncü, "On the nesterov–todd direction in semidefinite programming," *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 769–796, 1998.

[139] L. Vandenberghe, V. R. Balakrishnan, R. Wallin, A. Hansson, and T. Roh, "Interior-point algorithms for semidefinite programming problems derived from the kyp lemma," in *Positive polynomials in control*. Springer, 2005, pp. 195–238.

[140] Z. Liu, "Structured semidefinite programs in system identification and control," Ph.D. dissertation, University of California Los Angeles, 2009.

[141] Y. Zhang, "On extending some primal–dual interior-point algorithms from linear programming to semidefinite programming," *SIAM Journal on Optimization*, vol. 8, no. 2, pp. 365–386, 1998.

[142] M. Holmes, A. Gray, and C. Isbell, "Fast svd for large-scale matrices," in *Workshop on Efficient Machine Learning at NIPS*, 2007.

[143] S. Mallat, *A wavelet tour of signal processing*. Access Online via Elsevier, 1999.

[144] S. D. Ruikar and D. D. Doye, "Wavelet based image denoising technique," *International Journal of Advanced Computer Science and Applications*, vol. 2, no. 3, pp. 49–53, 2011.

[145] G. Cristobal, M. Chagoyen, B. Escalante-Ramirez, and J. R. Lopez, "Wavelet-based denoising methods: a comparative study with applications in microscopy," in *SPIE International Symposium on Optical Science, Engineering, and Instrumentation*. International Society for Optics and Photonics, 1996, pp. 660–671.

[146] A. C. To, J. R. Moore, and S. D. Glaser, "Wavelet denoising techniques with applications to experimental geophysical data," *Signal Processing*, vol. 89, no. 2, pp. 144–160, 2009.

[147] M. Misiti, Y. Misiti, G. Oppenheim, and J.-M. Poggi, *Wavelets and their Applications*. Wiley Online Library, 2007.

[148] D. L. Donoho and I. M. Johnstone, "Adapting to unknown smoothness via wavelet shrinkage," *Journal of the american statistical association*, vol. 90, no. 432, pp. 1200–1224, 1995.

[149] C. M. Stein, “Estimation of the mean of a multivariate normal distribution,” *The annals of Statistics*, pp. 1135–1151, 1981.

[150] J. Dahl and L. Vandenberghe, “Cvxopt: A python package for convex optimization,” in *Proc. eur. conf. op. res*, 2006.

[151] J. Lofberg, “Yalmip: A toolbox for modeling and optimization in matlab,” in *IEEE International Symposium on Computer Aided Control Systems Design*. IEEE, 2004, pp. 284–289.

[152] “Test program for matlab and python,” 2013, available on line: http://stackoverflow.com/questions/17559140/matlab-twice-as-fast-as-numpy.

[153] M. U. Guide, “The mathworks,” *Inc., Natick, MA*, vol. 5, 1998.

[154] R. Tempo, E.-W. Bai, and F. Dabbene, “Probabilistic robustness analysis: Explicit bounds for the minimum number of samples,” in *Proceedings of the 35th IEEE Conference on Decision and Control*, vol. 3. IEEE, 1996, pp. 3424–3428.

[155] R. Tempo and H. Ishii, “Monte carlo and las vegas randomized algorithms for systems and control: An introduction,” *European journal of control*, vol. 13, no. 2, pp. 189–203, 2007.

[156] G. Calafiore, F. Dabbene, and R. Tempo, “A survey of randomized algorithms for control synthesis and performance verification,” *Journal of Complexity*, vol. 23, no. 3, pp. 301–316, 2007.

[157] E.-W. Bai, R. Tempo, and M. Fu, "Worst-case properties of the uniform distribution and randomized algorithms for robustness analysis," *Mathematics of Control, Signals and Systems*, vol. 11, no. 3, pp. 183–196, 1998.

[158] W. Feller, *An introduction to probability theory and its applications*. John Wiley & Sons, 2008, vol. 2.

[159] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimisation problems," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 2, pp. 150–194, 2013.

[160] A.-R. Hedar, "Test problems for constrained global optimization," available on line: http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page422.htm.

[161] J. Liang, B. Qu, P. Suganthan, and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization," *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report*, 2013.

[162] H. L. Royden, P. Fitzpatrick, and P. Hall, *Real analysis*. Prentice Hall New York, 1988, vol. 4.

[163] I. Loshchilov, T. Stuetzle, and T. Liao, *Ranking Results of CEC'13 Special Session & Competition on Real-Parameter Single Objective Optimization*, 2013.

[164] S. M. Elsayed, R. A. Sarker, and D. L. Essam, “A genetic algorithm for solving the cec’2013 competition problems on real-parameter optimization,” in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2013, pp. 356–360.

[165] A. Qin and X. Li, “Differential evolution on the cec-2013 single-objective continuous optimization testbed,” in *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2013, pp. 1099–1106.

[166] F. Zheng, Q.-G. Wang, and T. H. Lee, “On the design of multivariable PID controllers via LMI approach,” *Automatica*, vol. 38, no. 3, pp. 517–526, 2002.

[167] G. F. Franklin, J. D. Powell, A. Emami-Naeini, and J. D. Powell, *Feedback control of dynamic systems*. Addison-Wesley Reading, MA, 1994, vol. 3.

[168] K. Gu, V. Kharitonov, and J. Chen, *Stability of time-delay systems*. Birkhauser, 2003.

[169] V. N. Vapnik, “The nature of statistical learning theory,” 1995.

[170] P. H. Chen, C. J. Lin, and B. Schölkopf, “A tutorial on ν-support vector machines,” *Applied Stochastic Models in Business and Industry*, vol. 21, no. 2, pp. 111–136, 2005.

Author's Publications

The author has contributed to the following publications:

Journal Papers:

[1] C. Yu, Q.-G. Wang, L. Wang and W. Feng, "Global Optimization by Randomized Group Search in Contracting Regions," manuscript is submitted to *IEEE Transactions on Evolutionary Computation.*

[2] C. Yu, Q.-G. Wang and D. Zhang, "System Identification in Presence of Outliers," manuscript is submitted to *Industrial & Engineering Chemistry Research.*

[3] Q.-G. Wang, C. Yu and Y. Zhang, "Model Assessment Through Renormalization Group In Statistical Learning," *Control and Intelligent Systems*, vol. 42, no. 2, pp. 126-135, 2014.

[4] Q. Qin, Q.-G. Wang, S. S. Ge and C. Yu, "Neural Networks Trained by Randomized Algorithms," *Transactions on Machine Learning and Artificial Intelligence*, vol. 2, no. 1, pp. 01-17, 2014.

[5] Q.-G. Wang, C. Yu and Y. Zhang, "Improved System Identification with Renormalization Group," *ISA Transactions*, Available online 17 Jan 2014.

[6] C. Yu, B.-N. Le, X. Li and Q.-G. Wang, “Randomized Algorithm for Determining Stabilizing Parameter Regions for General Delay Control Systems,” *Journal of Intelligent Learning Systems and Applications*, vol. 5, no. 2, pp. 99-107, 2013.

Conference Papers:

[1] C. Yu, Q.-G. Wang and D. Zhang, “System Identification in Presence of Outliers,” manuscript is accepted by *Euro Mini Conference on Stochastic Programming and Energy Applications (ECSP)*, 2014.

[2] Q.-G. Wang, C. Yu and Y. Zhang, “Model Assessment with Renormalization Group in Statistical Learning,” *10th IEEE International Conference on Control and Automation (ICCA)*, pp. 884-889, 2013.

[3] Q.-G. Wang, C. Yu and Y. Zhang, “Improved System Identification with Renormalization Group,” *10th IEEE International Conference on Control and Automation (ICCA)*, pp. 878-883, 2013.