STATE ESTIMATION OF SPATIO-TEMPORAL PHENOMENA

A Dissertation

by

DAN YU

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Suman Chakravorty
Committee Members,	John Junkins
	Sharath Girimaji
	Swaroop Darbha
Head of Department,	Rodney Bowersox

December 2016

Major Subject: Aerospace Engineering

ABSTRACT

This dissertation addresses the state estimation problem of spatio-temporal phenomena which can be modeled by partial differential equations (PDEs), such as pollutant dispersion in the atmosphere. After discretizing the PDE, the dynamical system has a large number of degrees of freedom (DOF). State estimation using Kalman Filter (KF) is computationally intractable, and hence, a reduced order model (ROM) needs to be constructed first. Moreover, the nonlinear terms, external disturbances or unknown boundary conditions can be modeled as unknown inputs, which leads to an unknown input filtering problem. Furthermore, the performance of KF could be improved by placing sensors at feasible locations. Therefore, the sensor scheduling problem to place multiple mobile sensors is of interest.

The first part of the dissertation focuses on model reduction for large scale systems with a large number of inputs/outputs. A commonly used model reduction algorithm, the balanced proper orthogonal decomposition (BPOD) algorithm, is not computationally tractable for large systems with a large number of inputs/outputs. Inspired by the BPOD and randomized algorithms, we propose a randomized proper orthogonal decomposition (RPOD) algorithm and a computationally optimal RPOD (RPOD*) algorithm, which construct an ROM to capture the input-output behaviour of the full order model, while reducing the computational cost of BPOD by orders of magnitude. It is demonstrated that the proposed RPOD* algorithm could construct the ROM in real-time, and the performance of the proposed algorithms on different advection-diffusion equations.

Next, we consider the state estimation problem of linear discrete-time systems

with unknown inputs which can be treated as a wide-sense stationary process with rational power spectral density, while no other prior information needs to be known. We propose an autoregressive (AR) model based unknown input realization technique which allows us to recover the input statistics from the output data by solving an appropriate least squares problem, then fit an AR model to the recovered input statistics and construct an innovations model of the unknown inputs using the eigensystem realization algorithm. The proposed algorithm outperforms the augumented two-stage Kalman Filter (ASKF) and the unbiased minimum-variance (UMV) algorithm are shown in several examples.

Finally, we propose a framework to place multiple mobile sensors to optimize the long-term performance of KF in the estimation of the state of a PDE. The major challenges are that placing multiple sensors is an NP-hard problem, and the optimization problem is non-convex in general. In this dissertation, first, we construct an ROM using RPOD* algorithm, and then reduce the feasible sensor locations into a subset using the ROM. The Information Space Receding Horizon Control (I-RHC) approach and a modified Monte Carlo Tree Search (MCTS) approach are applied to solve the sensor scheduling problem using the subset. Various applications have been provided to demonstrate the performance of the proposed approach.

DEDICATION

TO MY FAMILY

ACKNOWLEDGEMENTS

I am deeply indebted to many people who have helped and inspired me over the years. First and foremost, I would like to express my deepest respect and gratitude to my advisor, Prof. Suman Chakravorty for supporting and encouraging me throughout my graduate career. His positive outlook and enthusiasm are contagious; his keen insight would always nudge me in the right direction. Especially, I would like to thank Dr. Chakravorty for his great help and support with my career decision. This work would not have been possible without his support.

I would also like to thank my committee members, professor John Junkins, Sharath Girimaji and Darbha Swaroop for their time, encouragement and precious advice. In addition to my committee members, I would like to thank Dr. Sivakumar Rathinam for being defense examiner, and Dr. Robert Skelton for his constructive suggestions. I would like to thank Dr. Jianer Chen for his great courses. I have benefited from all of them over the years as a student or simply through friendly conversations.

I would like to thank the current and past members of Dr. Chakravorty's group: Ali-akbar Agha-mohammadi, Saurav Agarwal, Amirhossein Tamjidi, Dilshad Raihan and Weston Faber for their friendship and for all the discussions over the years. I am especially grateful to Anshu Narang and Xiao Li Bai, for sharing their precious experience, and their persistent help. I have been fortunate to have so many friends who make my Ph.D journey enjoyable.

I would like to thank the staff in the Department of Aerospace Engineering, particularly Karen Knabe and Rose Sauser, for their constant help.

Finally, to my family, I am eternally grateful of their love and support. Being so

far away from home has been difficult both for me and for my parents. I would like to thank my parents for their unconditional love, and their support for every decision I have made. I also wish to thank my grandparents for their encouragement. This dissertation is dedicated to them.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Motivation

Many real-world phenomena can be viewed as spatio-temporal processes which are modeled by partial differential equations (PDEs). For example, the motion of fluids such as precipitation [3], ocean currents [4, 5], air pollution [6] and water flow in a pipe [7] can be described by Navier-Stokes Equations [8]. In addition, the temperature distribution of a large battery pack [9], biological phenomena [10] , power system state-time behavior [11] and many other engineering applications are also modeled by PDEs.

Historically, there has been a lot of theoretical research in the Control Systems community on the estimation and control of systems driven by PDEs [8, 12–15]. A standard approach is to transform PDEs to a set of ordinary differential equations (ODEs) by discretizing the PDEs in space using the finite difference method (FDM) or finite element method (FEM). The discretized system has a large number of degrees of freedom (DOF) which leads to the following problem: state estimation of such a system using standard Kalman Filter (KF) is not computationally tractable. Therefore, a reduced order model (ROM) constructed via model reduction algorithms is used for state estimation. The benefit of using an ROM based KF is that the expensive computations for constructing the ROM can be done offline, and the online state estimation computations are essentially trivial.

In addition, in order to estimate the states of stochastic dynamical system, it is generally assumed that all system parameters, noise covariance, and inputs are known. However, in practice, unknown inputs are often present in the systems due to the unmodeled dynamics, nonlinear terms or external disturbances. The state

estimation of stochastic systems in presence of unknown inputs is known as the unknown input filtering (UIF) problem, and has been investigated for decades.

There are many applications where the unknown inputs can be modeled as a stochastic process. In [16], a stochastic disturbance model is used to force the linearized Navier-Stokes equation, which leads to a simulated flow state with certain second-order statistics closely matching the flow statistics computed from the Direct Numerical Simulation (DNS) of turbulent channel flow. Based on this result, the state estimation of perturbed laminar flows is considered in [17]. It shows that the external disturbances and nonlinear coupling terms can be modeled as unknown stochastic inputs, which perturb the linearized Navier-Stoke equations. Thus, the state estimation problem of such system is transformed into the unknown input filtering problem with stochastic unknown inputs. Also, there is some research that considers Kalman filtering with unknown noise covariances. The process noise is assumed to be white noise with unknown covariance [18, 19], while in our research, the process noise can be colored in time as well. There are also applications of unknown stochastic inputs estimation in signal processing, such as the wideband power spectrum estimation [20], where the problem is to recover the unknown power spectrum of a wide-sense stationary signal from the obtained sub-Nyquist rate samples. In this dissertation, we consider the UIF problem when the unknown inputs can be modeled as stochastic processes.

Another important problem considered in this dissertation is to place multiple mobile sensors to optimize the long-term behavior of the KF in the estimation of the state of a PDE. Since sensors have become much smaller and less expensive in the last few decades, there is increasing attention of using mobile sensors for monitoring spatio-temporal phenomena. Moreover, installing and maintaining stationary sensors are costly, and in scenarios when the phenomena changes in a short dura-

tion, stationary sensors collect limited information. The major challenges are that placing multiple sensors is an NP-hard problem [21], and the optimization problem is non-convex in general. Hence, most algorithms focus on placing sensors which can optimize performance of the current time instant, i.e., a myopic policy.

In this context, the motivation for this research arises from the desire to design a KF based approach for estimating and predicting spatio-temporal phenomena. The proposed approach is capable of handling uncertainties as well as unknown inputs. The performance of the state estimator is optimized by placing multiple mobile sensors to take measurements, and most importantly, the approach can be implemented efficiently. The general structure of the proposed approach is shown in Fig. 1.1.

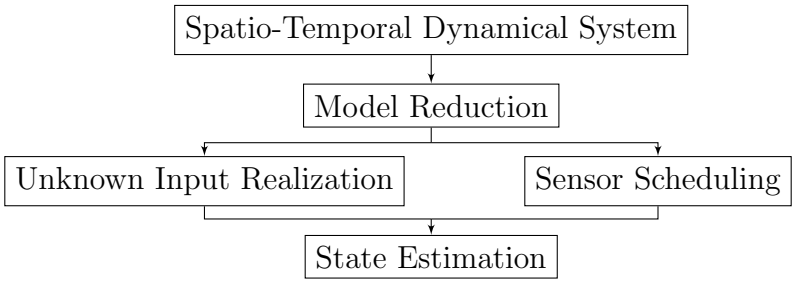


Figure 1.1: State estimation procedure of spatio-temporal phenomena

1.2 Literature Review

In this section, we review the literature most relevant to our research. Section 1.2.1 reviews different model reduction techniques that have been widely used. In Section 1.2.2, we give an overview of the unknown input filtering methods, and in Section 1.2.3, we review the literature relates to the sensor placement problem. The open research problems are enumerated in each subsection respectively.

1.2.1 Model Reduction Methods

Model reduction has attracted considerable attention in the past several decades. It is a technique that constructs a lower-dimensional subspace to approximate the original high-dimensional dynamic system.

The Proper Orthogonal Decomposition (POD), also known as Karhunen-Loeve decomposition or principle component analysis, followed by a Galerkin projection has been used extensively in the Fluids community to produce reduced order models (ROMs) of fluid phenomena such as turbulence and fluid structure interaction [22–24]. A review of the POD is given in [25]. An empirical basis of orthonormal eigenfunctions is obtained from experimental or simulation data, and the original higher-dimensional system is projected onto this basis. The POD modes are optimal in the sense that the energy captured in the ROM are optimized, and the most significant modes are the ones that carry most of the kinetic energy. However, the most energetic modes are not always the most dynamically significant ones. Hence, a major disadvantage of POD technique is that the ROM constructed using POD is not accurate [26]. Techniques to improve performance of POD are reviewed in [27].

In the context of control theory, balanced truncation introduced in [28] has been successfully applied to linear dynamical systems. Balanced truncation yields a stable reduced system with a bounded approximation error. Both the inputs and outputs of the dynamical system are taken into account, and consequently, the ROM captures the input-output behavior of the original system. However, balanced truncation suffers from high computational complexity when generating the controllability and observability Gramians for large scale systems.

To reduce the computational cost of balanced truncation, Balanced POD (BPOD) [29, 30], which is based on the snapshot POD and balanced truncation has been

proposed. Balancing transformations are constructed using the impulse responses of both the primal and adjoint system, and hence, the most controllable and observable modes can be kept in the ROM. In 1978, Kung [31] presented a new model reduction algorithm in conjunction with the singular value decomposition (SVD) technique, and the eigensystem realization algorithm (ERA) [32] was developed based on this technique. The BPOD is equivalent to the ERA procedure [33], and forms the Hankel matrix using the primal and adjoint system simulations as opposed to the input-output data as in ERA. More recently, there has been work on obtaining information regarding the dominant modes of the system based on the snapshot POD, followed by an eigendecomposition of an approximating linear operator, called the dynamic mode decomposition (DMD) [34, 35].

The primary drawback of BPOD and ERA is that for a large scale system, such as that obtained by discretizing a PDE, with a large number of inputs/outputs, the computational burden incurred is very high. There are two main parts to the computation: First is to collect datasets from computationally expensive primal and adjoint simulation in order to generate the Hankel matrix. The second part is to solve the SVD problem for the resulting Hankel matrix.

Improved algorithms have been proposed to reduce the computational complexity of BPOD. For example, [30] proposed an output projection method to address the problem when the number of outputs is large. The outputs are projected onto a small subspace via an orthogonal projection P_s that minimizes the error between the full impulse response and the projected impulse response. However, the method cannot make any claim regarding the closeness of the solution to one that is obtained from the full Hankel matrix, and is still faced with a very high computational burden when both the numbers of inputs and outputs are large. There have also been methods proposed [36] to reduce the number of snapshots. First, the DMD is used to estimate

the slowly decaying modes that dominate the long-term behavior of the impulse responses, and then analytic expressions are formulated for the contribution of these modes. Therefore, there is no need to run the long impulse response simulations. However, the primary problem regarding large number of inputs/outputs remains the same.

Randomized algorithms motivated by problems in large-scale data analysis have been developed recently. The advantage of the randomized algorithm is that: 1) often the execution time or space requirement of a randomized algorithm is smaller than that of the best deterministic algorithm, and 2) it lead to a simpler algorithm to implement. There are two major classes of randomization algorithms used for low-rank matrix approximations and factorizations: random sampling algorithms [37] and random projection algorithms [38, 39].

For a large scale matrix H, random sampling algorithms construct a rank k approximation matrix $\hat{H}$ by choosing and rescaling some columns of H according to certain sampling probabilities [38], so the error satisfies $\|H - \hat{H}\|_F \leq \|H - H_{(k)}\|_F + \epsilon\|H\|_F$, with high probability, where $H_{(k)}$ is a best rank k approximation of H, ϵ is a specified tolerance, and $\|H\|_F$ denotes the Frobenius norm of H. The improved algorithm proposed in [39] is to sample some columns according to leverage scores, where the leverage scores are calculated by performing the SVD of H, so that the error satisfies $\|H - \hat{H}\|_F \leq (1 + \epsilon)\|H - H_{(k)}\|_F$, with high probability.

A recently developed "Scenario Method" for systems and control design [40, 41] can also be related to the random sampling algorithm. For a robust convex optimization problem, the number of optimization variable d is finite, while the number of constraints may be infinite. The scenario method randomly samples N_s convex constraints from the uncountable set of constraints, and with the bound $N_s \geq \frac{2}{\epsilon}(log(\frac{1}{\beta}) + d)$, the optimal solution can be guaranteed to satisfy an ϵ- fraction

of the constraints, with probability no smaller than $1-\beta$, where ϵ and β are design parameters.

In random projection method [37], the large matrix H is projected on to an orthonormal basis Q such that the error satisfies $\|H - QQ^*H\| \leq (1+\epsilon)\|H - H_{(k)}\|$ with high probability, where $\|H\|$ denotes the spectral 2-norm of H, x^* denotes the complex conjugate transpose of x. A Gaussian test matrix Ω is generated, and the orthonormal basis Q is constructed by performing a QR factorization of the matrix product $H\Omega$.

A direct application of both the random sampling algorithm and random projection algorithm would require the full Hankel matrix to be constructed, however, such a construction of the Hankel matrix is computationally prohibitive when the number of inputs/outputs is large. Further, in random sampling algorithm, the leverage scores are calculated by performing the SVD of the Hankel matrix, which is also computationally prohibitive owing to the size of the problem.

In this dissertation, we aim to reduce the computational costs and storage requirements of BPOD while keeping the same accuracy as BPOD. In Section 2, we propose a Randomized POD (RPOD) algorithm, which is closely related to the "Scenario Method". We randomly choose a subset of the input/output trajectories and construct a sub-Hankel matrix, which has the same rank as the full Hankel matrix with a high probability. We derive a bound on the number of input/output trajectories to be sampled, and the derivation of our bound, albeit different from the bound in [40], is nonetheless inspired by the developments in that reference. Similar to the BPOD algorithm, the controllable and observable modes are retained in the ROM, the Markov parameters of the ROM are close to the Markov parameters of the full order system, while the error is bounded. The computations required by RPOD are orders of magnitude cheaper when compared to the BPOD/BPOD output projection

algorithm.

In Section 3, we propose a computationally optimal RPOD (RPOD*) algorithm, which is closely related to the random projection algorithm. The RPOD* algorithm can be viewed as applying the random projection on the full Hankel matrix H twice without constructing the full Hankel matrix H. We believe that RPOD* is the most computational efficient POD algorithm.

1.2.2 Unknown Input Filtering

Observers are dynamic systems that can be used to estimate the state of a plant using its input-output measurements. In some cases, the inputs to the plant are unknown or partially known, which leads to the development of the so-called unknown input observer (UIO). The unknown input observer has been well established for deterministic systems [42–44]. Various methods of building full-order or reduced-order observers have been developed, such as [45–47]. Recently, sliding mode observers have been proposed for systems with unknown inputs [48]. The design parameters and matrices need to be well chosen to satisfy certain conditions in order for the observers to perform well. For systems without the "observer matching" condition being satisfied, a high-gain approach is proposed [49]. The high-gain observers are used as approximate differentiators to obtain the estimates of the auxiliary outputs. In the presence of measurement noise, the high-gain observer amplifies the noise, and extra care needs to be taken when designing the gain matrix.

For stochastic systems, the state estimation problem with unknown inputs is known as unknown input filtering (UIF) problem, and many UIF approaches are based on the Kalman filter [50–52]. When the dynamics of the unknown inputs is available, for example, if it can be assumed to be a wide-sense stationary (WSS) process with known mean and covariance, one common approach called Augmented State

Kalman Filter (ASKF) is used, where the states are augmented with the unknown inputs [53]. To reduce the computational complexity of ASKF, optimal two-stage Kalman filters (OTSKF) and optimal three-stage Kalman filters have been developed to decouple the augmented filter into two parallel reduced-order filters by applying a U-V transformation [54–56]. When no prior information about the unknown input is available, the invalid assumption about the model may have a major adverse effect on the filter's performance, and hence, an unbiased minimum-variance (UMV) filtering technique has been developed [57, 58]. The problem is transformed into finding a gain matrix such that the trace of the estimation error matrix is minimized. The "observer matching" condition needs to be satisfied, and certain algebraic constraints must be satisfied for the unbiased estimator to exist. When the assumed unknown input model used in OTSKF is accurate, the performance of OTSKF is better than UMV algorithm in the sense that the error covariances are smaller, otherwise, UMV algorithm is more accurate than OTSKF.

In Section 4 we propose a new UIF algorithm when the unknown inputs can be treated as a wide-sense stationary process with rational power spectral density, while no other prior information needs to be known. The algorithm is based on the system identification technique, which is more accurate than OTSKF and UMV algorithms, and can tolerate more sensor noise. A milder assumption than the "observer matching" condition needs to be satisfied. Also, the algorithm we propose can be applied to estimate the locations of the unknown inputs.

1.2.3 Sensor Scheduling

The optimal sensor placement problem is to place multiple sensors in a spatial field, which can maximize a performance metric. Optimal sensor placement in spatial field faces the challenges such as placing multiple sensors is NP-hard [21], and the

optimization problem is non-convex in general. As the number of sensors or the number of possible sensor locations increases, an exhaustive search for global optimum is computationally infeasible, so many near-optimal or sub-optimal sensor placement methods have been developed. For the purpose of field estimation, there are two major classes of sensor placement techniques: Gaussian Process (GP) based [59, 60] and mode shape based algorithms [61, 62].

The Gaussian Process (GP) [63] has been widely used to estimate and predict spatial phenomena. For example, in [59], the temperature measurements in two-dimensional space are assumed to be spatially correlated, and can be modeled by a GP. Given measurements at some locations, we can predict the temperature at arbitrary locations. A near-optimal sensor placement approach is developed by adding sensors in sequence, and choosing the next sensor which provides the maximum increase in mutual information. However, for the spatio-temporal phenomena which also evolve with time, predictions using the learned GP model could result in large errors. Hence, current research has focused on modelling the desired phenomena using a spatio-temporal GP, with a mobile sensor network [64–66]. However, in such a case, a more complicated GP model needs to be learned, and the computational complexity of GP regression increases as the number of observations increases.

When the spatio-temporal phenomena can be modeled by PDEs, one research direction focuses on placing the sensors using spatial structure of the underlying phenomena [61, 62]. For example, in [67], sensors are placed using the eigenfunctions of the PDEs and is formulated as an optimization problem to maximize the system observability. In [68–70], the original systems are projected onto lower-dimensional subspace using the POD algorithm. The sensors are placed at the extrema of the POD projection bases, and the ROMs are used for state estimation, which is computationally tractable.

The mobile sensor placement problem can be viewed as an extension of the static sensor placement problem, which allows the sensors to move in time. When the sensors are placed to maximize the performance metric at the next time instant, it results in a myopic sensor scheduling problem. The non-myopic sensor scheduling problem is to optimize the performance metric over a suitably long time-horizon. Myopic scheduling is attractive due to its low computational complexity and ease of implementation, while the use of non-myopic scheduling is imperative for dynamical systems as it can perform significantly better than myopic scheduling. This is because myopic decisions do not take into account the long-term effects of these decisions. Although the sensor scheduling problem has been well studied, applying the non-myopic algorithms on large scale spatio-temporal system is still not computationally tractable. Hence, most mobile sensor placement algorithms are myopic when monitoring spatio-temporal systems.

There are two major classes of non-myopic sensor scheduling algorithms. The first class is based on the exhaustive tree search [71, 72], which requires the enumeration at every iteration. The benefits of the branch-and-bound based tree search algorithm is its wide range of applications, while the size of the decision space poses challenge. Another class of sensor scheduling algorithm is to relax the non-convex optimization problem into a convex problem [73,74]. The advantage of using the convex relaxation algorithm is that once the problem is convex, it can be solved efficiently.

In Section 6, we apply two non-myopic sensor scheduling algorithms to place multiple mobile sensors for spatio-temporal systems and compare the performance with myopic algorithm.

1.3 Contribution

This dissertation focuses on monitoring and predicting spatio-temporal phenomena which can be modeled by partial differential equations (PDEs), such as pollutant dispersion. Three problems are solved in this dissertation. First, new model reduction algorithms are proposed to reduce the computational cost of state estimation using the Kalman Filter (KF). Next, we consider the problem of state estimation in the presence of unknown inputs, and provide a new unknown input realization algorithm. Finally, we present a framework for non-myopic sensor scheduling to optimize the performance of a KF monitoring a spatio-temporal process. The major contributions are as follows:

- **Model Reduction:** We present a new perspective to analyze the relation between the snapshot ensembles and ROM. The concept of computationally optimal snapshot ensembles is introduced, and we propose a RPOD* algorithm which can reduce the computational cost of BPOD by orders of magnitude, while keeping the same accuracy. We also relate the RPOD* algorithm to randomized algorithms.

- **Unknown Input Realization:** We propose an autoregressive (AR) model based unknown input realization technique, which constructs an unknown input model using only the output information. The unknown inputs are assumed to be a wide-sense stationary process with rational power spectral density while no other information needs to be known. The so-called matching condition is avoided, and sufficient conditions for the convergence of estimation error in the presence of unknown inputs are derived. The ROM constructed using RPOD* algorithm is used to reduce the computational complexity. We show that the performance of AR model based algorithm is significantly better than

the OTSKF and UMV algorithms (two other UIF algorithms).

- **Sensor Scheduling:** We propose a three-step framework to place multiple mobile sensors for spatio-temporal systems efficiently. First, an ROM is constructed using RPOD* algorithm, and an optimization problem using the ROM is formulated to reduce the possible locations of sensors into a subset. Two sensor scheduling algorithms are used to place mobile sensors in a receding horizon fashion. The proposed algorithm is tested on moderate to large scale spatio-temporal phenomena.

This research has been reported in several publications. Different POD based model reduction algorithms are reported in [75–78]. The AR model based unknown input realization method is reported in [79, 80].

1.4 Organization

The rest of the dissertation is organized as follows:

Section 2 presents a brief review of existing model reduction techniques which are related to our research, with a simplified analysis of BPOD in a new perspective. The ROM is related to the controllable and observable eigenmodes of the dynamical system, which reveals some fundamental insights into the model reduction technique. A new model reduction algorithm RPOD is presented which can reduce the computational cost required by existing approaches, and several computational results comparing RPOD with BPOD are provided.

Section 3 extends the study of Section 2, and proposes a computationally optimal RPOD algorithm, which can further reduce the computational cost of RPOD. Discussion of implementation issues and comparison with all related model reduction algorithms are included. The computational complexity analysis is provided, and the proposed algorithm is tested on several advection-diffusion problems.

Section 4 considers the design of a stochastic unknown input realization algorithm. The problem is formulated, and necessary conditions for convergence of the estimation error are analyzed. Applications on heat problem and linearized channel flow problem are presented to demonstrate the performance of the proposed algorithm.

Section 5 discusses state estimation of spatio-temporal phenomena using Gaussian Process (GP) model. GPs are briefly reviewed, and simulation results are shown to illustrate the issues with using GP for estimation of spatio-temporal phenomena.

Section 6 proposes a three-step sensor scheduling framework for large scale systems. First, an ROM is constructed using RPOD* algorithm. Then the possible locations are reduced onto a subset using the ROM. Two sensor scheduling approaches are applied to place multiple mobile sensors in a receding horizon control fashion. Computational results are presented to compare performances of different sensor scheduling approaches.

Section 7 concludes the dissertation and discusses future research directions.

2. RANDOMIZED PROPER ORTHOGONAL DECOMPOSITION TECHNIQUE (RPOD)

2.1 Introduction

The main goal and contribution of this dissertation is to develop a model reduction algorithm for large scale systems with a large number of inputs/outputs. As discussed in Section 1, the spatio-temporal phenomena we are interested in can be modeled by partial differential equations (PDEs), and after discretizing the PDEs using finite element or finite difference method, the system can have dimension of $O(10^{5\sim9})$. Standard state estimation and control design methods for such a system are computationally intractable, and hence, the use of a reduced order model (ROM) is necessary.

In this section, we begin by reviewing standard model reduction methods which are related to our work. Proper orthogonal decomposition (POD) with Galerkin projection is a standard model reduction approach, where the original system is projected onto a lower-dimensional subspace using a set of orthogonal bases. In linear control theory, balance truncation is a widely used model reduction approach, which takes into account of both the inputs and outputs, and hence, the ROM can capture the input-output behavior of the original system. Balanced proper orthogonal decomposition (BPOD) algorithm is based on the snapshot POD and balance truncation, and is most related to our work. The goal of this section is to propose a randomized proper orthogonal decomposition (RPOD) algorithm, which reduce the computational cost of BPOD while the ROM retaining almost the same information as BPOD, especially for the system with a large number of inputs/outputs.

The rest of the section is organized as follows. In Section 2.2, we introduce

the main idea of POD via Galerkin projection. In Section 2.3, we introduce the balance truncation approaches, with a detail description of BPOD algorithm. Then we analyze the BPOD algorithm in a simplified fashion in Section 2.4. The simplified analysis is crucial to understanding the fundamental of the RPOD algorithm. We propose the RPOD algorithm in Section 2.5, and we show that the sub-Hankel matrix we construct retains almost the same information as the full Hankel matrix in terms of the numbers and accuracy of the underlying modes. In Section 2.6, we compare the simulation results using RPOD and BPOD for several advection-diffusion equations.

2.2 Preliminaries: POD-Galerkin Projection

POD-Galerkin Projection is a projection method where the dynamical system is projected onto a subspace of the original space. POD provides a method for finding the best approximating subspace to a given set of data in an optimal least-square sense, and Galerkin projection is a standard technique to reduce partial differential equations with a method of lines to a system of ordinary differential equations.

POD Algorithm. Consider a high-dimensional Hilbert space $\mathcal{H} = \Re^N$, where N is large, and a given set of data in $\mathcal{H}$:

$$X = \{x_1(t), x_2(t), \cdots, x_m(t)\} \in \Re^{N \times m}, \tag{2.1}$$

where $t \in [0, T]$. POD method [81] aims to find an orthogonal projection $P_s : \mathcal{H} \to \mathcal{H}_d$ which projects the original data onto a d-dimensional subspace that minimizes the total least-squares distances:

$$\|X - P_s X\|^2 = \sum_{i=1}^{m} \int_0^T \|x_i(t) - P_s x_i(t)\|^2 dt, \tag{2.2}$$

where $\|.\|$ denotes the L_2 norm.

Solving the optimization problem (2.2) leads to an eigenvalue problem:

$$K\phi_i = \lambda_i \phi_i, i = 1, \cdots, N, \tag{2.3}$$

where $K \in \Re^{N \times N}$ is called the kernel and is defined as:

$$K = \sum_{i=1}^{m} \int_0^T x_i(t) x_i(t)' dt, \tag{2.4}$$

where $(.)'$ denotes the transpose of $(.)$.

By definition, K is a symmetric positive semi-definite matrix with real, nonnegative ordered eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N \geq 0$. The optimal subspace $\mathcal{H}_d$ of dimension d is given by $\mathcal{H}_d =$ span $\{\phi_1, \cdots, \phi_d\}$, and the eigenvectors $\phi_i, i = 1, \cdots d$ are called the POD modes. The major results of POD is given as follows [82].

Theorem 1 *Let K be the kernel of the data, and $\lambda_1, \cdots, \lambda_N \geq 0$ be the ordered eigenvalues of K. Then it holds*

$$\min_{\mathcal{H}_d} \|X - P_s X\| = \sum_{i=N-d+1}^{N} \lambda_i, \tag{2.5}$$

where the minimum is taken over all subspaces $\mathcal{H}_d$ of dimension d. Further, the optimal orthogonal projection $P_s : \mathcal{H} \to \mathcal{H}_d$, with $P_s P_s' = I$ is given by:

$$P_s = \sum_{i=1}^{d} \phi_i \phi_i'. \tag{2.6}$$

For a large scale system, solving eigenvalue problem for matrix $K \in \Re^{N \times N}$, where $N = O(10^5 \sim 10^9)$ is computationally intractable. Therefore, in [83], the method of snapshots is introduced to approximate the kernel matrix K without explicitly calculating (2.4). Snapshots are constructed from the trajectories of the dynamical

system by evaluating them at certain discrete time instances $t_1, \cdots t_m \in [0, T]$, and the kernel matrix is approximated by:

$$K = \sum_{i=1}^{m} x(t_i)x(t_i)'. \tag{2.7}$$

where $x(t_i)$ is the instantaneous system state at time t_i, and $m \ll n$ is the number of snapshots. Denote the snapshot ensemble as $X = \Big(x(t_1), \cdots, x(t_m)\Big) \in \Re^{N\times m}$. In the method of snapshots, instead of solving eigenvalue problem for $K = XX' \in \Re^{N\times N}$, one considers to solve an $m \times m$ eigenvalue problem:

$$X'Xv_j = \lambda_j v_j, j = 1, \cdots, m, \tag{2.8}$$

where $\lambda_1 \geq \cdots \geq \lambda_m$ are the ordered eigenvalues, and v_j are the corresponding eigenvectors. The POD modes are given by:

$$\phi_j = \frac{1}{\sqrt{\lambda_j}} X v_j, j = 1, \cdots, d. \tag{2.9}$$

After constructing POD modes $\Phi = \{\phi_1, \cdots, \phi_d\}$, the ROM is obtained by projecting dynamical system onto the POD modes via Galerkin projection.

Galerkin Projection. Consider a dynamical system which is governed by PDE with variable $x(t) \in \mathcal{H}$. If the spatial domain is Ω, then $\mathcal{H}$ is a space of functions defined on Ω. The dynamical system satisfies

$$\dot{x}(t) = f(x(t)), \tag{2.10}$$

where $f : \mathcal{H} \to \mathcal{H}$ is a spatial differential operator. Given a subspace $\mathcal{H}_d \subset \mathcal{H}$, which is constructed via POD, Galerkin projection specifies a dynamical system which

evolves on $\mathcal{H}_d$ and approximates the original dynamical system. The ROM is given by:

$$\dot{x}_d(t) = P_s f(x_d(t)), \tag{2.11}$$

where $P_s : \mathcal{H} \to \mathcal{H}_d$ is the orthogonal projection map, and $x_d(t) \in \mathcal{H}_d$.

Let $\phi_k \in \mathcal{H}, k = 1, \cdots d$ be an orthogonal basis for the subspace, then $x_d(t)$ could be represented by

$$x_d(t) = \sum_{k=1}^{d} a_k(t)\phi_k, \tag{2.12}$$

where ϕ_k are time-independent basis functions, and $a_k(t)$ are the corresponding time coefficients. Therefore, the ROM can be obtained using POD modes, and the evolution of the time coefficients are governed by r ordinary differential equations (ODEs) as follows

$$\dot{a}_k(t) = \phi_k' f(x_d(t)) = \phi_k' f(\sum_{j=1}^{d} a_j(t)\phi_j), k = 1, \cdots d. \tag{2.13}$$

Discussion. The advantages of using POD approach are summarized as follows. First, the POD modes are constructed from the sampled data, and hence, the approach can be applied to linear and nonlinear system, as well as to parametrically varying systems. Also, solving eigenvalue problem takes time $O(m^3)$, where m is the number of snapshots, and $m \ll N$. However, POD modes are sensitive to the snapshot ensemble X, and could only capture the system behavior for a particular time range for particular inputs. A good snapshot set selection leads to a better performance of ROM, while there is no guidance on how to select the snapshots.

Furthermore, POD approach only takes into account the system inputs, and hence, inefficient modes may be obtained. From the control point of view, the input-output relationship is important, and needs to be taken into consideration.

2.3 Preliminaries: Balanced Truncation and BPOD

Balanced Truncation. Balanced truncation is a standard model reduction algorithm which is introduced by [84]. Both the inputs and outputs are taken into account, and hence, the ROM can capture the input-output behavior of the original dynamical system.

For a linear time-invariant (LTI) stable discrete-time input-output system:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k, \\ y_k &= Cx_k, \end{aligned} \tag{2.14}$$

where $x_k \in \Re^N$, $u_k \in \Re^p$, and $y_k \in \Re^q$ is the state vector, input vector, and output vector at time t_k respectively.

The adjoint/dual system is defined as:

$$\begin{aligned} z_{k+1} &= A'z_k + C'v_k, \\ \hat{y}_k &= B'z_k, \end{aligned} \tag{2.15}$$

where z_k is the adjoint state vector at time t_k.

Balanced truncation aims to find a transformation matrix T_s, such that the controllability and observability Gramians

$$W_c = \sum_{k=0}^{\infty} A^k BB'(A')^k, W_o = \sum_{k=0}^{\infty} (A')^k C'CA^k \tag{2.16}$$

are equal and diagonal.

The transformation matrix T_s is constructed by solving the eigenvalue problem:

$$W_{co}\phi_j = \lambda_j \phi_j, j = 1, \cdots n, \tag{2.17}$$

where $W_{co} = W_c W_o$, $\lambda_1 \geq \cdots \geq \lambda_n$, and $T_s = \left(\phi_1, \cdots, \phi_r\right) \in \Re^{n \times r}$ contains the first r eigenmodes.

The ROM is given by:

$$\begin{aligned} a_{k+1} &= T_s^{-1} A T_s a_k + T_s^{-1} B u_k, \\ y_k^r &= C T_s a_k. \end{aligned} \tag{2.18}$$

And the controllability and observability Gramians of the ROM are:

$$\tilde{W}_c = \tilde{W}_o = \Sigma, \tag{2.19}$$

where Σ is a diagonal matrix, with element $\sigma_i, i = 1, \cdots, r$, and $\sigma_i = \sqrt{\lambda_i}$ are known as the Hankel singular values of the system.

Balanced truncation has the following property.

Theorem 2 *The error bound on the output is:*

$$\|y_k - y_k^r\| \leq 2 \sum_{i=r+1}^{n} \sigma_i \|u_k\|. \tag{2.20}$$

Same as POD approach, generating the controllability and observability Gramians for large scale system is computationally expensive, and hence, in [29, 30], balanced truncation using the method of snapshots are proposed which do not require to compute the Gramians, and BPOD algorithm proposed in [30] is reviewed here.

BPOD Algorithm. BPOD algorithm constructs the snapshot ensembles X_b, Z_b by perturbing the primal system (2.14) and adjoint system (2.15) with impulse respectively, and it is shown in [85] that the Gramians in (2.16) can be approximated as

$$W_c \approx X_b X_b', W_o \approx Z_b Z_b'. \tag{2.21}$$

BPOD constructs two bi-orthogonal transformation matrices T_b, S_b by solving the singular value decomposition (SVD) problem of

$$H_b = Z_b' X_b, \tag{2.22}$$

where H_b is known as Hankel matrix.

The BPOD algorithm is summarized in Algorithm 1.

Discussion. Recall the impulse response snapshot ensembles collected in the BPOD are:

$$\begin{aligned} X_b &= \underbrace{[x_1(t_1), \cdots, x_p(t_1), \cdots, x_1(t_m), \cdots, x_p(t_m)]}_{N \times pm}, \\ Z_b &= \underbrace{[z_1(\hat{t}_1), \cdots, z_q(\hat{t}_1), \cdots, z_1(\hat{t}_n), \cdots, z_q(\hat{t}_n)]}_{N \times qn}, \end{aligned} \tag{2.29}$$

and the Hankel matrix is:

$$H_b = Z_b' X_b \in \Re^{qn \times pm}. \tag{2.30}$$

There are two main parts to the computation:

1. The primal and adjoint snapshot ensembles $X_b \in \Re^{N \times pm}, Z_b \in \Re^{N \times qn}$, and

Algorithm 1 BPOD Algorithm

1. Collect impulse response X_b of the primal system (2.14):
   Use $b_i, i = 1, \cdots, p$ as initial conditions for (2.14) with $u_k = 0$. Take m snapshots at discrete times $t_1, t_2, \cdots, t_m$, and

$$X_b = [x_1(t_1), \cdots, x_p(t_1), \cdots, x_1(t_m), \cdots, x_p(t_m)], \tag{2.23}$$

   where $x_i(t_k)$ is the state snapshot at time t_k with b_i as the initial condition, $k = 1, 2, \cdots, m$ and $i = 1, 2, \cdots, p$.

2. Collect impulse response Z_b of the adjoint system (2.15):
   Use $c_j', j = 1, \cdots, q$ as initial conditions for (2.15) with $v_k = 0$. Take n snapshots at time step $\hat{t}_1, \hat{t}_2, \cdots, \hat{t}_n$, and

$$Z_b = [z_1(\hat{t}_1), \cdots, z_q(\hat{t}_1), \cdots, z_1(\hat{t}_n), \cdots, z_q(\hat{t}_n)], \tag{2.24}$$

   where $z_j(\hat{t}_k)$ is the state snapshot of the adjoint system at time $\hat{t}_k$ with c_j' as the initial condition, $k = 1, 2, \cdots, n$ and $j = 1, 2, \cdots, q$.

3. Construct Hankel matrix H_b:

$$H_b = Z_b' X_b, \tag{2.25}$$

4. Solve the SVD problem of H_b:

$$H_b = \begin{pmatrix} L_b & L_1 \end{pmatrix} \begin{pmatrix} \Sigma_b & 0 \\ 0 & \Sigma_1 \end{pmatrix} \begin{pmatrix} R_b' \\ R_1' \end{pmatrix}, \tag{2.26}$$

   where Σ_b contains the first l non-zero singular values, and (L_b, R_b) are the corresponding left and right singular vectors. Σ_1 contains the rest of the singular values.

5. Construct the BPOD bases:

$$\begin{aligned} T_b &= X_b R_b \Sigma_b^{-1/2}, \\ S_b &= \Sigma_b^{-1/2} L_b' Z_b'. \end{aligned} \tag{2.27}$$

6. The ROM is:

$$\begin{aligned} A_b &= S_b A T_b, \\ B_b &= S_b B, \\ C_b &= C T_b. \end{aligned} \tag{2.28}$$

hence, the construction of H_b takes time $O(pqmnN)$.

2. The computational cost to solve the SVD of H_b is $O(\min\{p^2m^2qn, pmq^2n^2\})$.

For the large scale system with a large number of inputs/outputs considered in this work, p, q are large, for example, $p = q = N$. Therefore, it is computationally expensive to construct the Hankel matrix and solve SVD problem of resulting Hankel matrix, and the storage of the data set X_b, Z_b, H_b also causes problem. Thus, we are interested in developing model reduction algorithm which can reduce computational cost require by BPOD while retaining same information as BPOD.

We start from a simplified analysis of BPOD, which discusses the relationship between the BPOD snapshot ensembles and BPOD ROM, and explain what do we mean by the "information" retained in BPOD ROM.

2.4 Simplified Analysis

In this section, we illustrate in a simplified fashion how the transformation bases and Markov parameters of the ROM constructed using BPOD algorithm can be related to the controllable and observable modes of the dynamical system. The simplified analysis is critical to understand the intuition behind the proposed RPOD algorithm in Section 2.5 and RPOD* algorithm in Section 3.

Consider the dynamical system given in (2.14), we start from the following assumption.

Assumption 1 *Assume that A is diagonalizable and stable.*

Under Assumption 1, A could be diagonalized using its eigenvalues and eigenvectors, and let

$$A = V\Lambda U', \tag{2.31}$$

where Λ are the eigenvalues, (V, U) are the corresponding right and left eigenvectors.

For the discrete-time LTI system (2.14), the system is controllable if

$$\mathcal{C} = \begin{pmatrix} B & AB & \cdots & A^{N-1}B \end{pmatrix} \tag{2.32}$$

has full rank (i.e., rank $(\mathcal{C}) = N$), and the system is observable if the adjoint system (2.15) is controllable.

The controllability and observability could also be tested using the eigenvectors of the dynamical system, and the definition of the uncontrollable/unobservable eigenmodes is given as follows.

Definition 1 *A mode (Λ_i, U_i, V_i) is uncontrollable if $U_i' B = 0$, and is unobservable if $CV_i = 0$, where (Λ_i, V_i, U_i) is the i^{th} eigenvalue-eigenvector pair of matrix A.*

From Definition 1, we can partition the eigenvalues and eigenvectors (Λ, V, U) into four parts:

$$A = \begin{pmatrix} V_{co} & V_{c\bar{o}} & V_{\bar{c}o} & V_{\bar{c}\bar{o}} \end{pmatrix} \begin{pmatrix} \Lambda_{co} & & & \\ & \Lambda_{c\bar{o}} & & \\ & & \Lambda_{\bar{c}o} & \\ & & & \Lambda_{\bar{c}\bar{o}} \end{pmatrix} \begin{pmatrix} U_{co}' \\ U_{c\bar{o}}' \\ U_{\bar{c}o}' \\ U_{\bar{c}\bar{o}}' \end{pmatrix}, \tag{2.33}$$

where $(\Lambda_{co}, V_{co}, U_{co})$ are the controllable and observable modes, $(\Lambda_{c\bar{o}}, V_{c\bar{o}}, U_{c\bar{o}})$ are the controllable but unobservable modes, $(\Lambda_{\bar{c}o}, V_{\bar{c}o}, U_{\bar{c}o})$ are the uncontrollable but observable modes, and $(\Lambda_{\bar{c}\bar{o}}, V_{\bar{c}\bar{o}}, U_{\bar{c}\bar{o}})$ are the uncontrollable and unobservable modes.

We make the following assumptions.

Assumption 2 *Denote the number of the controllable and observable modes as l, and $l \ll N$.*

Assumption 3 $U'_{\bar{c}o}B = 0, U'_{\bar{c}\bar{o}}B = 0, CV_{c\bar{o}} = 0, CV_{\bar{c}\bar{o}} = 0.$

Assumption 4 $U'_{\bar{c}o}B = \epsilon C_1, U'_{\bar{c}\bar{o}}B = \epsilon C_2, CV_{c\bar{o}} = \epsilon C_3, CV_{\bar{c}\bar{o}} = \epsilon C_4$, *where ϵ is a small number, C_1, C_2, C_3, C_4 are constant matrices. And if $\|U'_{co}B\| = O(\|C_5\|)$, $\|CV_{co}\| = O(\|C_5\|)$, then $\|C_1\|, \|C_2\|, \|C_3\|, \|C_4\| = O(\|C_5\|)$, where C_5 is a constant matrix.*

Discussion on Assumptions. For most of the practical applications we consider, Assumption 1 is satisfied. Assumption 2 needs to be satisfied for the system to have an ROM, this assumption is typically satisfied for a large-scale system. It should be noticed that from Definition 1, Assumption 3 is the statement of controllability/observability of the different modes of the system. However, in practice, $U'_{\bar{c}o}B, CV_{c\bar{o}}$ are never exactly zero, and hence, in Assumption 4, we assume that $\|U'_{\bar{c}o}B\| \propto \epsilon, \|CV_{\bar{c}o}\| \propto \epsilon$, where ϵ is small.

In the following, BPOD algorithm is analyzed under Assumption 1, 2 and 3, and the formal proof using perturbation theory is provided in Section 3.

First, we construct a modal BPOD ROM by projecting the BPOD bases (T_b, S_b) onto the ROM eigenspace as in Algorithm 2.

Under Assumption 1, 2 and 3, we have the following result.

Theorem 3 *Denote $(\hat{A}_b, \hat{B}_b, \hat{C}_b)$ as the modal ROM constructed using Algorithm 2,*

Algorithm 2 BPOD modal ROM Algorithm

1. Construct BPOD ROM (A_b, B_b, C_b) and BPOD bases (T_b, S_b) using BPOD Algorithm 1.

2. Solve the eigenvalue problem of A_b:

$$A_b = P_b \Lambda_b P_b^{-1}, \tag{2.34}$$

   where Λ_b are the eigenvalues, and P_b are the corresponding eigenvectors.

3. Construct BPOD modal bases:

$$\Phi_b = P_b^{-1} S_b, \Psi_b = T_b P_b, \tag{2.35}$$

4. The modal ROM is:

$$\hat{A}_b = \Phi_b A \Psi_b, \hat{B}_b = \Phi_b' B, \hat{C}_b = C \Psi_b. \tag{2.36}$$

(Φ_b, Ψ_b) *are BPOD modal bases. Then*

$$\begin{aligned} \hat{A}_b &= \Phi_b A \Psi_b = \Lambda_{co}, \\ \hat{B}_b &= \Phi_b' B = U_{co}' B, \\ \hat{C}_b &= C \Psi_b = C V_{co}, \end{aligned} \tag{2.37}$$

where $(\Lambda_{co}, U_{co}, V_{co})$ *are the controllable and observable modes of the system, and the Markov parameters*

$$\hat{C}_b \hat{A}_b^i \hat{B}_b = C A^i B, i = 1, 2, \cdots . \tag{2.38}$$

Proof 1 *Consider the snapshots in the primal snapshot ensemble (2.23),*

$$x_i(t_k) = A^{t_k} b_i, \tag{2.39}$$

where $i = 1, \cdots, p$ *and* $k = 1, \cdots, m$. *Hence,*

$$\begin{pmatrix} x_1(t_k), & \cdots, & x_p(t_k) \end{pmatrix} = A^{t_k} B, \tag{2.40}$$

and the snapshot ensemble X_b *can be written as:*

$$X_b = \begin{pmatrix} A^{t_1} B & A^{t_2} B & \cdots & A^{t_m} B \end{pmatrix}. \tag{2.41}$$

Under Assumption 1, A *could be partitioned using its eigenvalue and eigenvectors, and written as in (2.33). Under Assumption 3,* $U'_{\bar{c}o} B = 0, U'_{\bar{c}\bar{o}} B = 0, CV_{c\bar{o}} = 0, CV_{\bar{c}\bar{o}} = 0$. *Therefore,*

$$\begin{aligned} A^{t_k} B &= \begin{pmatrix} V_{co} & V_{c\bar{o}} \end{pmatrix} \begin{pmatrix} \Lambda_{co} & \\ & \Lambda_{c\bar{o}} \end{pmatrix}^{t_k} \begin{pmatrix} U'_{co} \\ U'_{c\bar{o}} \end{pmatrix} B, \\ &= \begin{pmatrix} V_{co} & V_{c\bar{o}} \end{pmatrix} \begin{pmatrix} \alpha^k_{co} \\ \alpha^k_{c\bar{o}} \end{pmatrix}, \end{aligned} \tag{2.42}$$

where $\alpha^k_{co}, \alpha^k_{\bar{c}o}$ *are the coefficient matrices. Substitute (2.42) into (2.41), and the primal snapshot ensemble* X_b *can be written as:*

$$X_b = \begin{pmatrix} V_{co} & V_{c\bar{o}} \end{pmatrix} \begin{pmatrix} \alpha^b_{co} \\ \alpha^b_{c\bar{o}} \end{pmatrix}, \tag{2.43}$$

where $\alpha^b_{co}, \alpha^b_{c\bar{o}}$ *the coefficient matrices. Similarly, the adjoint snapshot ensemble* Z_b

can be written as:

$$Z_b = \begin{pmatrix} U_{co} & U_{\bar{c}o} \end{pmatrix} \begin{pmatrix} \beta^b_{co} \\ \beta^b_{\bar{c}o} \end{pmatrix}, \tag{2.44}$$

where $\beta^b_{co}, \beta^b_{\bar{c}o}$ *are some coefficient matrices.*

Hence, the Hankel matrix

$$\begin{aligned} Z'_b X_b &= ((\beta^b_{co})' U'_{co} + (\beta^b_{\bar{c}o})' U'_{\bar{c}o})(V_{co}\alpha^b_{co} + V_{c\bar{o}}\alpha^b_{c\bar{o}}), \\ &= (\beta^b_{co})' \alpha^b_{co}, \end{aligned} \tag{2.45}$$

where $U'_{co}V_{c\bar{o}} = 0, U'_{\bar{c}o}V_{co} = 0, U'_{\bar{c}o}V_{c\bar{o}} = 0$, $U'_{co}V_{co} = I_{l\times l}$ *since the left and right eigenvectors are orthogonal.*

Under Assumption 3, rank $(Z'_b X_b) = l$*, i.e., there are exactly* l *non-zero singular values in the resulting SVD problem, thus,*

$$Z'_b X_b = L_b \Sigma_b R'_b, \tag{2.46}$$

where $\Sigma_b \in \Re^{l\times l}$ *are the* l *non-zero singular values and* (L_b, R_b) *are the corresponding left and right singular vectors. Moreover, it can be proved that*

$$Z'_b A X_b = (\beta^b_{co})' \Lambda_{co} \alpha^b_{co}. \tag{2.47}$$

Now, consider the BPOD ROM:

$$\begin{aligned} A_b &= S_b A T_b = \Sigma_b^{-1/2} L_b' (Z_b' A X_b) R_b \Sigma_b^{-1/2} \\ &= \underbrace{\Sigma_b^{-1/2} L_b' (\beta_{co}^b)'}_{P_b} \Lambda_{co} \underbrace{\alpha_{co}^b R_b \Sigma_b^{-1/2}}_{\hat{P}_b}. \end{aligned} \tag{2.48}$$

We show that Λ_{co} are the eigenvalues of A_b, and P_b are the eigenvectors as follows:

$$\begin{aligned} \underbrace{P_b}_{l\times l} \underbrace{\hat{P}_b}_{l\times l} &= \Sigma_b^{-1/2} L_b' \underbrace{(\beta_{co}^b)' \alpha_{co}^b}_{Z_b' X_b} R_b \Sigma_b^{-1/2}, \\ &= \Sigma_b^{-1/2} L_b' L_b \Sigma_b R_b' R_b \Sigma_b^{-1/2} = I. \end{aligned} \tag{2.49}$$

Also,

$$\hat{P}_b P_b = \alpha_{co}^b R_b \Sigma_b^{-1/2} \Sigma_b^{-1/2} L_b' (\beta_{co}^b)' = \alpha_{co}^b ((\beta_{co}^b)' \alpha_{co}^b)^+ (\beta_{co}^b)' = I, \tag{2.50}$$

where $(.)^+$ denotes the pseudoinverse of $(.)$. Hence, $\hat{P}_b = P_b^{-1}$ and from (2.48),

$$\Lambda_{co} = \underbrace{(P_b^{-1} S_b)}_{\Phi_b} A \underbrace{(T_b P_b)}_{\Psi_b}. \tag{2.51}$$

We prove that Ψ_b, Φ_b are bi-orthogonal as follows.

$$\begin{aligned} \Psi_b \Phi_b &= T_b P_b P_b^{-1} S_b = T_b S_b, \\ \Phi_b \Psi_b &= P_b^{-1} S_b T_b P_b = P_b^{-1} P_b = I, \end{aligned} \tag{2.52}$$

where (T_b, S_b) are BPOD bases, and are bi-orthogonal.

Hence, the modal BPOD bases

$$\Psi_b = T_b P_b = X_b R_b \Sigma_b^{-1/2} \Sigma_b^{-1/2} L_b'(\beta_{co}^b)' = (V_{co}\alpha_{co}^b + V_{c\bar{o}}\alpha_{c\bar{o}}^b) R_b \Sigma_b^{-1} L_b'(\beta_{co}^b)'$$
$$= V_{co}\hat{P}_b P_b + V_{c\bar{o}} C_6 = V_{co} + V_{c\bar{o}} C_6, \tag{2.53}$$

where $C_6 = \alpha_{c\bar{o}}^b R_b \Sigma_b^{-1} L_b'(\beta_{co}^b)'$. *Similarly,*

$$\Phi_b = P_b^{-1} S_b = U_{co}' + C_7 U_{\bar{c}o}', \tag{2.54}$$

where $C_7 = \alpha_{co}^b R_b \Sigma_b^{-1} L_b'(\beta_{\bar{c}o}^b)'$. *The modal BPOD ROM constructed using* (Ψ_b, Φ_b) *is:*

$$\begin{aligned}
\hat{A}_b &= \Phi_b A \Psi_b = \Lambda_{co}, \\
\hat{B}_b &= \Phi_b' B = U_{co}' B + C_7 U_{\bar{c}o}' B = U_{co}' B, \\
\hat{C}_b &= C\Psi_b = CV_{co} + CV_{c\bar{o}} C_6 = CV_{co}.
\end{aligned} \tag{2.55}$$

And the Markov parameters of the ROM are:

$$\hat{C}_b \hat{A}_b^i \hat{B}_b = CV_{co}\Lambda_{co}^i U_{co}' B = CA^i B. \tag{2.56}$$

Discussion on Theorem 3. Recall the impulse response snapshot ensembles collected in the BPOD are:

$$X_b = \underbrace{V_{co}}_{N\times l}\underbrace{\alpha_{co}^b}_{l\times pm} + \underbrace{V_{c\bar{o}}\alpha_{c\bar{o}}^b}_{N\times pm}, Z_b = \underbrace{U_{co}}_{N\times l}\underbrace{\beta_{co}^b}_{l\times qn} + \underbrace{U_{\bar{c}o}\beta_{\bar{c}o}^b}_{N\times qn}, \tag{2.57}$$

From the development of Theorem 3, we see that the modal BPOD ROM given in (2.55) is completely determined by the l controllable and observable modes and

is invariant to the data X_b and Z_b, i.e., as long as the snapshot ensembles can be written as:

$$\underbrace{X_{opt}}_{N\times m} = \underbrace{V_{co}}_{N\times l}\underbrace{\alpha_{opt}}_{l\times m} + \underbrace{V_{c\bar{o}}\bar{\alpha}_{opt}}_{N\times m}, \underbrace{Z_{opt}}_{N\times n} = \underbrace{U_{co}}_{N\times l}\underbrace{\beta_{opt}}_{l\times n} + \underbrace{U_{\bar{c}o}\bar{\beta}_{opt}}_{N\times n}, \tag{2.58}$$

where $\alpha_{opt}, \beta_{opt}$ are rank l constant matrices, and $\bar{\alpha}_{opt}, \bar{\beta}_{opt}$ are some constant matrices of suitable dimensions, then under Assumption 1, 2 and 3, the following corollary holds.

Corollary 1 *Denote $(A_{opt}, B_{opt}, C_{opt})$ as the modal ROM constructed using Algorithm 2 with snapshot ensembles X_{opt}, Z_{opt} as in (2.58). Then*

$$A_{opt} = \Lambda_{co}, B_{opt} = U'_{co}B, C_{opt} = CV_{co}, \tag{2.59}$$

where $(\Lambda_{co}, U_{co}, V_{co})$ are the controllable and observable modes of the system, and the Markov parameters $C_{opt}A^i_{opt}B_{opt} = CA^iB, i = 1, 2, \cdots$.

Proof 2 *Corollary 1 can be proved by replacing X_b, Z_b in Theorem 3 with X_{opt}, Z_{opt}, and the proof is omitted here.*

We make the following observations.

Remark 1 *1) The snapshot ensembles do not have to be collections of the impulse responses as in BPOD. 2) Only l snapshots may be enough to extract all the controllable and observable modes of the system.*

Bearing this observation in mind, in the next section, we introduce the RPOD algorithm which generates subsets of snapshot ensembles, such that the Corollary 1 holds.

2.5 RPOD Algorithm

In this section, we introduce a randomized proper orthogonal decomposition (RPOD) algorithm, which randomly chooses a small subset of the inputs/outputs, and constructs a sub-Hankel matrix from the full Hankel matrix, such that Corollary 1 holds. The RPOD algorithm is based on BPOD, and is related to the random sampling algorithm.

First, the random sampling algorithm [37] is briefly reviewed here.

Random Sampling Algorithm. For a large scale matrix H, random sampling algorithm construct a rank k approximation matrix $\hat{H}$ by choosing and rescaling some columns of H according to certain sampling probabilities, so the error satisfies $\|H - \hat{H}\|_F \leq \|H - H_{(k)}\|_F + \epsilon\|H\|_F$ with high probability, where $H_{(k)}$ is a best rank k approximation of H, and ϵ is a specified tolerance.

From the discussion in Section 2.4, we see that Hankel matrix $H \in \Re^{qn \times pm}$ is a large matrix with small rank l. Therefore, we are motivated from random sampling algorithm that a sub-Hankel matrix could be constructed by randomly choosing some columns and rows from H, while all the controllable and observable modes will be preserved in the sub-Hankel matrix. However, the construction of the full Hankel matrix is required, which is computationally prohibitive when the number of inputs/outputs is large. Therefore, we construct the sub-Hankel matrix in the following procedure.

Consider the stable linear system (2.14), we randomly choose r columns from B according to the uniform distribution, denoted as $\hat{B}$, and randomly choose s rows from C with uniform distribution, denoted as $\hat{C}$. Denote $(.)_{(:,i)}$ as the ith column of $(.)$, and $(.)_{(j,:)}$ as the jth row of $(.)$.

Definition 2 *Define an M_1-block as*

$$X_i = [x_i(t_1), \cdots, x_i(t_{M_1})], \tag{2.60}$$

where $x_i(t_k)$ is the state snapshot at time t_k with b_i as the initial condition, $k = 1, 2, \cdots, M_1$ and $i = 1, 2, \cdots, p$.

Similarly, define an M_2-block $Y_i = [y_i(\hat{t}_1), \cdots, y_i(\hat{t}_{M_2})]$, where $y_i(\hat{t}_k)$ is the state snapshot of the adjoint system at time $\hat{t}_k$ with c_i' as the initial condition, $k = 1, 2, \cdots, M_2$ and $i = 1, 2, \cdots, q$.

The original Hankel matrix H was previously defined in (2.25). The reduced order Hankel matrix $\hat{H}$ is then constructed using $\hat{B}$, $\hat{C}$, and it essentially is equivalent to choosing a suitable random subset of the M_1-blocks and M_2-blocks of the primal/ adjoint responses, namely $\hat{X}$ and $\hat{Y}$ to generate the sub-Hankel matrix $\hat{H} = \hat{Y}'\hat{X}$. The RPOD procedure is summarized in Algorithm 3.

First, we provide a general result regarding randomly choosing a rank "l" submatrix from a large rank "l" matrix. Suppose $W \in \Re^{N \times a}$ is a rank l matrix, and suppose that W is spanned by the vectors $\{v_1, v_2, \cdots v_l\}$, where $v_i \in \Re^N$, $l \ll N, a$. Let $W^{(i)}$ denote the set of columns of W that contain the vector v_i. Let

$$\epsilon_i = \frac{\textbf{no.}(W^{(i)})}{N}, \tag{2.61}$$

denote the fraction of the columns in W in which vector v_i is present. Further let

$$\bar{\epsilon} = \min_i \epsilon_i, \tag{2.62}$$

and note that $\bar{\epsilon} > 0$.

Algorithm 3 RPOD Algorithm

1. Pick $c_i \in \{1, \cdots, p\}$ with probability $P[c_i = k] = \frac{1}{p}, k = 1, \cdots, p$, $i = 1, \cdots, \hat{p}$.
2. Pick $r_j \in \{1, \cdots, q\}$ with probability $P[r_j = k] = \frac{1}{q}, k = 1, \cdots, q$, $j = 1, \cdots, \hat{q}$.
3. Set $\hat{B}_{(:,i)} = B_{(:,c_i)}$, $\hat{C}_{(j,:)} = C_{(r_j,:)}$.
4. Use $\hat{B}_{(:,i)}, i = 1, \cdots, \hat{p}$ as the initial conditions for the primal simulation, collect the snapshots at $t = t_1, \cdots, t_{M_1}$, denoted as $\hat{X}$.
5. Use $\hat{C}'_{(j,:)}, j = 1, \cdots, \hat{q}$ as the initial conditions for the adjoint simulation, collect the snapshots at $t = \hat{t}_1, \cdots, \hat{t}_{M_2}$, denoted as $\hat{Y}$.
6. Construct the reduced order Hankel matrix $\hat{H} = \hat{Y}'\hat{X}$.
7. Solve the SVD problem of $\hat{H} = \hat{U}_p\hat{\Sigma}_p\hat{V}_p$.
8. Construct the BPOD basis: $\hat{T}_r = \hat{X}\hat{V}_p\hat{\Sigma}_p^{-1/2}$, $\hat{T}_l = \hat{Y}'\hat{U}'_p\hat{\Sigma}_p^{-1/2}$.
9. Construct the matrix: $\tilde{A} = \hat{T}'_l A \hat{T}_r$, and $(\hat{\Lambda}, \hat{P})$ are the eigenvalues and eigenvectors of $\tilde{A}$.
10. Construct the RPOD basis: $\hat{\Phi}'_{ij} = \hat{P}^{-1}\hat{T}'_l$ and $\hat{\Psi}_{ij} = \hat{T}_r\hat{P}$.
11. The ROM is: $\hat{A}_r = \hat{\Phi}'_{ij} A \hat{\Psi}_{ij}$, $\hat{B}_r = \hat{\Phi}'_{ij} B$, $\hat{C}_r = C\hat{\Psi}_{ij}$.

Lemma 1 *Let M columns be sampled uniformly from among the columns of the matrix W without replacement, and denote the sampled sub-matrix by $\hat{W}$. Let $(\Omega, \mathcal{F}, P_f)$ denote the underlying probability space for the experiment. Given any $\beta > 0$, if the number M is chosen such that*

$$M > max(l, \frac{1}{\bar{\epsilon}} log(\frac{l}{\beta})), \tag{2.63}$$

then $P_f(\rho(\hat{W}) < l) < \beta$, where $\rho(\hat{W})$ denotes the rank of the sampled matrix $\hat{W}$.

Proof 3 *Let $\hat{W}(\omega) = \{W_1(\omega), \cdots W_M(\omega)\}$ denote a random M-choice from the columns of W. If the ensemble $\hat{W}$ has rank less than l then note that at least one of the vectors v_i has to be absent from the ensemble. Define the events*

$$G = \{\omega \in \Omega : \rho(\hat{W}(\omega)) < l\}, \text{ and} \tag{2.64}$$

$$G_i = \{\omega \in \Omega : W_k(\omega) \in \tilde{W}^{(i)}, \forall k\}, \tag{2.65}$$

where $\tilde{W}^{(i)}$ denotes the complement set of columns in W to the set $W^{(i)}$. Due to the fact that the ensemble $\hat{W}$ is rank deficient if all of the columns of $\hat{W}$ are sampled from at least one of the sets $\tilde{W}^{(i)}$, and the fact that if $\hat{W}$ is rank deficient, all the columns of $\hat{W}$ have to be sampled from at least one of the sets $\tilde{W}_i$, it follows that:

$$G = \bigcup_i G_i. \tag{2.66}$$

If we sample the M columns with replacement, $P_f(G_i) = (1 - \epsilon_i)^M$, and $P_f(G_i) \leq (1 - \epsilon_i)^M$ if we sample the M columns without replacement. Thus, it follows that

$$P_f(G) \leq \sum_{i=1}^{l} P_f(G_i) = \sum_{i=1}^{l} (1 - \epsilon_i)^M \leq l(1 - \bar{\epsilon})^M. \tag{2.67}$$

Hence, it follows that $P_f(\rho(\hat{W}) < l) \leq l(1-\bar{\epsilon})^M$. *If we require this probability to be less than some given* $\beta > 0$, *then, it can be shown by taking log on both sides of the above expression that* M *should satisfy*

$$M > \frac{1}{\bar{\epsilon}} log(\frac{l}{\beta}). \tag{2.68}$$

Noting that $\hat{W}$ *is rank deficient unless* $M \geq l$, *the result follows.*

Remark 2 *Effect of* $l, \bar{\epsilon}$ *on the bound M: It can be seen that the number of choices* M *is influenced primarily by* $\bar{\epsilon}$ *and not significantly by the number of active modes/ rank of the ensemble* l, *since* l *appears in the bound under the logarithm. Thus, the difficulty of choosing a sub-ensemble that is rank* l *is essentially decided by the fraction* $\bar{\epsilon_i}$ *of the ensemble in which the rarest vector* v_i *is present. Moreover, note that as the number* l *increases, we need only sample* $O(l)$ *columns to have a rank "*l*" sub-ensemble.*

Remark 3 *Effect of Sampling non-uniformly: In certain instances, for instance, when we have a priori knowledge, we may choose to sample the columns of* W *non-uniformly. Define*

$$\epsilon_i^{\Pi} = \sum_{j=1}^{N} 1_i(W_j)\pi_j, \tag{2.69}$$

where π_j *is the probability of sampling column* W_j *from the ensemble* W, *and* $1_i(W_j)$ *represents the indicator function for vector* v_i *in column* W_j, *i.e, it is one if* v_i *is present in* W_j *and 0 otherwise. Note that* ϵ_i *as defined before is the above quantity with the uniform sampling distribution* $\pi_j = \frac{1}{N}$ *for all* j. *It is reasonably straightforward to show that Proposition 1 holds with* $\bar{\epsilon}^{\Pi} = \min_i \bar{\epsilon_i}^{\Pi}$ *for any sampling distribution* Π *(we replace* ϵ_i *in (2.67) with* $\bar{\epsilon}_i^{\Pi}$*). The effect of a good sampling distribution is to lower*

the bound M by raising the number $\bar{\epsilon}^{\Pi}$ over that of a uniform distribution. This may be an intelligent option when otherwise the bound on M with uniform sampling can be very high, for instance when one of the vectors v_i is present in only a very small fraction of the ensemble W. However, we might have some a priori information regarding the columns where v_i may be present and thus, bias the sampling towards that sub-ensemble.

Next, it can be seen how the RPOD procedure extends the above result to the Balanced POD scenario where we consider the Hankel matrix. Denote $X^{(i)}$ as the set of M_1-blocks that the right eigenvector v_i is present, where M_1-block is defined in (2.60), and $\epsilon_{X,i} = \frac{no.(X^{(i)})}{p}$ as the fraction of the M_1-blocks in X which the right eigenvector v_i is present. Similarly, $\epsilon_{Y,i}$ is the fraction of the M_2-blocks in Y which the left eigenvector u_i is present. Define:

$$\bar{\epsilon}_X = \min_i \epsilon_{X,i}, \bar{\epsilon}_Y = \min_j \epsilon_{Y,j}, \tag{2.70}$$

where $\epsilon_{X,i}$ is the fraction of columns in X in which the right eigenvector v_i is present, and $\epsilon_{Y,j}$ is the fraction of the columns in Y in which the left eigenvector u_j is present.

Note that due to Lemma 1, given any $\beta > 0$, if we choose $\hat{p}m$ and $\hat{q}n$ satisfy the bounds:

$$\hat{p} > max(l, \frac{1}{\bar{\epsilon}_X} log(\frac{l}{\beta})), \hat{q} > max(l, \frac{1}{\bar{\epsilon}_Y} log(\frac{l}{\beta})), \tag{2.71}$$

then the probability of $\hat{H}$ having rank less than l is less than $\gamma = 1 - (1 - \beta)^2$, since then the probability that the ranks of the sampled input and output ensembles are less than l, is less than β. Thus, if we repeatedly choose K such ensembles with replacement, the probability of having a sub-Hankel matrix $\hat{H}$ that is still less than

rank l after the K picks, has to be less than γ^K. Thus, the probability of choosing a rank l sub-Hankel matrix $\hat{H}$ exponentially approaches unity with the number of trials. Noting that the value of β does not have a significant influence on the bounds above, it follows that β can be chosen to be quite small without significantly affecting the number of columns that need to be chosen to satisfy the confidence level of β, and thus, the probability of choosing a rank l sub-Hankel matrix can be made arbitrarily high by judiciously choosing the number of columns in the input/output ensembles according to the bounds in (2.71).

We summarize the development above as follows.

Theorem 4 *Let Hankel matrix $H = Y'X \in \Re^{qM_2 \times pM_1}$ with p inputs, q outputs, M_1, M_2 time snapshots in every input and output trajectory respectively. Let the left/ right eigenvectors $U_S = \{u_1, \cdots u_l\}$, and $V_S = \{v_1, \cdots v_l\}$ denote the eigenvectors spanning the input and output ensembles X and Y respectively. Let $\bar{\epsilon}_X, \bar{\epsilon}_Y$ be as defined in (2.70) and $\beta > 0$ be given. Suppose we construct a sub-Hankel matrix $\hat{H}$ according to the RPOD procedure: by uniformly sampling $\hat{p}$ inputs and $\hat{q}$ outputs respectively, and that $\hat{p}$ and $\hat{q}$ are chosen as in (2.71), then the probability that the sub-Hankel matrix has rank less than l is less than $\gamma = 1 - (1 - \beta)^2$. Moreover, the probability that after K RPOD choices, with replacement, the probability that the sub-Hankel matrix is less than rank l is less than γ^K.*

The following corollary immediately follows due to the developments in section 2.4.

Corollary 2 *Let $(\Lambda_{co}, U_{co}, V_{co})$ be the eigenvalues, left and right eigenvectors underlying the data in the full Hankel matrix. Given any $\beta > 0$, and that a sub-Hankel matrix $\hat{H}$ is chosen as in Proposition 4, let $(\hat{A}_r, \hat{B}_r, \hat{C}_r)$ be the ROM constructed from the sub-Hankel matrix $\hat{H}$ using RPOD algorithm. Then $\hat{A}_r = \Lambda_{co}, \hat{B}_r = U'_{co}B, \hat{C}_r = CV_{co}$*

with probability at least $(1-\beta)^2$*, and hence, with probability* $(1-\beta)^2$*, the information contained in* H *and* $\hat{H}$ *is identical in terms of the* $(\Lambda_{co}, U_{co}, V_{co})$ *triple.*

Remark 4 *Several remarks are made below about the above results.*

1. *The fractions* $\bar{\epsilon}_X$ *and* $\bar{\epsilon}_Y$ *are metrics of the "difficulty" of the problem. For instance, if all the relevant modes were controllable/observable from every input/output, then these fractions are unity, and any RPOD choice would have rank* l*. The lower these fractions are, the higher the number of rows and columns* $\hat{q}m_2$ *and* $\hat{p}m_1$ *need to be chosen such that Proposition 4 holds for the sampled sub-Hankel matrix. This corresponds to a mode, or set of modes, being controllable/ observable only from a very sparse set of actuator/ sensor locations respectively.*

2. *We do not know* $\bar{\epsilon}_X, \bar{\epsilon}_Y$ *a priori, and thus, we cannot directly apply Proposition 4. In practice, we repeatedly sample sub-Hankel matrices, and check the underlying eigenmodes from each choice. If the underlying modes from different choices are identical, then we can give a guarantee that the Hankel matrix is actually rank* l*, given a difficulty level* $\bar{\epsilon}$*. Thus, we are able to quantify the confidence in our ROMs for different values of the difficulty level* $\bar{\epsilon}$*. Typically, we have seen that if the number of rows/ columns sampled are large enough, we are able to extract all the relevant modes.*

3. *We can also vary the size of the sampled sub-Hankel matrices which in turn raises the probability of sampling a random choice with rank equal to that of the full Hankel matrix.*

4. *If we have a priori knowledge of the system, we can sample the sub-Hankel matrix using some sampling distribution other than the uniform distribution*

function, which as mentioned previously, has the effect of raising the fractions $\bar{\epsilon}_X, \bar{\epsilon}_Y$, and thus, lower the required size of the sub-Hankel matrix.

5. *In reality, the Hankel matrix is not exactly rank l but approximately rank l. In such a case, we can show that the errors incurred due to this fact is small if the contribution from the modes other than the dominant l modes are small.*

2.6 Computational Results

In this section, we compare the RPOD algorithm with BPOD algorithm for two examples: a 2D pollutant transport problem and a 2D linearized channel flow problem.

First, we define the output relative error:

$$E_{output} = \frac{\|Y_{true} - Y_{rom}\|}{\|Y_{true}\|}, \tag{2.72}$$

where Y_{true} are the outputs of the full order system, and Y_{rom} are the outputs of the reduced order system.

2.6.1 Pollutant Transport Problem

The two-dimensional advection-diffusion equation describing the contaminant transport is:

$$\frac{\partial c(x,y,t)}{\partial t} = D_x \frac{\partial^2 c(x,t)}{\partial x^2} + D_y \frac{\partial^2 c(x,t)}{\partial y^2} - v_x \frac{\partial c(x,t)}{\partial x} - v_y \frac{\partial c(x,t)}{\partial y} + S_s, \tag{2.73}$$

where c is concentration of the contaminant, D_x, D_y are the dispersions and take value 0.6 here. v_x, v_y are the velocities in x and y direction respectively, and take value 1. S_s denotes the source of pollutant, and we assume that there are three fixed sources. Also, we assume that there are three obstacles in the field. The locations

of the sources and obstacles are plotted in Fig. 2.1. The field is a square with the length of each edge $5m$.

The PDE is discretized using Finite Difference Method. We discretize the x direction into 50 equi-spaced points, and y direction into 50 equi-spaced points. Hence, the size of discretized system is 2500×2500. The initial condition for the simulation is zero, and we use Neumann boundary conditions. The model is simulated for a period of 10 minutes, and the numerical solution of the actual field at time $t = 10min$ is shown in Fig.2.1.

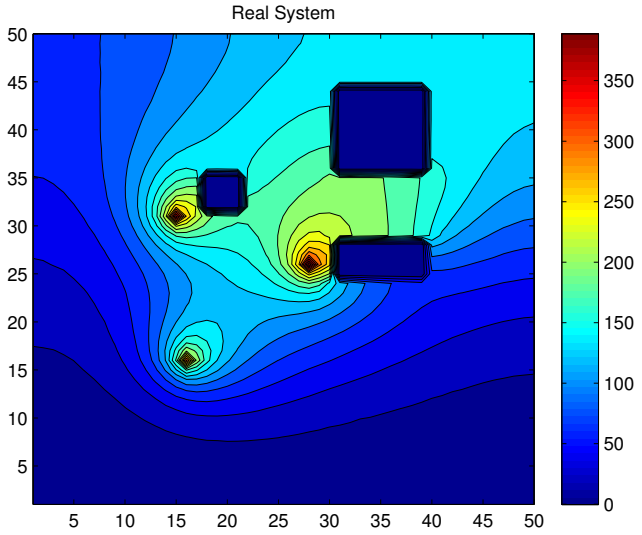
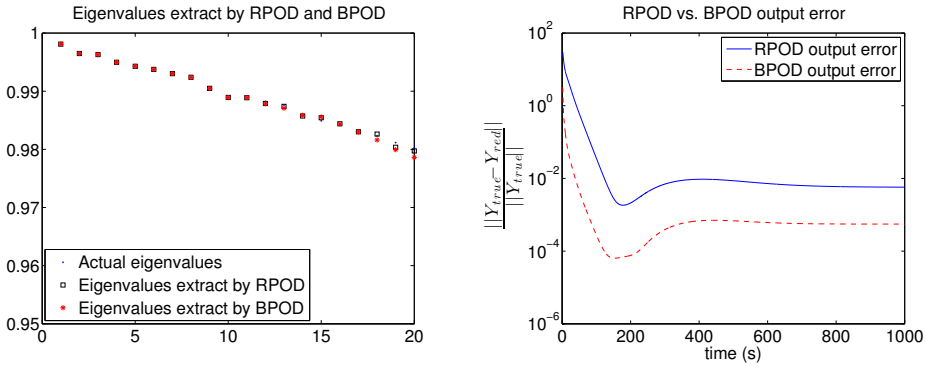


Figure 2.1: Contour plot of 2D contaminant concentration at time $t = 10min$, numerical solution using full order system

We construct ROMs using BPOD algorithm and RPOD algorithm as follows. We take full field measurements, i.e., the number of outputs is 2500. For both algorithms, we perturb the primal and adjoint system using impulse. For both algorithm, we take 500 snapshots from $t \in [0min, 10min]$ with 3 input trajectories. For the adjoint

simulation, taking 500 snapshots using BPOD algorithm is not computationally feasible, since there are 2500 output trajectories. Notice that taking snapshots earlier will allow us to extract more modes before they die out, and thus, 3 snapshots are collected from $t \in [0min, 1min]$ for each output trajectories. Therefore, for BPOD, we need to solve a 7500×1500 SVD problem. For RPOD, we randomly choose 500 measurements, and take 3 snapshots from $t \in [0min, 1min]$ for the adjoint simulation. Thus, we only need to solve a 1500×1500 SVD problem for RPOD. We extract 80 modes using both methods, and construct the ROM using these modes. In Fig.2.2(a), we show the comparison of the first twenty eigenvalues extracted by two methods.



(a) (b)

Figure 2.2: Comparison of ROM constructed using RPOD and BPOD for 2D pollutant transport problem. (a) Comparison of extracted eigenvalues with the actual eigenvalues of the full order system. (b) Comparison of the output/state relative error over time, each plot is the average performance of 3 trials.

To test the ROM, we take the perturb the system with white noise, and take average output/state error over the 3 different simulations. The state errors and the output errors are the same because we take the full state measurements, thus, we

show the comparison of the output errors in Fig. 2.2(b). We can see that BPOD is more accurate than RPOD, but both the errors are less than 1%, however, there is significant computational savings in using the RPOD over the BPOD in solving the SVD problem.

2.6.2 Linearized Channel Flow Problem

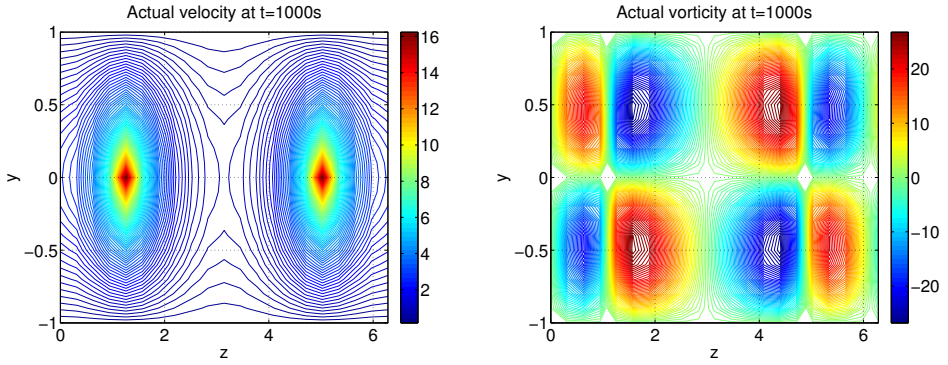
Consider the problem of the fluid flow in a plane channel. We focus on the linearized case when there are small perturbations about a steady laminar flow. The flow is perturbed by body force $B(y,z)f(t)$, which means the force is acting in the wall-normal direction. There is no-slip boundary condition at the walls $y = \pm 1$ and the flow is assumed to be periodic in the x and z direction. Assume there is no variations in the x direction, then the linearized equation of the wall-normal velocity v and the wall-normal vorticity η are given by:

$$\begin{aligned} \frac{\partial v}{\partial t} &= \frac{1}{R}\nabla^2 v + Bf, \\ \frac{\partial \eta}{\partial t} &= \frac{1}{R}\nabla^2 \eta - U'\frac{\partial v}{\partial z}, \end{aligned} \tag{2.74}$$

where $R = 100$ is the Reynolds number and $U(y) = 1 - y^2$ is the steady state velocity. The domain $z \in [0, 2\pi]$.

We discretize the system using the finite difference method, where both the y direction and z direction are discretized into 21 nodes. Thus, the size of the system is 882×882. There are 2 constant body forces on $y = 0$, and the measurements are taken on all the nodes on boundaries. The actual velocity and vorticity at $t = 1000s$ are shown in Fig. 2.3.

For BPOD, we use 80 measurements on the boundaries, and take 1000 snapshots from $t \in [0, 1000s]$ for the primal simulation, 50 snapshots from $t \in [0, 500s]$ for



(a) (b)

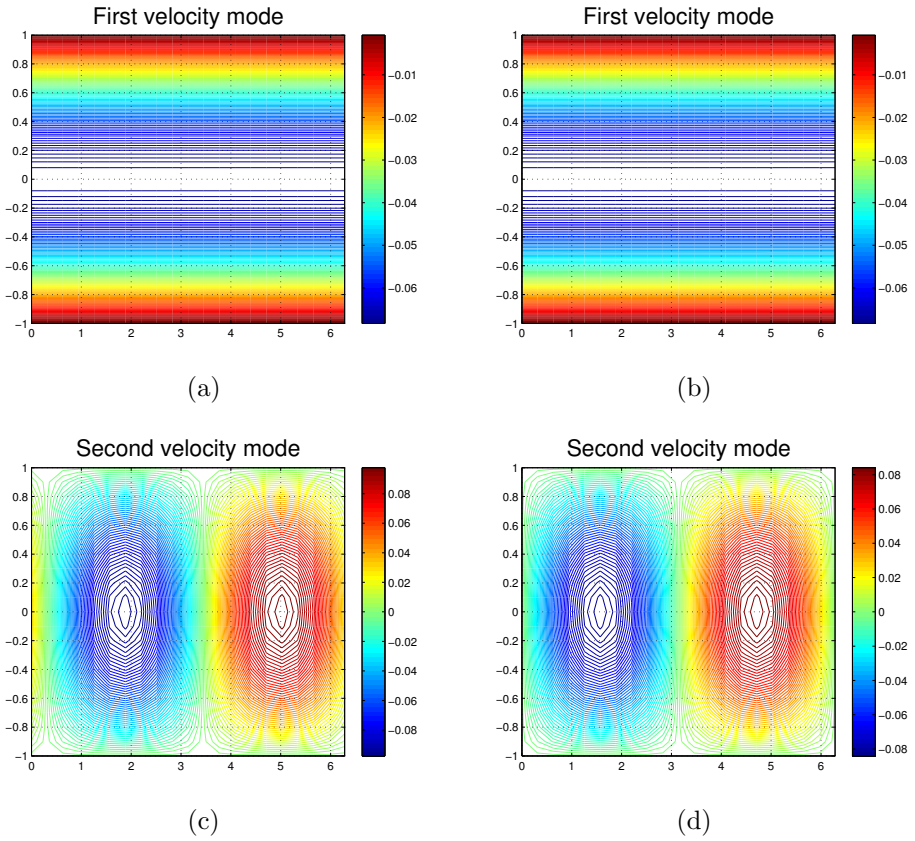
Figure 2.3: Contour plot of 2D linearized channel flow at $t = 1000s$. (a) Actual wall-normal velocity field. (b) Actual wall-normal vorticity field.

the adjoint simulation, which leads to a 8000×2000 SVD problem. For RPOD, we randomly choose 50 measurements from the 80 measurements on the boundaries, take 200 snapshots from $t \in [0, 200s]$ for the primal simulation, and take 20 snapshots from $t \in [0, 200s]$ for the adjoint simulation. Thus, we need to solve a 2000×400 SVD problem for RPOD.

In Fig. 2.4, we compare the velocity modes of the system using RPOD with the actual velocity modes. The comparison of the vorticity modes are shown in Fig. 2.5.

Here, we should note that the sign and the modulus of the ROM velocity modes are not the same as the actual modes, however, if needed, we can rescale the ROM modes to make them match. For both methods, we extract 40 modes, the first 30 extracted eigenvalues are compared in Fig. 2.6.

The comparison of the state errors and output errors are shown in Fig. 2.7. To test the ROM, we use 20 different white noise forcings and take the average output/state error over these 20 simulation. We can see that the eigenvalues extracted by RPOD and BPOD are almost the same. In this simulation, we notice that at first,



(a) (b)

(c) (d)

Figure 2.4: Comparison between ROM wall-normal velocity modes and actual velocity modes. (a) Actual first velocity mode. (b) ROM first velocity mode. (c) Actual second velocity mode. (d) ROM second velocity mode.

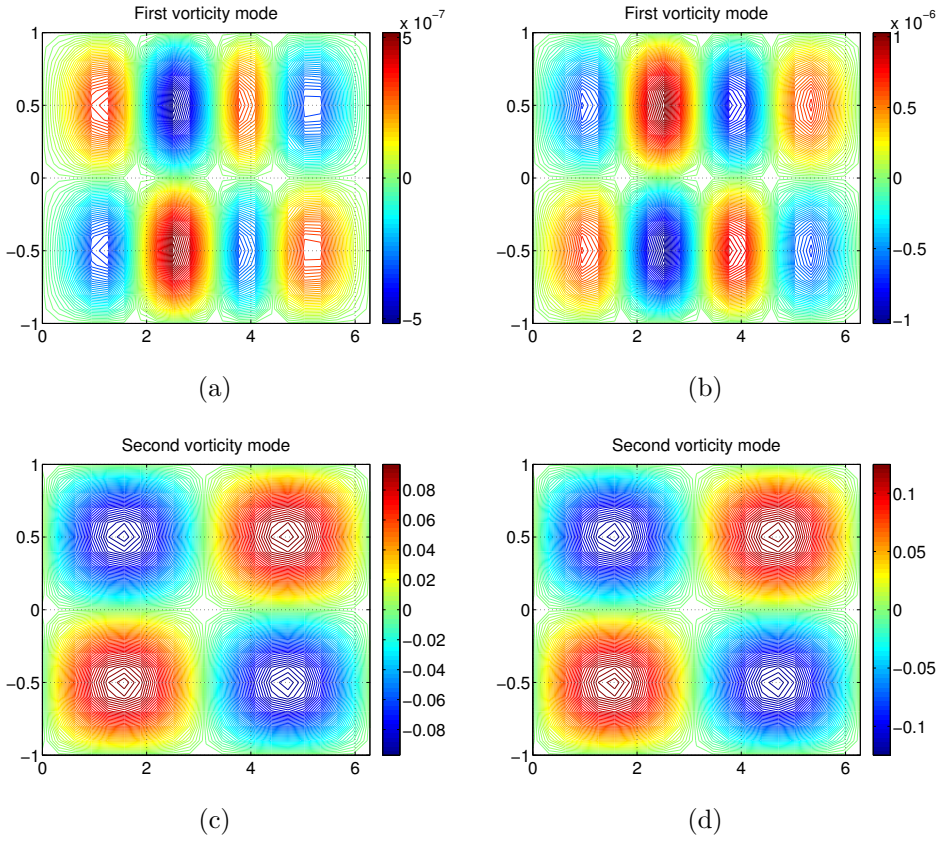


Figure 2.5: Comparison between ROM wall-normal vorticity modes and actual vorticity modes. (a) Actual first vorticity mode. (b) ROM first vorticity mode. (c) Actual second vorticity mode. (d) ROM second vorticity mode.

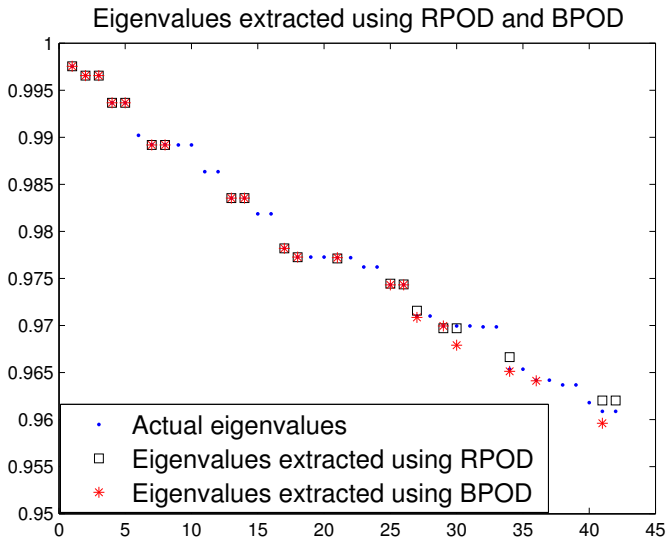


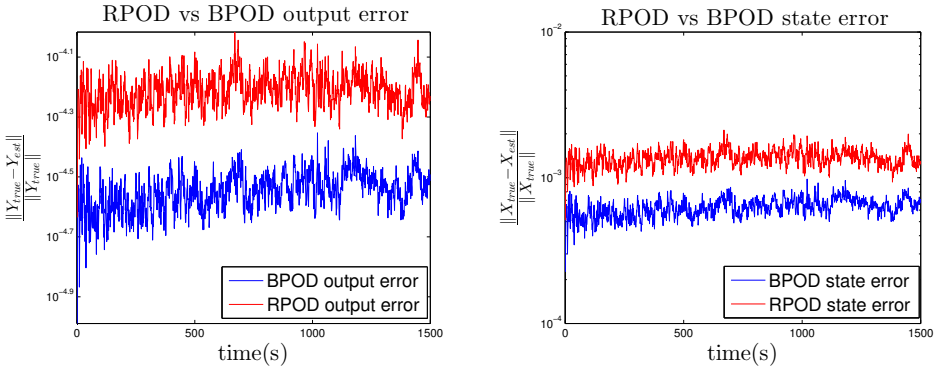
Figure 2.6: Comparison of eigenvalues extract by RPOD and BPOD for linearized channel flow problem

the state error and output error using BPOD are slightly better than using RPOD, but after some time, the errors are almost the same. The output errors using both methods are less than 0.1%, and the state errors using both methods are around 5%. Thus, we can conclude that RPOD is comparable to BPOD but requires far less computation.

2.6.3 Discussion

We compare the computational requirements/accuracy of the ROMs resulting from the BPOD and RPOD for the pollutant transport problem and linearized channel flow problem in Table 2.1.

We can see that RPOD solves a much smaller SVD problem than the BPOD, and although the errors incurred using RPOD are more than the BPOD, they are small enough not to make a major difference to the results. Thus, using the RPOD to generate a ROM is much more efficient while not sacrificing too much accuracy.



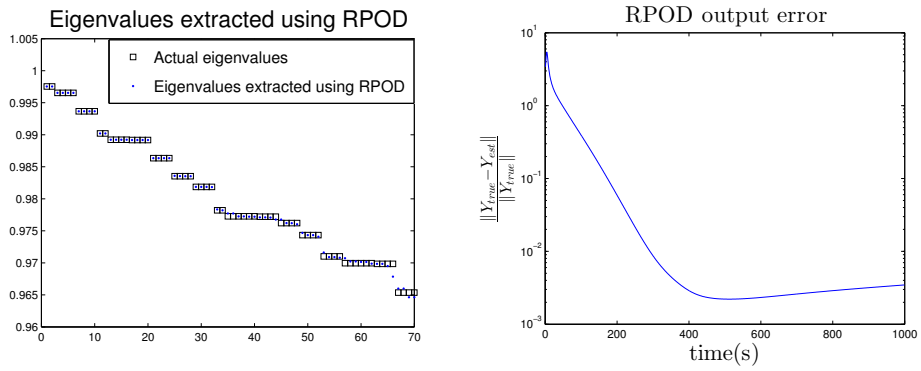
(a) (b)

Figure 2.7: Comparison of ROM errors between RPOD and BPOD for linearized channel flow problem. (a) Comparison of the output relative error over time. (b) Comparison of the state relative error over time. Each plot is the average performance of 20 trials.

Table 2.1: Comparison of performance (BPOD *V.S.* RPOD)

	Hankel matrix size	average output error
Pollutant Transport	$(7500 \times 1500) : (1500 \times 1500)$	$0.055\% : 0.6\%$
Linearized Channel Flow	$(8000 \times 2000) : (2000 \times 400)$	$0.13\% : 0.16\%$

Moreover, sometimes, it may be impossible to solve the SVD problem resulting from BPOD. For example, in the channel flow problem, if we use the full state measurements (882 measurements) and we take 20 snapshots for the adjoint simulation, there are 80 sources on the boundary and we take 1000 snapshots for the primal simulation, then we need to solve a 17640×80000 SVD problem for BPOD, which is not solvable in Matlab. For RPOD, we randomly choose 50 sources on the boundaries, and randomly choose 400 measurements. If we take 100 snapshots for the primal simulation, and 20 snapshots for the adjoint simulation, then it leads to a 8000×5000 SVD problem, which is a relatively small problem. We compare the first 70 extracted eigenvalues with the actual eigenvalues and the output errors in Figure 2.8.



(a) (b)

Figure 2.8: Simulation results using RPOD for linearized channel flow problem when BPOD is not feasible. (a) Eigenvalues extracted using RPOD. (b) Output relative errors using RPOD.

Thus, in problems where there are a large numbers of actuators/sensors, the savings can be very significant. In terms of an experiment, this observation may have added implications as it implies that we can reduce the scale of the instrumentation

required to get the data required to form an ROM by orders of magnitude without losing much information that can be extracted from the resulting data, which can result in significant cost savings.

2.7 Summary

In this section, we propose a randomized proper orthogonal decomposition technique for the extraction of ROMs for large scale systems such as those governed by PDEs. The RPOD algorithm constructs a sub-Hankel matrix of the full order Hankel matrix constructed by the BPOD algorithm without sacrificing too much accuracy. This leads to an orders of magnitude reduction in the computation required for constructing ROMs for large scale systems with a large number of inputs/outputs over the BPOD procedure.

3. COMPUTATIONALLY OPTIMAL RANDOMIZED PROPER ORTHOGONAL DECOMPOSITION (RPOD*)

3.1 Introduction

In this section, we consider the same model reduction problem as in Section 2, and we propose a computational optimal randomized proper orthogonal decomposition (RPOD*) algorithm for constructing ROMs of large scale systems. In Section 2, we had introduced the RPOD algorithm that randomly chooses a subset of the input/output trajectories. A sub-Hankel matrix is constructed using these sampled trajectories, which is then used to form a ROM in the usual BPOD fashion. The Markov parameters of the ROM constructed using the sub-Hankel matrix were shown to be close to the Markov parameters of the full order system with high probability. We proved that a lower bound exists for the number of the input/output trajectories that need to be sampled. The major problem associated with this algorithm is that different choices of the sampling algorithms would lead to different lower bounds, and the choice of a good sampling algorithm other than the uniform distribution is unclear.

In this section, we propose the RPOD* algorithm which can construct the reduced order model by perturbing the primal and adjoint system using Gaussian white noise. We show that the computations required by the RPOD* algorithm are orders of magnitude cheaper when compared to the BPOD algorithm and BPOD output projection algorithm, while the performance of the RPOD* algorithm is much better than BPOD output projection algorithm. The algorithm is optimal in the sense that a minimal number of snapshots are needed.

Both the RPOD and RPOD* algorithm aim to reduce the computational cost of

constructing the full Hankel matrix and solving the SVD problem of the resulting Hankel matrix. The differences between RPOD and RPOD* algorithm are that the information preserved in the sub-Hankel matrix constructed using RPOD is an approximation of the information preserved in the full Hankel matrix, while the exact information is preserved in the Hankel matrix constructed using RPOD*. Also, RPOD* algorithm takes advantage of the white noise perturbation, and constructs a much smaller Hankel matrix than RPOD, and hence, the computational cost is further reduced. RPOD* can also be related to the random projection algorithm.

The contribution of RPOD* algorithm is that we answer the question of how many snapshots are needed to construct a good ROM. The minimum number of snapshots needed for the snapshot based POD algorithm is defined, and is used to construct the RPOD* ROM, such that the construction of ROM can be implemented in real-time even for large scale systems. Moreover, we provide the detailed instructions on when to take snapshots and how to select ROM size, which are important in POD, and have not been discussed yet.

The organization of this section is shown as follows. In Section 3.2, we define the computationally optimal snapshot ensemble, and prove that the snapshot ensembles generated by perturbing the primal/adjoint system with white noise are computationally optimal snapshot ensembles. Then we propose the RPOD* algorithm in Section 3.3 with the formal proofs and discuss some implementation issues of the algorithm in Section 3.4. In Section 3.5, we compare the RPOD* algorithm with the related model reduction algorithm: random projection algorithm BPOD output projection algorithm, and RPOD algorithm. The computational complexity of RPOD* algorithm is shown in Section 3.6, with a comparison with BPOD output algorithm. In Section 3.7, we provide the simulation results comparing the RPOD* with the BPOD/BPOD output projection for several advection-diffusion equations.

3.2 Computationally Optimal Snapshot Ensemble

Recall that from the simplified analysis in Section 2, under Assumption 1, 2, and 3, the modal ROM consists of the controllable and observable modes, and is invariant to the snapshot ensembles, i.e., as long as the snapshot ensembles can be written as:

$$\underbrace{X_{opt}}_{N\times m} = \underbrace{V_{co}}_{N\times l}\underbrace{\alpha_{opt}}_{l\times m} + V_{c\bar{o}}\bar{\alpha}_{opt}, \underbrace{Z_{opt}}_{N\times n} = \underbrace{U_{co}}_{N\times l}\underbrace{\beta_{opt}}_{l\times n} + U_{\bar{c}o}\bar{\beta}_{opt}, \tag{3.1}$$

where $(\Lambda_{co}, V_{co}, U_{co})$ are the controllable and observable modes, and $\alpha_{opt}, \bar{\alpha}_{opt}, \beta_{opt}, \bar{\beta}_{opt}$ are coefficient matrices. Here, m, n are the number of the primal and adjoint snapshots respectively, l is the number of controllable and observable modes, and $l \ll m, n$. If $\alpha_{opt}, \beta_{opt}$ has full rank l, then as a result of Corollary 1, the modal ROM can be written as:

$$A_{opt} = \underbrace{\Lambda_{co}}_{l\times l}, B_{opt} = U_{co}'B, C_{opt} = CV_{co}. \tag{3.2}$$

Therefore, we can see that the minimum number of snapshots required is l, and under Assumption 1, 2 and 3 in Section 2, we define a computationally optimal snapshot ensemble of the system as follows.

Definition 3 *A computationally optimal snapshot ensemble X_{opt}/Z_{opt} is an l-snapshot ensemble of rank l which can be written as (3.1), i.e., $m = l, n = l$.*

The problem now is reduced to how to construct such a snapshot ensemble, and we are motivated from the random projection algorithm.

Consider the system (2.14)-(2.15), under Assumption 1 that A is stable, there exists a finite number t_{ss}, such that $\|A^{t_{ss}}\| \approx 0$. Under Assumption 1, 2 and 3, the BPOD primal snapshot ensemble which collects the state response from time $t = 0$

to $t = t_{ss}$ can be written as:

$$\underbrace{X_b}_{N\times pt_{ss}} = \begin{pmatrix} B & AB & \cdots & A^{t_{ss}-1}B \end{pmatrix} = \begin{pmatrix} V_{co} & V_{c\bar{o}} \end{pmatrix} \begin{pmatrix} \underbrace{\alpha^b_{co}}_{l\times pt_{ss}} \\ \alpha^b_{c\bar{o}} \end{pmatrix} \tag{3.3}$$

If we project X_b onto a Gaussian test matrix $\Omega \in \Re^{pt_{ss}\times l}$, where the Gaussian test matrix is a matrix whose elements are generated from a Gaussian distribution, then

$$\underbrace{X_p}_{N\times l} = \underbrace{X_b\Omega}_{N\times l} = \begin{pmatrix} V_{co} & V_{c\bar{o}} \end{pmatrix} \begin{pmatrix} \underbrace{\alpha^b_{co}\Omega}_{l\times l} \\ \alpha^b_{c\bar{o}}\Omega \end{pmatrix}, \tag{3.4}$$

where from random projection algorithm, rank $(\alpha^b_{co}\Omega) = l$. Notice that $\alpha^b_{c\bar{o}}\Omega$ does not need to be full rank, since $V_{c\bar{o}}$ would be cancelled out when constructing the Hankel matrix.

Therefore, we see that a computationally optimal snapshot ensemble can be constructed by projecting the BPOD snapshot ensemble onto a Gaussian test matrix. Furthermore, the coefficient matrix $\alpha^b_{co}\Omega$ is a Gaussian matrix, and hence, can be viewed as white noise inputs.

Thus, under Assumption 1, 2 and 3, we have the following result.

Theorem 5 *Perturb the primal system (2.14) with white noise u_k, and collect m snapshots at time $t_1, t_2, \cdots, t_m$, where $t_m \geq t_{ss}$, and $\|A^{t_{ss}}\| \approx 0$. Denote the snapshot ensemble as $X_r = \begin{pmatrix} x_1 & x_2 & \cdots & x_m \end{pmatrix}$. If $m = l$, where l is the number of the controllable and observable modes of the system, then X_r is a computationally optimal snapshot ensemble.*

Proof 4 *For the snapshots taken before t_{ss}, the state snapshot x_k at time k is:*

$$x_k = \sum_{i=0}^{k-1} A^i B u(k-i), k \leq t_{ss}. \tag{3.5}$$

Suppose there is a snapshot ensemble X_f which takes t_{ss} snapshots at time $k = 1$ to $k = t_{ss}$, then from (3.5), the snapshot ensemble X_f can be written as:

$$\begin{aligned} X_f &= \begin{pmatrix} x_1 & x_2 & \cdots & x_{t_{ss}} \end{pmatrix} \\ &= \underbrace{\begin{pmatrix} B & AB & \cdots & A^{t_{ss}-1}B \end{pmatrix}}_{X_b} \underbrace{\begin{pmatrix} u(1) & u(2) & \cdots & u(t_{ss}-1) & u(t_{ss}) \\ 0 & u(1) & \cdots & u(t_{ss}-2) & u(t_{ss}-1) \\ 0 & 0 & \cdots & u(t_{ss}-3) & u(t_{ss}-2) \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 & u(1) \end{pmatrix}}_{\Omega}, \end{aligned} \tag{3.6}$$

where X_b is the BPOD snapshot ensemble from time $k = 0$ to $k = t_{ss} - 1$. Denote $\Omega = \begin{pmatrix} \omega_1 & \omega_2 \cdots & \omega_{t_{ss}} \end{pmatrix}$, where ω_i is the i^{th} column of Ω. The Ω matrix above has columns that are linearly independent since it is upper triangular.

Since X_r consists of m columns of X_f, X_r can be written as:

$$X_r = \begin{pmatrix} x_1 & \cdots & x_m \end{pmatrix} = X_b \underbrace{\begin{pmatrix} \omega_1 & \cdots & \omega_m \end{pmatrix}}_{\Omega_1}, \tag{3.7}$$

where $\begin{pmatrix} \omega_1 & \cdots & \omega_m \end{pmatrix}$ are the corresponding columns of Ω, and hence, Ω_1 has full column rank.

For the snapshot x_k taken after t_{ss}, x_k could also be written as: $x_k = X_b \omega_k$, where ω_k is a column vector whose entries are white noises. Therefore, ω_k is independent

of $\omega_1, \cdots, \omega_m$ in (3.7), and hence, for all the snapshots collected in X_r, $X_r = X_b\Omega_1$, where Ω_1 has full column rank.

Recall that from (2.43), X_b can be written as:

$$X_b = \underbrace{V_{co}}_{N\times l}\underbrace{\alpha^b_{co}}_{l\times pt_{ss}} + \underbrace{V_{c\bar{o}}\alpha^b_{c\bar{o}}}_{N\times pt_{ss}}, \tag{3.8}$$

and α^b_{co} has full row rank. Thus, when $m = l$,

$$X_r = X_b\Omega_1 = \underbrace{V_{co}}_{N\times l}\underbrace{\alpha^b_{co}}_{l\times pt_{ss}}\underbrace{\Omega_1}_{pt_{ss}\times l} + V_{c\bar{o}}\alpha^b_{c\bar{o}}\Omega_1 = \underbrace{V_{co}}_{N\times l}\underbrace{\alpha_{co}}_{l\times l} + V_{c\bar{o}}\alpha_{c\bar{o}}, \tag{3.9}$$

where rank $\alpha_{co} = l$. Hence, X_r is a computationally optimal snapshot ensemble.

Similarly, we take $n = l$ snapshots by perturbing the adjoint system (2.15) with white noise v_k, and the adjoint snapshot ensemble can be written as:

$$Z_r = \underbrace{U_{co}}_{N\times l}\underbrace{\beta_{co}}_{l\times l} + U_{\bar{c}o}\beta_{\bar{c}o}, \tag{3.10}$$

where rank $\beta_{co} = l$, and $\beta_{\bar{c}o}$ is a constant matrix. Thus, Z_r is a computationally optimal snapshot ensemble. In Section 3.4, we discuss about how to choose m, n and the snapshots in practice.

3.3 RPOD* Algorithm

In this section, we introduce the RPOD* algorithm, and provide a formal proof of the error bound. In the simplified analysis, we assume that $U'_{\bar{c}o}B = 0, U'_{\bar{c}\bar{o}}B = 0, CV_{c\bar{o}} = 0, CV_{\bar{c}\bar{o}} = 0$, which are not true in practice, and in the formal proof, we relax this assumption.

The RPOD* algorithm constructs ROM by perturbing the primal and adjoint

system with Gaussian white noise, and the algorithm is summarized in Algorithm 4.

Error Analysis In the simplified analysis, the modal ROM only consists of the controllable and observable modes, which does not hold in practice. Under Assumption 1, 2 and 4, we have the following results.

Theorem 6 *Denote (A_r, B_r, C_r) as the ROM constructed using RPOD* following Algorithm 4. If we keep the first l non-zero singular values in (3.13), then*

$$\|A_r - \Lambda_{co}\| \propto O(\epsilon), \|B_r - U'_{co}B\| \propto O(\epsilon), \|C_r - CV_{co}\| \propto O(\epsilon), \tag{3.18}$$

and the Markov parameters $\|C_r A_r^i B_r - CA^i B\| \propto O(\epsilon), i = 1, \cdots$, where ϵ is a small number defined in Assumption 4.

The proof of Theorem 6 uses perturbation theory [86, 87] to extend the proof of the idealized Theorem 3 such that Assumption 4 holds instead of Assumption 3.

Proof 5 *Under Assumption 4, the actual snapshot ensembles can be written as:*

$$X_r = \begin{pmatrix} V_{co} & V_{c\bar{o}} & V_{\bar{c}o} & V_{\bar{c}\bar{o}} \end{pmatrix} \begin{pmatrix} \alpha_{co} \\ \alpha_{c\bar{o}} \\ \delta\alpha_{\bar{c}o} \\ \delta\alpha_{\bar{c}\bar{o}} \end{pmatrix}, \tag{3.19}$$

where $\delta\alpha_{\bar{c}o} = \epsilon\alpha_{\bar{c}o}$, $\delta\alpha_{\bar{c}\bar{o}} = \epsilon\alpha_{\bar{c}\bar{o}}$ and ϵ is defined in Assumption 4, and is a small number . Therefore, $\|V_{\bar{c}o}\delta\alpha_{\bar{c}o} + V_{\bar{c}\bar{o}}\delta\alpha_{\bar{c}\bar{o}}\| = O(\epsilon)$ are small perturbations of the ideal snapshot ensemble, and may expect the ideal result to be perturbed by a small amount as well.

Algorithm 4 RPOD* Algorithm

1. Perturb the primal system (2.14) with white noise u_k, collect m snapshots at time step $t_1, t_2, \cdots, t_m$, where $t_m \geq t_{ss}$, $\|A^{t_{ss}}\| \approx 0$, $m \geq l$. Denote the snapshot ensemble X_r as:

$$X_r = \begin{pmatrix} x_1 & x_2 & \cdots & x_m \end{pmatrix}. \tag{3.11}$$

2. Perturb the adjoint system (2.15) with white noise v_k, collect n snapshots at time step $\hat{t}_1, \hat{t}_2, \cdots, \hat{t}_n$, where $\hat{t}_n \geq t_{ss}$, $n \geq l$. Denote the adjoint snapshot ensemble Z_r as:

$$Z_r = \begin{pmatrix} z_1 & z_2 & \cdots & z_n \end{pmatrix}. \tag{3.12}$$

3. Solve the SVD problem:

$$Z_r' X_r = \begin{pmatrix} L_r & L_o \end{pmatrix} \begin{pmatrix} \Sigma_r & 0 \\ 0 & \Sigma_o \end{pmatrix} \begin{pmatrix} R_r' \\ R_o' \end{pmatrix}, \tag{3.13}$$

   and truncate at σ_l, where l is the number of controllable and observable modes present in the snapshot ensembles. Σ_r contains the first l non-zero singular values $\sigma_1 \geq \sigma_2 \geq \cdots, \geq \sigma_l > 0$, (R_r, L_r) are the corresponding right and left singular vectors.

4. Construct the POD bases:

$$T_r = X_r R_r \Sigma_r^{-1/2}, S_r = \Sigma_r^{-1/2} L_r' Z_r'. \tag{3.14}$$

5. Construct the ROM $\tilde{A}$, find the eigenvalues Λ_r and eigenvectors P_r of $\tilde{A}$.

$$\tilde{A} = S_r A T_r = P_r \Lambda_r P_r^{-1}, \tag{3.15}$$

6. Construct modal RPOD* bases:

$$\Phi_r = P_r^{-1} S_r, \Psi_r = T_r P_r. \tag{3.16}$$

7. The modal ROM is:

$$A_r = \Phi_r A \Psi_r, B_r = \Phi_r B, C_r = C \Psi_r \tag{3.17}$$

The adjoint snapshot ensemble Z_r can be written as:

$$Z_r = U_{co}\beta_{co} + U_{c\bar{o}}\delta\beta_{c\bar{o}} + U_{\bar{c}o}\beta_{\bar{c}o} + U_{\bar{c}\bar{o}}\delta\beta_{\bar{c}\bar{o}}, \tag{3.20}$$

where $\delta\beta_{c\bar{o}} = \epsilon\beta_{c\bar{o}}$, $\delta\beta_{\bar{c}\bar{o}} = \epsilon\beta_{\bar{c}\bar{o}}$, and $\beta_{c\bar{o}}, \beta_{\bar{c}\bar{o}}$ are coefficient matrices.

From (3.19) and (3.20), the Hankel matrix

$$\begin{aligned} H_r = Z_r'X_r &= \beta_{co}'\alpha_{co} + \delta\beta_{c\bar{o}}'\alpha_{c\bar{o}} + \beta_{\bar{c}o}'\delta\alpha_{\bar{c}o} + \delta\beta_{\bar{c}\bar{o}}'\delta\alpha_{\bar{c}\bar{o}} \\ &= \beta_{co}'\alpha_{co} + \epsilon\underbrace{(\beta_{c\bar{o}}'\alpha_{c\bar{o}} + \beta_{\bar{c}o}'\alpha_{\bar{c}o})}_{E_1} + O(\epsilon^2), \\ &= \beta_{co}'\alpha_{co} + \epsilon E_1 + O(\epsilon^2). \end{aligned} \tag{3.21}$$

And similarly,

$$\begin{aligned} Z_r'AX_r &= \beta_{co}'\Lambda_{co}\alpha_{co} + \epsilon\underbrace{(\beta_{c\bar{o}}'\Lambda_{c\bar{o}}\alpha_{c\bar{o}} + \beta_{\bar{c}o}'\Lambda_{\bar{c}o}\alpha_{\bar{c}o})}_{E_2} + O(\epsilon^2) \\ &= \beta_{co}'\Lambda_{co}\alpha_{co} + \epsilon E_2 + O(\epsilon^2). \end{aligned} \tag{3.22}$$

Here, E_1, E_2 are some matrices.

Under Assumption 2, there are exactly "l" controllable and observable modes, and hence, rank $(Z_r'X_r) \geq l$ due to the small perturbations.

Denote

$$\begin{aligned} \bar{H}_r &= \beta_{co}'\alpha_{co} = \bar{L}_r\bar{\Sigma}_r\bar{R}_r' + \bar{L}_o\bar{\Sigma}_o\bar{R}_o', \\ H_r &= Z_r'X_r = \beta_{co}'\alpha_{co} + \epsilon E_1 = L_r\Sigma_r R_r' + L_o\Sigma_o R_o, \end{aligned} \tag{3.23}$$

where $\bar{H}_r, H_r \in \Re^{n\times m}$, and WLOG, $n \leq m$.

Here, $\bar{H}_r$ is the ideal Hankel matrix constructed with the simplifying assumption (Assumption 3 is satisfied), and it can be seen that the true Hankel matrix H_r (Assumption 4 is satisfied) can be viewed as adding a small perturbation of $\bar{H}_r$, i.e., $H_r = \bar{H}_r + \epsilon E_1$. $\bar{\Sigma}_r \in \Re^{l\times l}$ contains the l non-zero singular values of $\bar{H}_r$ and $(\bar{L}_r, \bar{R}_r)$ are the corresponding left and right singular vectors. $\bar{\Sigma}_o \in \Re^{(n-l)\times(n-l)} = 0$ are the rest singular values, and $(\bar{L}_o, \bar{R}_o)$ are the corresponding left and right singular vectors. Similarly, $\Sigma_r \in \Re^{l\times l}$ contains the first l non-zeros singular values of H_r, and $\Sigma_o \in \Re^{(n-l)\times(n-l)}$ contains the rest singular values. The left and right singular vectors are partitioned in the same way.

First, we prove the following properties.

1. *The perturbed singular values and singular vectors (Σ_r, L_r, R_r) are related to the singular values and singular vectors $(\bar{\Sigma}_r, \bar{L}_r, \bar{R}_r)$ as:*

$$\begin{aligned} \Sigma_r &= \bar{\Sigma}_r + \epsilon E_3 + O(\epsilon^2), \\ L_r &= \bar{L}_r + \Delta L_r, R_r = \bar{R}_r + \Delta R_r, \end{aligned} \tag{3.24}$$

   where $E_3, \Delta L_r, \Delta R_r$ are some matrices, and $\|\Delta L_r\|, \|\Delta R_r\| \propto O(\epsilon)$.

2. *$\Sigma_r^{-1/2} = \bar{\Sigma}_r^{-1/2} + \epsilon C_r$, where C_r is a diagonal coefficient matrix.*

3. *The ROM $\tilde{A} = S_r A T_r$ is a perturbation of A_b in (2.48), i.e., $\tilde{A} = A_b + \Delta_3 + O(\epsilon^2)$, where $\|\Delta_3\| = O(\epsilon)$.*

Property 1 is proved in [86, 87] using perturbation theory, and the expression of E_3 is given following the results from [86] as follows.

For $\bar{\sigma}_i \in \bar{\Sigma}_r$ (strictly positive singular values) with multiplicity t, $\bar{L}_t, \bar{R}_t$ are the corresponding left and right singular vectors, the perturbed singular values $\sigma_i \in \Sigma_r$,

and

$$\sigma_i = \bar{\sigma}_i + \epsilon \underbrace{\frac{\lambda_i(\bar{R}_l' E_1' \bar{L}_t + \bar{L}_l' E_1 \bar{R}_t)}{2}}_{e_i} + O(\epsilon^2), i = 1, \cdots, t \tag{3.25}$$

where $\lambda_i(.)$ *denotes the* i^{th} *eigenvalue of* $(.)$*. Thus,* E_3 *is a diagonal matrix with diagonal element* $e_i, i = 1, \cdots, l$ *computed from (3.25).*

Property 2 is proved as follows. From (3.25), for $i = 1, \cdots, l$*,*

$$\frac{1}{\sqrt{\sigma_i}} = \frac{1}{\sqrt{\bar{\sigma}_i + e_i\epsilon}} = \frac{1}{\sqrt{\bar{\sigma}_r}\sqrt{1 + \frac{e_i\epsilon}{\bar{\sigma}_r}}} \leq \frac{1}{\sqrt{\bar{\sigma}_r}} \frac{1}{\sqrt{1 - \frac{e_i\epsilon}{\bar{\sigma}_i}}}, \tag{3.26}$$

where $e_i \geq 0, \epsilon \geq 0, \bar{\sigma}_r \geq 0$*, and* $\frac{\bar{\sigma}_i}{\epsilon} \gg 1$ *such that* $\frac{e_i\epsilon}{\bar{\sigma}_i} \ll 1$*. Furthermore,*

$$(\frac{1}{1 - \frac{e_i\epsilon}{\bar{\sigma}_i}})/(1 + \frac{e_i\epsilon}{\bar{\sigma}_i})^2 = \frac{1}{(1 - (\frac{e_i\epsilon}{\bar{\sigma}_i}))^2(1 + \frac{e_i\epsilon}{\bar{\sigma}_i})} \leq 1. \tag{3.27}$$

Therefore,

$$\frac{1}{\sqrt{\sigma_i}} = \frac{1}{\sqrt{\bar{\sigma}_i + e_i\epsilon}} \leq \frac{1}{\sqrt{\bar{\sigma}_r}} \frac{1}{\sqrt{1 - \frac{e_i\epsilon}{\bar{\sigma}_i}}} \leq \frac{1}{\sqrt{\bar{\sigma}_r}}(1 + \frac{e_i\epsilon}{\bar{\sigma}_i}). \tag{3.28}$$

Thus, $|\sigma_i^{-1/2} - \bar{\sigma_i}^{-1/2}| \leq \frac{e_i\epsilon}{\bar{\sigma}_i^{3/2}}$*, and could be written in the matrix form as*

$$\Sigma_r^{-1/2} = \bar{\Sigma}_r^{-1/2} + \epsilon C_r, \tag{3.29}$$

where C_r *is a diagonal coefficient matrix.*

Property 3 is proved as follows. Recall that the POD bases T_r, S_r *are:*

$$T_r = X_r R_r \Sigma_r^{-1/2}, S_r = \Sigma_r^{-1/2} L_r' Z_r'. \tag{3.30}$$

And the ROM $\tilde{A}$:

$$\tilde{A} = S_r A T_r = \Sigma_r^{-1/2} L_r' Z_r' A X_r R_r \Sigma_r^{-1/2} = \Sigma_r^{-1/2} L_r' (\beta_{co}' \Lambda_{co} \alpha_{co} + \epsilon E_2) R_r \Sigma_r^{-1/2}. \quad (3.31)$$

As a result of Property 1 and 2, we can prove that

$$\Sigma_r^{-1/2} L_r' = (\bar{\Sigma}_r^{-1/2} + \epsilon C_r)(\bar{L}_r' + \Delta L_r') = \bar{\Sigma}_r^{-1/2} \bar{L}_r' + \Delta_1, \quad (3.32)$$

where Δ_1 is some matrix, and $\|\Delta_1\|_2 = k_1\epsilon$, k_1 is a constant. Similarly,

$$R_r \Sigma_r^{-1/2} = \bar{R}_r \bar{\Sigma}_r^{-1/2} + \Delta_2, \quad (3.33)$$

where Δ_2 is some matrix, and $\|\Delta_2\|_2 = k_2\epsilon$, k_2 is a constant. Thus,

$$\tilde{A} = \underbrace{\bar{\Sigma}_r^{-1/2} \bar{L}_r' \beta_{co}' \Lambda_{co} \alpha_{co} \bar{R}_r \bar{\Sigma}_r^{-1/2}}_{A_b} + \Delta_3 + O(\epsilon^2), \quad (3.34)$$

where Δ_3 is some matrix, and $\|\Delta_3\|_2 = k_3\epsilon$, k_3 is a constant. Therefore,

$$\tilde{A} = A_b + \Delta_3 + O(\epsilon^2). \quad (3.35)$$

If we let

$$A_b = \underbrace{\bar{\Sigma}_r^{-1/2} \bar{L}_r' \beta_{co}'}_{\bar{P}_r} \Lambda_{co} \underbrace{\alpha_{co} \bar{R}_r \bar{\Sigma}_r^{-1/2}}_{\bar{P}_r^{-1}}, \quad (3.36)$$

from the proof in Section 2.4 ((2.48) - (2.50)), Λ_{co} are the eigenvalues of A_b, and

$\bar{P}_r$ *are the corresponding eigenvectors. Then from perturbation theory [88],*

$$\tilde{A} = A_b + \Delta_3 + O(\epsilon^2) = P_r \Lambda_r P_r^{-1}, \tag{3.37}$$

where $\|P_r - \bar{P}_r\| \leq \|\Delta_3\| = k_3\epsilon, \|\Lambda_r - \Lambda_{co}\| \leq \|\Delta_3\| = k_3\epsilon$.

Thus, RPOD bases can be written as*

$$\begin{aligned} \Psi_r = T_r P_r &= X_r \bar{R}_r \bar{\Sigma}_r^{-1/2} \bar{P}_r + \Delta_4, \\ &= V_{co} + V_{c\bar{o}} \alpha_p + \Delta_5, \end{aligned} \tag{3.38}$$

where Δ_4, Δ_5 *are some matrices, and* $\|\Delta_4\|, \|\Delta_5\| \propto O(\epsilon)$, α_p *is a coefficient matrix.*

Similarly, we can prove that

$$\Phi_r = P_r^{-1} S_r = U'_{co} + \beta_p U'_{\bar{c}o} + \Delta_6, \tag{3.39}$$

where $\|\Delta_6\| \propto O(\epsilon)$ *is some matrice and* β_p *is a coefficient matrix. Substitute (3.38), (3.39) into the modal ROM, we have:*

$$\begin{aligned} A_r &= \Lambda_r = \Lambda_{co} + \Delta_7, \\ B_r &= \Phi_r B = U'_{co} B + \Delta_8, \\ C_r &= C\Psi_r = CV_{co} + \Delta_9, \end{aligned} \tag{3.40}$$

where $U'_{\bar{c}o} B = \epsilon\Delta_{10}, CV_{c\bar{o}} = \epsilon\Delta_{11}$, *and* $\|\Delta_7\|, \|\Delta_8\|, \|\Delta_9\| \propto O(\epsilon)$, *and the Markov parameters are:*

$$C_r A_r^i B_r = CV_{co} \Lambda_{co}^i U'_{co} B + \Delta, \tag{3.41}$$

where $\|\Delta\| \propto O(\epsilon)$.

In Assumption 4, ϵ is assumed to be a small number, and ϵ can also be related to σ_{l+1} as follows.

Corollary 3

$$\|C_r A_r^i B_r - CA^i B\| \propto O(\epsilon) \propto O(\sigma_{l+1}). \tag{3.42}$$

Proof 6 *For* $\bar{\sigma}_i \in \bar{\Sigma}_o$ *(zero singular values) [86] with multiplicity* $n-l$*, the corresponding left and right singular vectors are* $\bar{L}_o, \bar{R}_o$*. The perturbed singular values* $\sigma_i \in \Sigma_o$ *and*

$$\sigma_i = \epsilon\sqrt{\lambda_i(\bar{R}_o' E_1' \bar{L}_o \bar{L}_o' E_1 \bar{R}_o)}, i = 1, \cdots n-l. \tag{3.43}$$

From (3.25) and (3.43), we can see that:

$$\sigma_l = \bar{\sigma}_l + e_l\epsilon + O(\epsilon^2), \sigma_{l+1} = e_{l+1}\epsilon, \tag{3.44}$$

Hence, we have:

$$\sigma_{l+1} \propto O(\epsilon), \tag{3.45}$$

and

$$\|C_r A_r^i B_r - CA^i B\| \propto O(\epsilon) \propto O(\sigma_{l+1}). \tag{3.46}$$

3.4 Implementation Issues

In this section, we discuss some implementation problems in the RPOD* algorithm. We give the insight into how to collect the snapshot ensembles, and how to select the ROM size.

Snapshot selection From the analysis in Section 3.2, we only need to collect $m = l$ snapshots from the primal simulations. However, l is not known a priori, thus, in practice, we start with a random guess $m << N$, where N is the dimension of the system, or we can choose m from experience. For instance, in a fluid system with 10^6 degrees of freedom, m is $O(10) \sim O(10^2)$. Similarly, we guess n, and then we check the rank of $Z_r' X_r$. From the analysis in the proof, we see that rank $(Z_r' X_r) \geq l$ since there are small perturbations, and thus, if $Z_r' X_r$ has full rank, i.e., rank $(Z_r' X_r) = \min(m, n)$, then it is possible that we did not take enough snapshots, and hence, we increase m, n, until rank $(Z_r' X_r) < \min(m, n)$.

Now we consider the question to when to take the snapshots. In the RPOD* algorithm, we only require that if we take m snapshots from time $t_1, \cdots, t_m$, then $t_m \geq t_{ss}$, where $\|A^{t_{ss}}\| \approx 0$. Now, for simplicity, we assume that the snapshots are taken at $\Delta T, 2\Delta T \cdots, m\Delta T$, where, ΔT is a small constant. In Fig. 3.1, we show one simulation result comparing the accuracy of the ROMs using $\Delta T = 3, 5, 10, 20, 50$ for the atmospheric dispersion problem introduced in Section 3.7.2. Here $t_{ss} = 900$, and $m = 300$. The output relative error is defined in (2.72), and we take the average of the output relative error for each ROM. It can be seen that as ΔT increases, each column in Ω_1 is well separated, and hence, the ROM is more accurate, while it takes longer time to generate the snapshots. Thus, this is a trade-off between the accuracy and the computational efficiency.

ROM size selection In Theorem 6, we assume that there are l controllable and

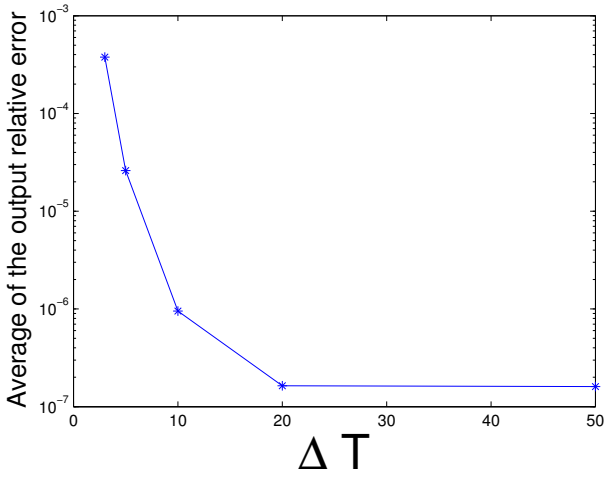


Figure 3.1: This figure explains when to take the snapshots in RPOD*. The snapshots are taken at every ΔT time, and the averaged output relative error is plotted as a function of ΔT.

observable modes, and we keep exact l non-zero singular values. However, l is not known as a priori. In the following, first, we give a proof regarding the ROM errors when $k \geq l$.

Proof 7 *Consider the SVD problem of the Hankel matrix:*

$$H_r = Z_r' X_r = L\Sigma R, \tag{3.47}$$

where $\Sigma = diag\{\sigma_1, \cdots, \sigma_n\}$ *is a diagonal matrix, and* $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$ *are the singular values,* (R, L) *are the right and left singular vectors. If we keep the first* k *singular values, and* $k \geq l$*, then denote*

$$\Sigma_p = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{pmatrix} = \begin{pmatrix} \Sigma_r & \\ & \Sigma_n \end{pmatrix}, \tag{3.48}$$

where $\Sigma_r \in \Re^{l\times l}, \Sigma_n \in \Re^{(k-l)\times(k-l)}$. *The corresponding left and right singular vectors can be partitioned in the same way, and are denoted as* $L_p = \begin{pmatrix} L_r & L_n \end{pmatrix}, R_p = \begin{pmatrix} R_r & R_n \end{pmatrix}$.

Then the transformation matrices are partitioned as:

$$S = \begin{pmatrix} S_1 \\ S_2 \end{pmatrix} = \begin{pmatrix} \Sigma_r^{-1/2} L_r' Z_r' \\ \Sigma_n^{-1/2} L_n' Z_r' \end{pmatrix}, \tag{3.49}$$

$$T = \begin{pmatrix} T_1 & T_2 \end{pmatrix} = \begin{pmatrix} X_r R_r \Sigma_r^{-1/2} & X_r R_n \Sigma_n^{-1/2} \end{pmatrix}, \tag{3.50}$$

and thus, the ROM is:

$$\tilde{A} = SAT = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} \Sigma_r^{-1/2} L_r' Z_r' A X_r R_r \Sigma_r^{-1/2} & \Sigma_r^{-1/2} L_r' Z_r' A X_r R_n \Sigma_n^{-1/2} \\ \Sigma_n^{-1/2} L_n' Z_r' A X_r R_r \Sigma_r^{-1/2} & \Sigma_n^{-1/2} L_n' Z_r' A X_r R_n \Sigma_n^{-1/2} \end{pmatrix} \tag{3.51}$$

Here, A_{11} *is exactly the ROM if we keep* $k = l$ *non-zero singular values, and as we proved in Theorem 6,* $A_{11} = P_r \Lambda_r P_r^{-1}$, *where* $\|\Lambda_r - \Lambda_{co}\| \propto O(\epsilon)$.

Consider $A_{12} \in \Re^{l\times(k-l)}$ *for example,*

$$\begin{aligned} A_{12} &= \Sigma_r^{-1/2} L_r' Z_r' A X_r R_n \Sigma_n^{-1/2}, \\ \|A_{12}\| &\le \|\Sigma_r^{-1/2} L_r' Z_r' A X_r R_n\| \|\Sigma_n^{-1/2}\|, \end{aligned} \tag{3.52}$$

and $\|\Sigma_n^{-1/2}\| = \sigma_k^{-1/2}$, *thus, it can be proved that*

$$\|A_{12}\| \le k_4 \sigma_k^{-1/2}, \|A_{21}\| \le k_5 \sigma_k^{-1/2}, \|A_{22}\| \le k_6 \sigma_k^{-1}. \tag{3.53}$$

where k_4, k_5, k_6 are some constants. Therefore,

$$\tilde{A} = \begin{pmatrix} A_b & 0 \\ 0 & 0 \end{pmatrix} + \underbrace{\begin{pmatrix} \Delta_3 & A_{12} \\ A_{21} & A_{22} \end{pmatrix}}_{E} \tag{3.54}$$

where

$$\begin{aligned} \|E\|^2 \leq \|E\|_F^2 &= \|\Delta_3\|_F^2 + \|A_{12}\|_F^2 + \|A_{21}\|_F^2 + \|A_{22}\|_F^2 \\ &\leq l\|\Delta_3\|^2 + (k-l)\|A_{12}\|^2 + (k-l)\|A_{21}\|^2 + (k-l)\|A_{22}\|^2, \end{aligned} \tag{3.55}$$

and suppose $(k-l) \leq l$. Hence,

$$\|E\| \leq k_3\sqrt{l}\epsilon + k_7\sqrt{k-l}\sigma_k^{-1/2} + k_6\sqrt{k-l}\sigma_k^{-1}, \tag{3.56}$$

where k_7 is a constant.

The eigenvalues of $\begin{pmatrix} \bar{A}_{11} & 0 \\ 0 & 0 \end{pmatrix}$ *are* $\bar{\Lambda}_{ij} = \begin{pmatrix} \Lambda_{co} & \\ & 0 \end{pmatrix}$*, and the corresponding eigenvectors are* $\bar{P}_{ij} = \begin{pmatrix} \bar{P} & 0 \\ 0 & I \end{pmatrix}$*, thus, the perturbed eigenvalues and eigenvectors of $\tilde{A}$ are:*

$$\tilde{A} = P_{ij}\Lambda_{ij}P_{ij}^{-1}, \tag{3.57}$$

where $P_{ij} = \bar{P}_{ij} + \Delta_{12}, \Lambda_{ij} = \bar{\Lambda}_{ij} + \Delta_{13}$, and Δ_{12}, Δ_{13} are some matrices, $\|\Delta_{12}\|, \|\Delta_{13}\| \propto$

$O(\|E\|)$. *Thus,*

$$\Psi = TP_{ij} = \begin{pmatrix} X_r \bar{R}_r \bar{\Sigma}_r^{-1/2} \bar{P} & 0 \end{pmatrix} + \Delta_{14},$$
$$\Phi = P_{ij}^{-1} S = \begin{pmatrix} \bar{P}^{-1} \bar{\Sigma}_r^{-1/2} \bar{L}_r' Z_r' \\ 0 \end{pmatrix} + \Delta_{15}, \tag{3.58}$$

and Δ_{14}, Δ_{15} *are some matrices, and* $\|\Delta_{14}\|, \|\Delta_{15}\| \propto O(\|E\|)$. *Following the same proof in Section 2.4, the ROM Markov parameters:*

$$C_r A_r^i B_r = C\Psi\Lambda^i\Phi B = CV_{co}\Lambda_{co}^i U_{co}' B' + \Delta_{16}, \tag{3.59}$$

where Δ_{16} *is a matrix, and* $\|\Delta_{16}\| \propto O(\|E\|)$. *Thus, if we truncate at* σ_k, *and* $\sigma_k = O(\epsilon)$, *then* $\|E\| \propto O(\sigma_{k+1}^{-1})$, *and* $\|C_r A_r^i B_r - CA^i B\| \propto O(\sigma_k^{-1})$. *Therefore, the errors between the Markov parameters of the true system and the ROM system increase as* k *increases, when* $k \geq l$.

As proved above, we see that if we keep k non-zero singular values, and $k > l$, then undesired noise will be introduced. If $k < l$, not all the controllable and observable modes can be recovered. Therefore, in practice, we decide l by trial and error. Since $\text{rank}\,(Z_r' X_r) \geq l$ is always true, we start with $k = \text{rank}\,(Z_r' X_r)$, and check the eigenvalues of $\tilde{A} = SAT$. From the proof above, we can see that when $k > l$, $k - l$ eigenvalues of $\tilde{A}$ are small, which are the perturbations of the zero eigenvalues. If $k >> l$, then the perturbations in the ROM is too large to be neglected, which results in unstable eigenvalues of $\tilde{A}$. Thus, we keep decreasing the value of k until $\tilde{A}$ is stable. If there are some small eigenvalues which are approximately zero, we know $k > l$, and we decrease the value of k until we reach a region $[l, l + a]$, where a is a small number, such that most of the eigenvalues of $\tilde{A}$ remain the same for

different value of k (l controllable and observable modes with $k-l$ perturbations of the zero eigenvalues), then we stop and pick the number l as the number of non-zero eigenvalues of $\tilde{A}$.

3.5 Comparison with Related Algorithms

In this section, we compare the RPOD* algorithm with the most related algorithms, which include BPOD, random projection, BPOD output projection and RPOD algorithm.

3.5.1 Comparison with BPOD

In the following, we summarize the differences between the BPOD and RPOD* algorithm.

As we mentioned in Section 2.4, $(p+q)$ simulations are needed for BPOD algorithm, where p is the number of inputs and q is the number of outputs. Let X_b, Z_b denote the impulse response snapshot ensembles that need to be collected in BPOD algorithm from time step $(1, \cdots, t_{ss})$. Thus, $X_b \in \Re^{N \times pt_{ss}}, Z_b \in \Re^{N \times qt_{ss}}$. It is expensive to store $(p+q)t_{ss}$ snapshots, and it is expensive to solve the resulting SVD problem due to the large size of the problem. For RPOD* algorithm, only (1 primal + 1 adjoint) simulations are needed, and only $m+n$, where $m, n \ll t_{ss}$ snapshots need to be stored. Also, it is easy to solve the resulting SVD problem.

Another practical problem with impulse responses snapshots is that the snapshots after some time are dominated by very few slow modes, and including these snapshots does not give much new information. On the other hand, the RPOD* trajectories are white noise forced, and all the modes are always present in all the snapshots due to the persistent excitation of the white noise terms. Hence, the RPOD* snapshots can be taken till t_{ss}, and be assured that all of the relevant modes will be captured.

3.5.2 Comparison with Random Projection

From the analysis in Section 3.2, we see that the snapshot ensembles collected in RPOD* can be written as:

$$X_r = X_b\Omega_1, Z_r = Z_b\Omega_2, \tag{3.60}$$

where $X_b \in \Re^{N\times pt_{ss}}, Z_b \in \Re^{N\times qt_{ss}}$ are the impulse response snapshot ensembles that need to be collected in the BPOD algorithm from time step $(0, \cdots, t_{ss}-1)$, and $\|A^{t_{ss}}\| \approx 0$. $\Omega_1 \in \Re^{pt_{ss}\times m}$ and $\Omega_2 \in \Re^{qt_{ss}\times n}$ are full rank Gaussian matrices. Also, we have:

$$\underbrace{H_r}_{\text{rank } l} = Z_r'X_r = \Omega_2'Z_b'X_b\Omega_1 = \Omega_2'\underbrace{H_b}_{\text{rank } l}\Omega_1, \tag{3.61}$$

Therefore, the Hankel matrix constructed in RPOD* algorithm can be viewed as projecting the full Hankel matrix onto two Gaussian matrices using random projection algorithm. While a direct application of the random projection would require to generate the Hankel matrix H_b (and X_b, Z_b) first. However, in practice, the construction and the storage of the Hankel matrix is computationally prohibitive when N is large and the number of inputs/outputs is large. Also, the bottleneck of the random projection algorithm is the projection of X_b, Z_b onto the Gaussian test matrices. In RPOD* algorithm, the snapshot ensembles are constructed directly from the primal and adjoint simulations, and hence, the computational cost to generate the Hankel matrix and to project it onto the Gaussian test matrices is saved.

3.5.3 Comparison with BPOD output projection

As we mentioned in Section 2.4, when the number of the outputs q is large, the computation of the BPOD adjoint simulations may not be tractable. To reduce the number of the outputs, the output projection method in [30, 89] is proposed. In this section, we compare the RPOD* algorithm with BPOD output projection algorithm.

When the number of the outputs is large, BPOD output projection projects the outputs onto a lower-dimensional subspace using POD algorithm first, i.e., construct POD basis $\Theta_s \in \Re^{q\times s}$, such that $\tilde{y}_k = \Theta_s' C x_k \in \Re^{s\times 1}$, where $s \ll q$, and $\|y - \Theta_s \tilde{y}\|$ is minimized. Then collect the adjoint snapshot ensemble using s output channels.

The BPOD output projection method is summarized in Algorithm 5.

In the following, we compare the BPOD output projection method with RPOD* algorithm. From (3.63), the adjoint snapshot ensemble Z_s can be written as:

$$\begin{aligned}
\underbrace{Z_s}_{N\times st_{ss}} &= \begin{pmatrix} C'\Theta_s & A'C'\Theta_s & \cdots & (A')^{t_{ss}}C'\Theta_s \end{pmatrix} \\
&= \underbrace{\begin{pmatrix} C' & A'C' & \cdots & (A')^{t_{ss}}C' \end{pmatrix}}_{Z_b} \underbrace{\begin{pmatrix} \Theta_s & & & \\ & \Theta_s & & \\ & & \cdots & \\ & & & \Theta_s \end{pmatrix}}_{\Theta} \\
&= \underbrace{Z_b}_{N\times qt_{ss}} \underbrace{\Theta}_{qt_{ss}\times st_{ss}} .
\end{aligned} \tag{3.66}$$

And hence, the projected Hankel matrix H_s is:

$$H_s = \Theta' Z_b' X_b = \Theta' H_b, \tag{3.67}$$

Algorithm 5 BPOD output projection algorithm

1. Collect the primal snapshot ensemble X_b in (5).

2. Compute the POD modes of the dataset $Y_b = CX_b$.

$$Y_b' Y_b \phi_k = \lambda_k \phi_k, \lambda_1 \geq \cdots \geq \lambda_n \geq 0, \tag{3.62}$$

   where λ_k are the eigenvalues, and ϕ_k are the corresponding eigenvectors. Thus, the POD modes $\Theta_s = [\phi_1, \cdots, \phi_s]$, where s is the rank of the output projection, and $s \ll q$.

3. Collect the impulse responses of the adjoint system:

$$z_{k+1} = A' z_k + C' \Theta_s v, w = B' z_k. \tag{3.63}$$

   The adjoint snapshot ensemble is denoted as Z_s.

4. Construct the Hankel matrix:

$$H_s = Z_s' X_b. \tag{3.64}$$

5. Solve the SVD problem of H_s, and construct the BPOD output projection bases as (T_s, S_s) using equations (8) and (9). The ROM is:

$$A_s = S_s A T_s, B_s = S_s B, C_s = C T_s. \tag{3.65}$$

Recall that the adjoint snapshot ensembles collected in RPOD* can be written as $Z_r = Z_b\Omega_2$, and the projected Hankel matrix in RPOD* is $H_r = \Omega_2' H_b \Omega_1$. Thus, we make the following remark.

Remark 5 *Both the BPOD output projection and RPOD* algorithms can be viewed as projecting the full order Hankel matrix onto a reduced order Hankel matrix with projection matrices* Θ *and* (Ω_1, Ω_2).

However, the differences between two algorithms are summarized as follows.

Differences between two algorithms. First, the information preserved in H_s, H_r are not the same. The output projection $P_s = \Theta_s\Theta_s'$ minimizes the 2-norm of the difference between the original transfer function and the output-projected transfer function, i.e. $\|CA^iB - \Theta_s\Theta_s'CA^iB\|, i = 1, \cdots$, is minimized. Thus, the controllable and observable modes preserved in H_s are an approximation of those in H_b, while in RPOD* algorithm, the exact controllable and observable modes are preserved using the Gaussian random projection matrix. As mentioned in [89], when $s < l$, where l is the number of non-zero Hankel singular values (number of controllable and observable modes), then only the first s Hankel singular values are the same as the full balanced truncations Hankel singular values.

Another difference between output projection algorithm and RPOD* algorithm is that RPOD* algorithm can be used when both the number of the inputs and outputs are large, while output projection can be used when the number of the inputs or the outputs is large. When the number of inputs is large, we can construct an input projection using the adjoint snapshot ensemble, but when both the number of inputs and outputs are large, the construction of the projection matrix is not helpful.

3.5.4 Comparison with RPOD

Now, we can compare the RPOD algorithm and RPOD* algorithm proposed in this dissertation.

Recall that for a system with p inputs and q outputs, RPOD algorithm randomly chooses $\hat{p}, \hat{q}$ inputs and outputs respectively, where the lower bound of $\hat{p}, \hat{q}$ is given. Then the snapshot ensembles are constructed by perturbing the primal and adjoint system with impulses. Consider the RPOD primal snapshot ensemble $\hat{X}$, suppose $b_1, \cdots, b_{\hat{p}}$ columns of B are selected, WLOG, and denote $\hat{B} = [b_1, \cdots, b_{\hat{p}}]$. The snapshots are taken from $k = 0$ to $k = t_{ss} - 1$.

From the analysis in section 2.5, if BPOD snapshot ensemble is spanned by $V_{co}, V_{c\bar{o}}$, i.e.,

$$\underbrace{X_b}_{N \times pt_{ss}} = \begin{pmatrix} B & AB & \cdots & A^{t_{ss}-1}B \end{pmatrix} = \begin{pmatrix} V_{co} & V_{c\bar{o}} \end{pmatrix} \begin{pmatrix} \underbrace{\alpha^b_{co}}_{l \times pt_{ss}} \\ \alpha^b_{c\bar{o}} \end{pmatrix}, \tag{3.68}$$

then with a high probability, the sub-snapshot ensemble $\hat{X}$ constructed using RPOD is also spanned by $V_{co}, V_{c\bar{o}}$, i.e.,

$$\underbrace{\hat{X}}_{N \times \hat{p}t_{ss}} = \begin{pmatrix} \hat{B} & A\hat{B} & \cdots & A^{t_{ss}-1}\hat{B} \end{pmatrix} = \begin{pmatrix} V_{co} & V_{c\bar{o}} \end{pmatrix} \begin{pmatrix} \underbrace{\hat{\alpha}^b_{co}}_{l \times \hat{p}t_{ss}} \\ \hat{\alpha}^b_{c\bar{o}} \end{pmatrix}, \tag{3.69}$$

where the RPOD* snapshot ensemble X_r could be written as:

$$\underbrace{X_r}_{N \times l} = \begin{pmatrix} V_{co} & V_{c\bar{o}} \end{pmatrix} \begin{pmatrix} \underbrace{\alpha^r_{co}}_{l \times l} \\ \alpha^r_{c\bar{o}} \end{pmatrix}. \tag{3.70}$$

Therefore, both the algorithms construct a sub-snapshot ensemble which could be spanned by V_{co} and $V_{c\bar{o}}$. The difference is that V_{co} is guaranteed to be preserved in RPOD* algorithm, while V_{co} is preserved in RPOD algorithm with a high probability, so RPOD* is more accurate than RPOD algorithm. Moreover, RPOD* algorithm takes fewer snapshots, and only need 1 simulation, and hence, the computational cost of RPOD algorithm is further reduced.

3.6 Computational Cost Analysis

The comparison of the computational cost of BPOD output projection algorithm with RPOD* algorithm is shown in Table 3.1. We collect m, n snapshots for RPOD* algorithm respectively, and t_{ss} snapshots for BPOD output projection algorithm. N is the dimension of the system, p, q are the number of inputs and outputs respectively, the rank of the output projection is s, and without loss of generality, we assume $p \leq q$, $m \leq n$, and $p \leq s$. Denote T as the time to propagate the primal/adjoint system once.

Table 3.1: Computational Complexity Analysis for RPOD* and BPOD Output Projection

	RPOD*	BPOD output projection
Generate snapshot ensembles	$2t_{ss}T$	$(p+s)t_{ss}T$
Construction of H	$O(mnN)$	$O(pst_{ss}^2N)$
Solve SVD	$O(m^2n)$	$O(p^2st_{ss}^3)$

First, we compare the computation time to generate the snapshot ensembles. For BPOD output projection algorithm, $(p + s)$ simulations are needed, and for each simulation, we need to collect the snapshots up to $t = t_{ss}$. Thus, the total computation time to generate the snapshot ensembles is $(p + s)t_{ss}T$.

The RPOD* algorithm needs 1 primal and 1 adjoint simulation till time $m\Delta T, n\Delta T$ respectively, where $m\Delta T \geq t_{ss}, n\Delta T \geq t_{ss}$. When we choose ΔT such that $m\Delta T = t_{ss}, n\Delta T = t_{ss}$, the performance of the ROM constructed using RPOD* is better than those constructed using BPOD/BPOD output projection algorithm. While, the total computation time to generate the snapshot ensembles is $2t_{ss}T$, which means that RPOD* computational cost is the same as BPOD in a single input single output (SISO) system. The comparison of the computation time to generate the snapshot ensembles is shown in Table 3.3 for two examples.

Next, we compare the computational cost to construct the Hankel matrix and solving the SVD problem. In practice, $m, n \propto O(10) \sim O(10^2)$, and $t_{ss} \propto O(10^2) \sim O(10^3)$, $p, s \propto O(10)$. Thus, for constructing the Hankel matrix, the computational time using RPOD* is $\frac{mn}{pst_{ss}^2} \propto O(10^{-4})$ time that of using BPOD output projection, and for solving the SVD problem, the computation time using RPOD* is $\frac{m^2n}{p^2st_{ss}^3} \propto O(10^{-6})$ times that of using BPOD output projection.

3.7 Computational Results

In this section, we compare the performance of ROMs constructed using RPOD* algorithm with BPOD and BPOD output projection algorithms. We start with a small one-dimensional heat problem, which all three algorithms could be utilized. Then we present the comparison between the RPOD* algorithm and BPOD output projection algorithm for a large scale three-dimensional atmospheric dispersion problem, while the BPOD algorithm is not computationally feasible for the atmospheric dispersion problem.

The frequency response for multiple input multiple output systems can be represented by plotting the maximum singular value of the transfer function matrix $\max(\sigma(H(j\omega)))$ as a function of frequency ω. We define the frequency responses

error as:

$$E_{fre}(j\omega) = |\max(\sigma(H_{true}(j\omega))) - \max(\sigma(H_{rom}(j\omega)))|, \tag{3.71}$$

where $H_{true}(j\omega)$ is the transfer function of the full order system, and $H_{rom}(j\omega)$ is the transfer function of the ROM.

3.7.1 Heat Problem

The heat transformation along a slab is described by the partial differential equation:

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} + f, \tag{3.72}$$

with boundary conditions

$$T|_{x=0} = 0, \frac{\partial T}{\partial x}|_{x=L} = 0, \tag{3.73}$$

where α is the thermal diffusivity, and f is the forcing.

The parameters of the system are summarized in Table 3.2. There are two point sources located at $x = 0.15m$ and $x = 0.45m$. The system is discretized using the finite difference method, and there are 100 grids which are equally spaced. We take full field measurements, i.e., measurements at every node. In the following, we compare the ROM constructed using RPOD*, BPOD and BPOD output projection.

For RPOD*, the system is perturbed by the white noise with distribution $N(0, I_{2\times 2})$. At time $t_{ss} = 3000s$, $\|A^{t_{ss}}\| \approx 0$, thus, for the primal/adjoint simulation, one realization is needed, and we collect 80 equispaced snapshots during time $t \in [0, 3200]s$.

For BPOD, we need $p = 2$ realizations for the primal simulation, and $q = 100$ realizations for the adjoint simulations. In general, 3000 equispaced snapshots between $[0, 3000]s$ should be taken in each realization for BPOD/output projection. However, due to the memory limits on the platform, only 400 equispaced snapshots can be taken and for the optimal performance of BPOD/output projection, the 400 equispaced snapshots are taken from time $t \in [0, 400]s$ (first 400 dominant impulse responses).

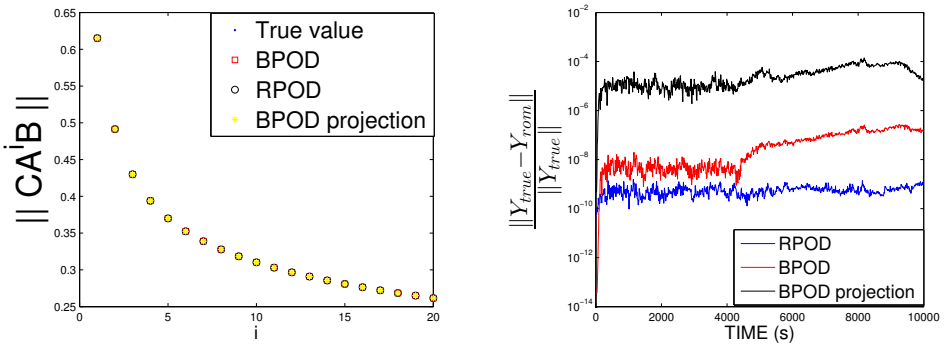
The rank of the output projection is 40, and hence, for the BPOD output projection algorithm, 40 realizations are needed for the adjoint simulations. RPOD* algorithm and BPOD algorithm extract 70 modes, and BPOD output projection algorithm extract 50 modes. Here, we take 80 snapshots by trial and error following the procedure in Section 3.4.

Table 3.2: Parameters of Heat and Atmospheric Dispersion System

	Heat	Atmospheric Dispersion
Domain	$x \in [0, 1](m)$	$x \in [0, 2000](m)$, $y \in [-100, 400](m)$, $z \in [0, 50](m)$
Dimension of system	$N = 100$	$N = 10^5$
Parameters	$\alpha = 4.2 \times 10^{-6}(m^2/s)$	Wind velocity $\vec{u} = (4m/s, 0, 0)$
Number of Inputs	2	10
Number of Outputs	100	810
Primal Snapshots	400 V.S. 80	200 V.S. 400
Adjoint Snapshots	400 V.S. 80	200 V.S. 400
Snapshots taken during	$t \in [0, 400s]$ V.S. $t \in [0s, 3200s]$	$t \in [0, 200]s$ V.S. $t \in [0, 4000]s$
ROM modes	50 V.S. 70	200 V.S. 380
Hankel matrix	16000×800 V.S. 80×80	4000×2000 V.S. 400×400

In Fig. 3.2(a), we compare the norm of the Markov parameters of the ROM

constructed using three algorithms with the true Markov parameters of the full order system. We perturb the system with random Gaussian noise, and compare the output relative errors in Fig. 3.2(b).

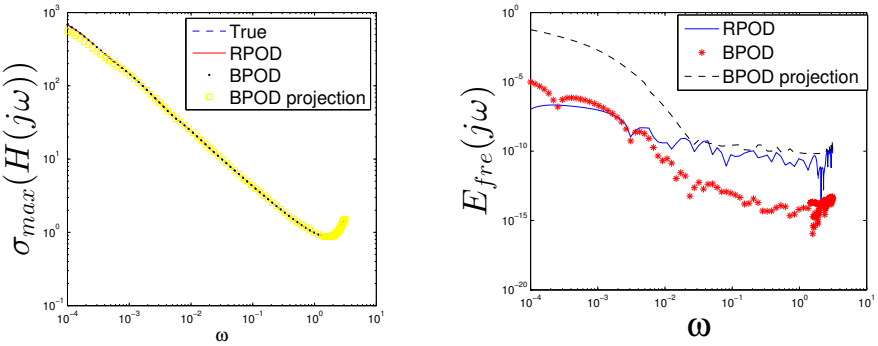


(a) (b)

Figure 3.2: Comparison of time domain errors between RPOD*, BPOD and BPOD output projection for heat transfer problem. (a) Comparison of Markov parameters. (b) Comparison of output relative errors.

In Fig. 3.3(a), we compare the frequency responses of the ROM constructed using three algorithms with the true frequency responses. In Fig. 3.3(b), we plot the error of the maximum singular value of the input-output model as a function of frequency.

For all three methods, we can see that the Markov parameters of the ROM are close to the true Markov parameters of the full order system. From Fig. 3.2(b), it can be seen that all three methods are accurate enough. The output relative error of BPOD output projection is less than 0.01%, and the performance of the RPOD* algorithm is much better than the BPOD output projection algorithm, the performance of RPOD* algorithm is better than BPOD algorithm because we do not take all the snapshots up to t_{ss} due to the memory limits. From Fig. 3.3, we can see that the frequency responses error of RPOD* algorithm is smaller than BPOD



(a) (b)

Figure 3.3: Comparison of frequency responses between RPOD*, BPOD and BPOD output projection for heat transfer problem. (a) Comparison of frequency responses. (b) Comparison of frequency response errors.

and BPOD output projection algorithm in low frequencies. With the increase of the frequency, the BPOD algorithm performs better than RPOD* algorithm, however, the errors are below 10^{-10}, and hence, the difference is negligible. The comparison of computational time of RPOD* and BPOD output projection algorithm is shown in Section 3.7.3. We can see that the construction of the snapshot ensembles using RPOD* takes almost the same time as BPOD output projection, while the dominant computational cost is solving the SVD problem, and it can be seen that RPOD* is about 24 times faster than BPOD output projection.

3.7.2 Atmospheric Dispersion Problem

The three-dimensional advection-diffusion equation describing the contaminant transport in the atmosphere is:

$$\frac{\partial c}{\partial t} + \nabla \cdot (c\vec{u}) = \nabla \cdot (K(\vec{X})\nabla c) + Q\delta(\vec{X} - \vec{X}_s), \tag{3.74}$$

where

$c(\vec{X}, t)$: mass concentration at location $\vec{X} = (x, y, z)$.

$\vec{X}_s = (x_s, y_s, z_s)$: location of the point source.

$\vec{u} = (ucos(\alpha), usin(\alpha), 0)$: wind velocity. α is the direction of the wind in the horizontal plane and the wind velocity is aligned with the positive x-axis when $\alpha = 0$, $u \geq 0$ is constant.

Q: contaminant emitted rate.

∇: gradient operator.

$K(\vec{X}) = diag(K_x(x), K_y(x), K_z(x))$: diagonal matrix whose entries are the turbulent eddy diffusivities. In general $K(\vec{X})$ is a function of the downwind distance x only.

In practice, the wind velocity is sufficiently large that the diffusion in the x-direction is much smaller than advection, and hence, assume that the term $K_x \partial_x^2 c$ can be neglected.

Define $\sigma_y^2(x) = \frac{2}{u}\int_0^x K_y(\eta)d\eta$, $\sigma_z^2(x) = \frac{2}{u}\int_0^x K_z(\eta)d\eta$, where $\sigma_y(x) = a_y x(1 + b_y x)^{0.5}$, $\sigma_z(x) = a_z x(1 + b_z x)^{0.5}$, and $a_y = 0.008, b_y = 0.00001, a_z = 0.006, b_z = 0.00015$.

The boundary conditions are:

$$\begin{aligned} &c(0, y, z) = 0, c(\infty, y, z) = 0, c(x, \pm\infty, z) = 0, \\ &c(x, y, \infty) = 0, K_z \frac{\partial c}{\partial z}(x, y, 0) = 0. \end{aligned} \tag{3.75}$$

The system is discretized using finite difference method, and there are $100 \times 100 \times 10$ grids which are equally spaced. The parameters are summarized in Table 3.2. There are 10 point sources which are shown in Fig. 3.4. We take the full field measurements (except the nodes on $x = 0, \infty$ and $y = \pm\infty$). In Fig. 3.4, we show the actual concentration of the full field at time $t = 200s$ with Q as Gaussian white

noise where sources are the dotted points in the figure.

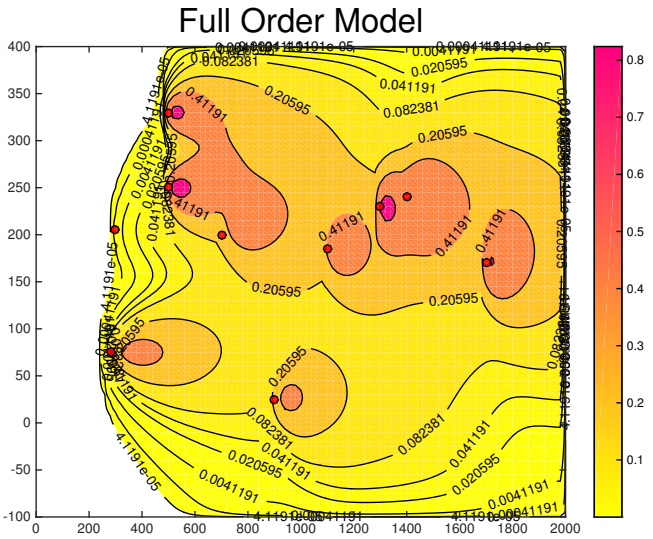


Figure 3.4: Contour plot of air pollutant concentration at $t = 200s$.

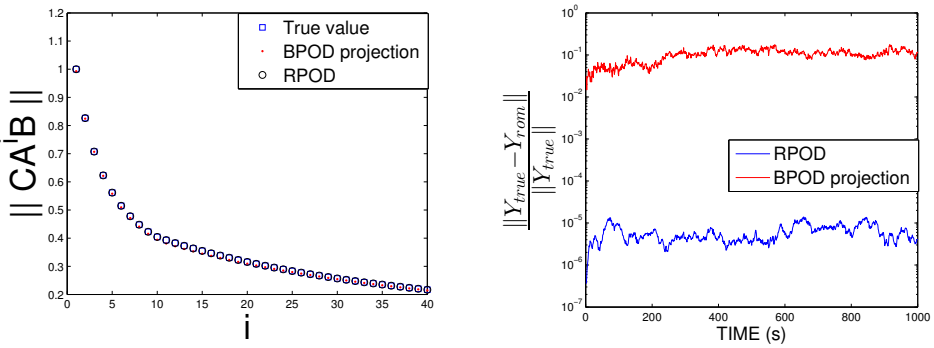
In this example, since the system dimension is $N = 10^5$, constructing the ROM with the full field measurements using BPOD is computationally impossible, and thus, we only compare the RPOD* algorithm with BPOD output projection algorithm.

For RPOD* algorithm, we collect the snapshots sequentially. The system is perturbed by white noise with distribution $N(0, I_{10\times 10})$. One primal simulation and one adjoint simulation are needed. We collect 400 equispaced snapshots from time $t \in [0, 4000]s$, where at time $t_{ss} = 4000$, $\|A^{t_{ss}}\| \approx 0$, and extract 380 modes.

For BPOD output projection algorithm, we collect the impulse responses from the primal simulations, and $p = 10$ realizations are needed. Same as the heat example, we could not collect all the impulse responses from $t \in [0, 4000]s$ due to the memory limits on the platform. Hence, for the best performance available in this example, we

collect 200 equispaced snapshots from $t \in [0, 200]s$ for each primal simulation. The rank of the output projection is 80, and hence, 80 adjoint simulations are needed, and in the adjoint simulations, we collect 50 equispaced snapshots from $t \in [0, 50]s$. We can extract 200 modes. The parameters are summarized in Table 3.2.

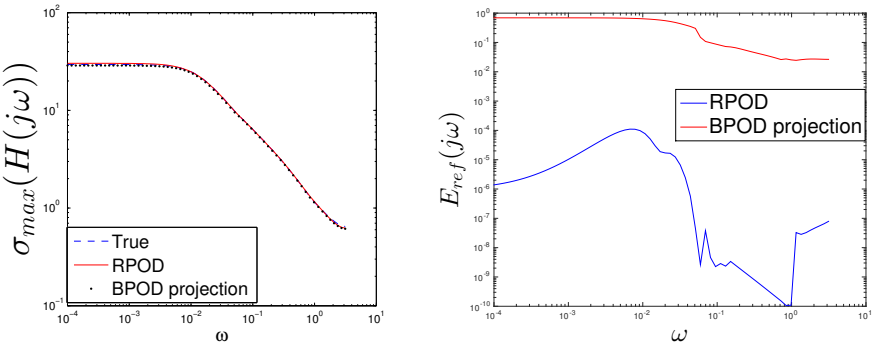
In Fig. 3.5(a), we compare the Markov parameters of the ROM constructed using RPOD* and BPOD output projection with the full order system. Also, we perturb the system with random Gaussian noise, and compare the output relative errors in Fig. 3.5(b). The comparison of the computational time is shown in Section 3.7.3.



(a) (b)

Figure 3.5: Comparison of time domain errors between RPOD* and BPOD output projection for atmospheric dispersion problem. (a) Comparison of Markov parameters. (b) Comparison of output relative errors.

In Fig. 3.6(a), we compare the frequency responses of the ROM constructed using RPOD* and BPOD output projection with the full order system. We can see that the frequency responses of the ROMs are almost the same as the frequency responses of the full order system. In Fig. 3.6(b), we show the errors between the frequency responses.



(a) (b)

Figure 3.6: Comparison of frequency responses between RPOD* and BPOD output projection for atmospheric dispersion problem. (a) Comparison of frequency responses. (b) Comparison of frequency response errors.

The comparison of the computational time is shown in Section 3.7.3. It can be seen that for this example, the construction of the snapshot ensembles using RPOD* is faster than the BPOD output projection, and the dominant computation cost is the construction of $Z'X$, where RPOD* algorithm is almost 16500 times faster than BPOD output projection.

From the examples above, we can see that for both examples showed in this paper, the RPOD* algorithm is much faster than BPOD/BPOD output projection algorithm, and is much more accurate than BPOD output projection algorithm.

3.7.3 Comparison of Computational Time

Comparison of computational time is shown in Table 3.3 for two examples. All of the experiments reported in this paper were performed using Matlab 2013b on a Dell OptiPlex 9020, Intel(R) Core (TM) i7-4770CPU, 3.40GHz, 4GB RAM machine.

Table 3.3: Comparison of Computational Time using RPOD* and BPOD output projection for Heat Transfer and Atmospheric Dispersion

	Heat		Atmospheric Dispersion	
	RPOD*	output projection	RPOD*	output projection
Generate X	0.0148s	0.0518s	55.59s	30.461s
Generate Z	0.0340s	0.0864s + 0.1582s (projection)	56.23s	421.99s + 9.287s (projection)
Construct $Z'X$	0.0292s	0.0461s	0.321s	5311s
Solve SVD	0.1052s	2.5550s	0.4859s	9.118s
Total time	0.1832s	2.8975s	112.6269s	5781.856s

3.8 Summary

In this section, we develop a computationally optimal randomized proper orthogonal decomposition algorithms for the extraction of ROMs of large scale systems such as those governed by PDEs. The ROM is constructed by perturbing the primal and adjoint system with Gaussian white noise, and the error bound using the RPOD* algorithm is derived. The RPOD* algorithm is compared with the BPOD output projection and random projection algorithm, and we show that the computational cost to construct the snapshot ensembles is saved and also RPOD* algorithm leads to a much smaller SVD problem. The simulation results show that the RPOD* algorithm is much more accurate than the BPOD output projection algorithm.

4. AN AUTOREGRESSIVE (AR) MODEL BASED STOCHASTIC UNKNOWN INPUT REALIZATION AND FILTERING TECHNIQUE *

4.1 Introduction

In this section, we consider the state estimation problem of systems with stochastic unknown inputs. We assume that the unknown inputs can be treated as a wide-sense stationary process (WSS) with rational power spectral density (PSD), while no other prior information about the unknown inputs is known. We develop an AR model based method which can recover the statistics of unknown inputs from measurements. An innovations model of the unknown inputs is constructed, and the Augmented State Kalman Filter (ASKF) is applied for state estimation. An ROM constructed via RPOD* algorithm is used to reduce the computations of ASKF.

We start this section by formulating the problem and making some general assumptions. Then we present the procedure of the AR model based unknown input realization approach, and provide formal proofs regarding the estimation errors. Next, we construct an augmented state model, and use KF for state estimation. Finally, we compare the simulation results using the AR model based algorithm with optimal two-stage Kalman filter (OTSKF) and unbiased minimum-variance (UMV) algorithms for several advection-diffusion equations. Brief reviews of eigensystem realization algorithm (ERA), OTSKF and UMV algorithms are included in Appendix B - D for completeness.

*Reprinted with permission from "A stochastic unknown input realization and filtering technique" by Dan Yu, Suman Chakravorty, 2016, Automatica, 26: 26-33, Copyright [2016] by Elsevier.

4.2 Problem Formulation

Consider a complex valued linear time-invariant (LTI) discrete time system:

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_{k-1}, \\ y_k &= Cx_k + v_k, \end{aligned} \tag{4.1}$$

where $x_k \in \mathbb{C}^n$, $y_k \in \mathbb{C}^q$, $v_k \in \mathbb{C}^q$, $u_k \in \mathbb{C}^p$ are the state vector, the measurement vector, the measurement white noise with zero mean and known covariance Ω, and the unknown stochastic inputs respectively. The process u_k is used to model the presence of the external disturbances, process noise, and unmodeled terms. Here, $A \in \mathbb{C}^{n\times n}$, $B \in \mathbb{C}^{n\times p}$, $C \in \mathbb{C}^{q\times n}$ are known.

The following assumptions are made about system (4.1):

- A1. A is a stable matrix, and (A, C) is detectable.
- A2. rank$(B) = p$, rank$(C) = q$, $p \leq q$ and rank (CAB) = rank $(B) = p$.
- A3. u_k and v_k are uncorrelated.
- A4. We further assume that the unknown input u_k can be treated as a WSS process:

$$\begin{aligned} \xi_k &= A_e\xi_{k-1} + B_e\nu_{k-1}, \\ u_k &= C_e\xi_k + \mu_k, \end{aligned} \tag{4.2}$$

  where ν_k, μ_k are uncorrelated white noise processes.

Discussion on Assumptions A2 is a weaker assumption than the so-called "observer matching" condition used in unknown input observer design. The observer

matching condition requires rank (CB) = rank $(B) = p$, which in practice, may be too restrictive. A2 implies that if there are p inputs, then there should be at least p controllable and observable modes. A4 implies that u_k is a WSS process with a rational power spectrum.

We consider the state estimation problem when the system (4.2), i.e., (A_e, B_e, C_e) are unknown. Given the output data y_k, we want to construct an innovations model for the unknown stochastic input u_k, such that the output statistics of the innovations model and system (4.2) are the same. Given such a realization of the unknown input, we apply the standard Kalman filter for the state estimation, augmented with the unknown input states.

4.3 AR Model Based Unknown Input Realization Technique

In this section, we propose an AR model based unknown input realization technique which can construct an innovations model of the unknown inputs such that the ASKF can be applied for state estimation. First, a least squares problem is formulated based on the relationship between the inputs and outputs to recover the statistics of the unknown inputs. Then an AR model is constructed using the recovered input statistics, and a balanced realization model is then constructed using the ERA.

4.3.1 Extraction of Input Autocorrelations via a Least Squares Problem

Consider system (4.1) with zero initial conditions, the output y_k can be written as:

$$y_k = \sum_{i=1}^{\infty} h_i u_{k-i} + v_k. \tag{4.3}$$

Since v_k is the measurement noise with known covariance Ω, the autocorrelation

function $R_{vv}(m)$ can be written as:

$$R_{vv}(m) = \begin{cases} \Omega, m = 0, \\ 0, \text{ otherwise .} \end{cases}$$

For a LTI system, the output $\{y_k\}$ is a WSS process when $\{u_k\}$ is WSS. Therefore, under assumption A1 that A is stable, the output autocorrelation can be written as:

$$\begin{aligned} R_{yy}(m) = E[y_k y_{k+m}^*] &= \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} h_i u_{k-i} u_{k+m-j}^* h_j^* + R_{vv}(m) \\ &= \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} h_i R_{uu}(m+i-j) h_j^* + R_{vv}(m), \end{aligned} \tag{4.4}$$

where $m = 0, \pm 1, \pm 2, \cdots$ is the time-lag between y_k and y_{k+m}. Here, assumption A3 is used.

We denote $\hat{R}_{yy}(m) = R_{yy}(m) - R_{vv}(m)$. Therefore, the relationship between input and output autocorrelation function is given by:

$$\hat{R}_{yy}(m) = \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} h_i R_{uu}(m+i-j) h_j^*. \tag{4.5}$$

To solve for the unknown input autocorrelations $R_{uu}(m)$, first we need to use a theorem from linear matrix equations [90].

Theorem 7 *Consider the matrix equation*

$$AXB = C, \tag{4.6}$$

where A, B, C, X are all matrices. If $A \in \mathbb{C}^{m\times n} = (a_1, a_2, \cdots, a_n)$, where a_i are the

columns of A, then define $vec(A) \in \mathbb{C}^{mn\times 1}$ *and the Kronecker product* $A \otimes B$ *as:*

$$vec(A) = \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix}, A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \cdots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix}. \tag{4.7}$$

If A *is an* $m \times n$ *matrix and* B *is a* $p \times q$ *matrix, then the Kronecker product* $A \otimes B$ *is an* $mp \times nq$ *block matrix.*

The matrix equation (4.6) can be transformed into one vector equation:

$$(B^T \otimes A)vec(X) = vec(C), \tag{4.8}$$

where $B^T \otimes A$ *is the Kronecker product of* B^T *and* A*.*

Thus, by applying Theorem 7, (4.5) can be written as:

$$\underbrace{\text{vec}(\hat{R}_{yy}(m))}_{\in R^{q^2\times 1}} = \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \underbrace{\bar{h}_j \otimes h_i}_{\in R^{q^2\times p^2}} \underbrace{\text{vec}(R_{uu}(m+i-j))}_{\in R^{p^2\times 1}}, \tag{4.9}$$

Now, we estimate the unknown input autocorrelations by the following procedure.

Choose design parameter M**.** Under assumption A1, i.e., the system is stable, the Markov parameters of the system (4.1) have the following property:

$$\|h_i\| \to 0 \text{ as } i \to \infty. \tag{4.10}$$

We choose a design parameter M, such that (4.9) can be written as:

$$\text{vec}(\hat{R}_{yy}(m)) = \sum_{i=1}^{M}\sum_{j=1}^{M} \bar{h}_j \otimes h_i \text{vec}(R_{uu}(m+i-j)). \tag{4.11}$$

where M varies with different systems and can be chosen as large as desired.

Choose design parameters N_o, N_i. Under assumption A1 and A4, we have the following property:

$$\begin{aligned} &\|R_{uu}(m)\| \to 0, \text{ as } m \to \infty, \\ &\|\hat{R}_{yy}(m)\| \to 0, \text{ as } m \to \infty. \end{aligned} \tag{4.12}$$

As a standard method when computing a power spectrum from an autocorrelation function, we choose design parameters N_i and N_o, such that the input autocorrelations are calculated when $|m| \leq N_i$, and the output autocorrelations are calculated when $|m| \leq N_o$. The numbers N_o and N_i depend on the dynamic system and unknown inputs, and can be chosen as large as required. We have the following result.

Lemma 2 *The relation $N_i \leq N_o$ holds, which implies that all significant input autocorrelations can be recovered from the output autocorrelations.*

Proof 8 *The support of $\hat{R}_{yy}$ is limited to $(-N_o, N_o)$, thus, we have:*

$$\hat{R}_{yy}(N_o + 1) = 0. \tag{4.13}$$

From (4.9),

$$vec(\hat{R}_{yy}(N_o + 1)) = \sum_{i=1}^{\infty} \bar{h}_i \otimes h_i vec(R_{uu}(N_o + 1)) + \sum_{i=2}^{\infty} \bar{h}_{i-1} \otimes h_i vec(R_{uu}(N_o)) + \cdots . \tag{4.14}$$

If $N_i > N_o$, which implies that

$$R_{uu}(N_o + 1) \neq 0, \tag{4.15}$$

then it follows that $R_{yy}(N_o+1)$ is also not negligible, which contradicts the assumption, and hence, as a consequence, $N_i \leq N_o$.

Thus, the following equation is used for computation of the unknown input autocorrelations.

$$\mathrm{vec}(\hat{R}_{yy}(m)) = \sum_{i=1}^{M}\sum_{j=1}^{M} \bar{h}_j \otimes h_i \mathrm{vec}\underbrace{(R_{uu}(m+i-j))}_{|m+i-j|\leq N_i}, |m| \leq N_o. \tag{4.16}$$

Solve the least squares problem We collect $2N_o+1$ output autocorrelations, and from the above assumptions, there are $2N_i+1$ unknown input autocorrelations:

$$\underbrace{\begin{pmatrix} \mathrm{vec}(\hat{R}_{yy}(-N_o)) \\ \mathrm{vec}(\hat{R}_{yy}(-N_o+1)) \\ \vdots \\ \mathrm{vec}(\hat{R}_{yy}(0)) \\ \mathrm{vec}\hat{R}_{yy}(1)) \\ \vdots \\ \mathrm{vec}(\hat{R}_{yy}(N_o)) \end{pmatrix}}_{\mathrm{vec}(\hat{R}_{yy})} = C_{yu} \underbrace{\begin{pmatrix} \mathrm{vec}(R_{uu}(-N_i)) \\ \vdots \\ \mathrm{vec}(R_{uu}(0)) \\ \mathrm{vec}(R_{uu}(1)) \\ \vdots \\ \mathrm{vec}(R_{uu}(N_i)) \end{pmatrix}}_{\mathrm{vec}(R_{uu})}, \tag{4.17}$$

where C_{yu} is the coefficient matrix and can be calculated from (4.16). Under assumption A1, A2 and A4, we have the following result.

Theorem 8 *Equation (4.17) has a unique least-squares solution $\hat{R}_{uu}(m)$, where $m = 0, \pm 1, \pm 2, \cdots, \pm N_i$.*

Proof 9 *We partition the matrix C_{yu} into three parts as $C_{yu} = \begin{pmatrix} C_t \\ C_m \\ C_b \end{pmatrix}$, where C_m*

contains the $q^2(N_o - N_i) + 1, \cdots, q^2(N_o + N_i + 1)$ *rows of* C_{yu} *and can be expressed as:*

$$C_m = \begin{pmatrix} \sum_{j=1}^{M} \bar{h}_j \otimes h_j & \sum_{j=1}^{M-1} \bar{h}_j \otimes h_{j+1} & \cdots & \cdots \\ \sum_{j=1}^{M-1} \bar{h}_{j+1} \otimes h_j & \sum_{j=1}^{M} \bar{h}_j \otimes h_j & \cdots & \cdots \\ \cdots & \cdots & \ddots & \cdots \\ \cdots & \cdots & \cdots & \sum_{j=1}^{M} \bar{h}_j \otimes h_j \end{pmatrix}. \tag{4.18}$$

In the following, we prove that $C_m \in \mathbb{C}^{q^2(2N_i+1) \times p^2(2N_i+1)}$ *has full column rank* $p^2(2N_i + 1)$ *by induction.*

Let $N_i = 0$*, then*

$$\begin{aligned} C_m(0) &= \sum_{j=1}^{M} \bar{h}_j \otimes h_j \\ &= (CV_{co} \otimes CV_{co})(I + \Lambda_{co} \otimes \Lambda_{co} + \cdots + \Lambda_{co}^{M-1} \otimes \Lambda_{co}^{M-1})(U'_{co}B \otimes U'_{co}B), \end{aligned} \tag{4.19}$$

where Λ_{co} *are the controllable and observable eigenvalues of* A*, and* (V_{co}, U_{co}) *are the corresponding right and left eigenvectors. Under the assumption A2, if rank* $(CAB) = p$*, and since* $CAB = CV_{co}\Lambda_{co}U'_{co}B$*, which implies that rank* $(C_m(0)) = p^2$*.*

If rank $C_m(N_i - 1)$ *has rank* $p^2(2N_i - 1)$*, then consider* $C_m(N_i)$*:*

$$C_m(N_i) = \begin{pmatrix} C_m(0) & C_{12} & C_{13} \\ C_{21} & C_m(N_i - 1) & C_{23} \\ C_{31} & C_{32} & C_m(0) \end{pmatrix}, \tag{4.20}$$

where $C_{12}, C_{13}, C_{21}, C_{23}, C_{31}, C_{32}$ *are some matrices, and it can be proved that* $C_m(N_i)$ *has* $p^2 + p^2(2N_i - 1) + p^2 = p^2(2N_i + 1)$ *independent columns, and hence, rank* $(C_m(N_i)) = p^2(2N_i + 1)$.

Thus, by induction, C_m *has full column rank, and hence,* C_{yu} *has full column rank. Since* $q \geq p$*, it is an overdetermined system, so there exists a unique solution to the least squares problem.*

Remark 6 *The size of* C_{yu} *is* $q^2(2N_o + 1) \times p^2(2N_i + 1)$ *and it would be large when* p *and* q *increase, and hence, large scale least squares problem needs to be solved for systems with large number of inputs/outputs. For example, a modified conjugate gradients method [91] could be used as follows.*

The least squares problem need to be solved is:

$$vec(\hat{R}_{yy}) = C_{yu} vec(R_{uu}). \tag{4.21}$$

Multiply C_{yu}^* *on both sides:*

$$C_{yu}^* vec(\hat{R}_{yy}) = C_{yu}^* C_{yu} vec(R_{uu}). \tag{4.22}$$

If we denote $L_s = C_{yu}^* vec(\hat{R}_{yy})$, $\bar{x} = vec(R_{uu})$, *and* $C_s = C_{yu}^* C_{yu}$, *then* $C_s = C_s^*$, *and the problem is equivalent to solve the least squares problem for* $\bar{x}$*:*

$$C_s \bar{x} = L_s, \tag{4.23}$$

and a conjugate gradient method to solve this problem is summarized in Algorithm 6.

The error of the input autocorrelations we extract results from two design pa-

Algorithm 6 Conjugate gradient algorithm

1. For a least-squares problem $C_s\bar{x} = L_s$, where $C_s = C_s^*$, $\bar{x}$ is unknown.
2. Start with a random initial solution $\bar{x}_0$.
3. $r_0 = L_s - C_s\bar{x}_0$, $p_0 = r_0$.
4. for $k = 0$, repeat
5. $\alpha_k = \frac{r_k^* r_k}{p_k^* C_s p_k}$,
   $\bar{x}_{k+1} = \bar{x}_k + \alpha_k p_k$,
   $r_{k+1} = r_k - \alpha_k C_s p_k$,
   if r_{k+1} is sufficient small then exit loop.
   $\beta_k = \frac{r_{k+1}^* r_{k+1}}{r_k^* r_k}$,
   $p_{k+1} = r_{k+1} + \beta_k p_k$,
   $k = k + 1$,
   end repeat.
6. The optimal estimation is x_{k+1}.

rameters: the choice of M and N_o, N_i. The following proposition considers the total errors of input autocorrelations we recover.

Theorem 9 *Denote $R_{uu}(m)$ as the "true" input autocorrelations, $\hat{R}_{uu}(m)$ as the input autocorrelation function we estimate from the output autocorrelations, and let $\Delta(m) = R_{uu}(m) - \hat{R}_{uu}(m)$ be the error between the estimated input autocorrelation and the "true" input autocorrelation. We assume that $\|h_i\| \leq \delta, i > M$, $\|R_{uu}(m)\| \leq \delta, |m| > N_i$, and $\|\hat{R}_{yy}(m)\| \leq \delta, |m| > N_o$ where δ is small enough. Then $\|\Delta(m)\| \leq k\delta$, where k is some constant.*

Proof 10 *Denote $R_{uu}^M(m)$ as the input autocorrelations we extract by using M Markrov parameters of the dynamic system, $\Delta_M(m) = R_{uu}(m) - R_{uu}^M(m)$ as the error of the input autocorrelations resulting choosing M such that $\|h_i\| \leq \delta, i > M$. Denote $R_{uu}^N(m)$ as the input autocorrelations we extract by using N_i input autocorrela-*

tions, and N_o output autocorrelations, and $\Delta_N(m) = R_{uu}(m) - R^N_{uu}(m)$ as the errors from choosing design parameters N_i, N_o, such that $\|R_{uu}(m)\| \leq \delta, |m| > N_i$, and $\|\hat{R}_{yy}(m)\| \leq \delta, |m| > N_o$. First, we analyze the errors separately.

The output autocorrelation function using the first M Markov parameters is:

$$\hat{R}^M_{yy}(m) = \sum_{i=1}^{M}\sum_{j=1}^{M} h_i R_{uu}(m+i-j)h_j^*. \tag{4.24}$$

Comparing with (4.5), the output autocorrelation errors resulting from using M Markov parameters is:

$$\begin{aligned}\Delta_1(m) = &\sum_{i=M+1}^{\infty}\sum_{j=1}^{M} h_i R_{uu}(m+i-j)h_j^* + \sum_{i=M+1}^{\infty}\sum_{j=M+1}^{\infty} h_i R_{uu}(m+i-j)h_j^* \\ &+ \sum_{i=1}^{M}\sum_{j=M+1}^{\infty} h_i R_{uu}(m+i-j)h_j^*.\end{aligned} \tag{4.25}$$

By choosing M large enough, we have $\|h_i\| \leq \delta, i > M$, where δ is small enough, thus,

$$\begin{aligned}\|\Delta_1(m)\| \leq &\sum_{i=M+1}^{\infty}\sum_{j=1}^{M} \delta \times \|R_{uu}(m+i-j)\|\|h_j^*\| \\ &+ \sum_{i=M+1}^{\infty}\sum_{j=M+1}^{\infty} \delta \times \|R_{uu}(m+i-j)\| \times \delta \\ &+ \sum_{i=1}^{M}\sum_{j=M+1}^{\infty} \|h_i\|\|R_{uu}(m+i-j)\| \times \delta \\ &\leq k_1\delta,\end{aligned} \tag{4.26}$$

where k_1 is some constant.

Denote C_{yu} as the "true" coefficient matrix and C^M_{yu} as the coefficient matrix

using M Markov papameters, we need to solve the least squares problem:

$$vec(\hat{R}_{yy}) = C_{yu}^{M} vec(R_{uu}^{M}). \tag{4.27}$$

where R_{uu}^{M} is the input autocorrelation we recover from using M Markov parameters, and $vec(\hat{R}_{yy})$ is defined in (4.17).

Since

$$\|vec(\hat{R}_{yy}(m)) - vec(\hat{R}_{yy}^{M}(m))\|_2 = \|\hat{R}_{yy}(m) - \hat{R}_{yy}^{M}(m)\| = \|\Delta_1(m)\| \leq k_3\delta, \tag{4.28}$$

we have $vec(\hat{R}_{yy}(m)) = vec(\hat{R}_{yy}^{M}(m)) + \Delta_2(m)$, where $\|\Delta_2(m)\|_2 \leq k_1\delta$, or equivalently

$$vec(\hat{R}_{yy}) = vec(\hat{R}_{yy}^{M}) + \Delta_2, \tag{4.29}$$

Consider (4.17), $vec(\hat{R}_{yy})$ and $vec(\hat{R}_{yy}^{M})$ can be written as:

$$\begin{aligned} vec(\hat{R}_{yy}) &= C_{yu} vec(R_{uu}), \\ vec(\hat{R}_{yy}^{M}(m)) &= C_{yu}^{M} vec(R_{uu}), \end{aligned} \tag{4.30}$$

Substitute into (4.29), we have:

$$C_{yu} vec(R_{uu}) - C_{yu}^{M} vec(R_{uu}) = \Delta_2. \tag{4.31}$$

Since $(C_{yu}^{M})^{-1}$ exists, we have:

$$vec\ (R_{uu}) - vec(R_{uu}^{M}) = (C_{yu}^{M})^{-1}\Delta_2, \tag{4.32}$$

which implies:

$$\|vec(R_{uu}) - vec(R_{uu}^M)\|_2 \leq k_M \delta, \tag{4.33}$$

where k_M is some constant. Hence, $\|\Delta_M(m)\| \leq k_M \delta$.

Equation (4.9) can be seperated into two parts:

$$\begin{aligned} vec(\hat{R}_{yy}(m)) = &\sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \bar{h}_j \otimes h_i vec(\underbrace{R_{uu}(m+i-j)}_{|m+i-j|\leq N_i}) \\ &+ \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \bar{h}_j \otimes h_i vec(\underbrace{R_{uu}(m+i-j)}_{|m+i-j|> N_i}). \end{aligned} \tag{4.34}$$

Thus, it can be written as:

$$vec(\hat{R}_{yy}(m)) = vec(\hat{R}_{yy}^N(m)) + \Delta_4(m), \tag{4.35}$$

where

$$\begin{aligned} \|\Delta_4(m)\|_2 &= \|\sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \bar{h}_j \otimes h_i vec(\underbrace{R_{uu}(m+i-j)}_{|m+i-j|> N_i})\|_2 \\ &\leq \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \|\bar{h}_j \otimes h_i\|_2 \times \delta \\ &\leq k_4 \delta, \end{aligned} \tag{4.36}$$

where k_4 is some constant. Following the same procedure above, it can be proved that $\|\Delta_N(m)\| \leq k_N \delta$, where k_N is some constant.

Denote output autocorrelation in (4.16) as $\hat{R}_{yy}^c(m)$, comparing (4.16) with (4.9),

the output autocorrelation error is:

$$vec(\hat{R}_{yy}) - vec(\hat{R}^c_{yy}) = \Delta_2 + \sum_{i=1}^{M}\sum_{j=1}^{M} \bar{h}_j \otimes h_i vec(\underbrace{R_{uu}(m+i-j)}_{|m+i-j|>N_i})$$

$$\leq \Delta_2 + \Delta_4. \tag{4.37}$$

Thus

$$\|vec(\hat{R}_{yy}) - vec(\hat{R}^c_{yy})\|_2 \leq \|\Delta_2\|_2 + \|\Delta_4\|_2 \leq k_5\delta, \tag{4.38}$$

where k_5 is some constant. Following the same procedure, we can prove:

$$\|\Delta(m) = R_{uu}(m) - \hat{R}_{uu}(m)\| \leq k\delta. \tag{4.39}$$

The results above show that if M, N_i, N_o are chosen large enough, the errors in estimating the input autocorrelations can be made arbitrarily small.

4.3.2 Construction of the AR Based Innovations Model

After we extract the input autocorrelations from the output autocorrelations, we want to construct a system which will generate the same statistics as the ones we recovered in Section 4.3.1.

If assumption A4 is satisfied, i.e., $\{u_k\}$ is WSS with a rational power spectrum, the power spectrum of u_k is continuous, and can be modelled as the output of a casual linear time invariant system driven by white noise [92]. Such system can be constructed by using an autoregressive moving average (ARMA) model, and in practice, a MA model can often be approximated by a high-order AR model, and thus, with enough coefficients, any stationary process can be well approximated by

using either AR or MA models (Chapter 9, [93]), and in this research, we use an AR model to fit the data.

In an AR model, the time series can be expressed as a linear function of its past values, i.e.,

$$u(k) = \sum_{i=1}^{M_i} a_i u(k-i) + \epsilon(k), \tag{4.40}$$

where $\epsilon(k)$ is white noise with distribution $N(0, \Omega_r)$, M_i is the order of the AR model, and $a_i, i = 1, 2, \cdots, M_i$ are the coefficient matrices.

For a vector autoregressive model with complex values, the Yule-Walker equation [94] which is used to solve for the coefficients needs to be modified. The modified Yule-Walker equation can be written as:

$$\begin{pmatrix} R_{uu}(-1) & R_{uu}(-2) & \cdots & R_{uu}(-M_i) \end{pmatrix} = \\ \begin{pmatrix} a_1 & a_2 & \cdots & a_{M_i} \end{pmatrix} \begin{pmatrix} R_{uu}(0) & \cdots & R_{uu}(1-M_i) \\ R_{uu}(1) & \cdots & R_{uu}(2-M_i) \\ \vdots & \vdots & \vdots \\ R_{uu}(M_i-1) & \cdots & R_{uu}(0) \end{pmatrix}. \tag{4.41}$$

Equation (4.41) is used to solve for the coefficient matrices $a_i, i = 1, 2, \cdots, M_i$.

The covariance of the residual white noise $\epsilon(k)$ can be solved using the following equation:

$$R_{\epsilon\epsilon}(m) = R_{uu}(m) - \sum_{i=1}^{M_i} \sum_{j=1}^{M_i} a_i R_{uu}(m+i-j) a_j^*, \tag{4.42}$$

where $\Omega_r = R_{\epsilon\epsilon}(0)$.

The balanced minimal realization for the AR model (4.40) can be expressed as:

$$\begin{aligned} \eta_k &= A_n \eta_{k-1} + B_n u_{k-1}, \\ u_k &= C_n \eta_k + \epsilon_k, \end{aligned} \tag{4.43}$$

where (A_n, B_n, C_n) are solved by using the ERA technique [95] with $a_i, i = 1, \cdots, M_i$ as the Markov parameters of the system.

Equation (4.43) is equivalent to:

$$\begin{aligned} \eta_k &= (A_n + B_n C_n)\eta_{k-1} + B_n \epsilon_{k-1}, \\ u_k &= C_n \eta_k + \epsilon_k, \end{aligned} \tag{4.44}$$

where ϵ_k is white noise with covariance Ω_r. We make the following remark.

Remark 7 *We need to find a stable $A_n + B_n C_n$ in (4.44). In practice, we calculate the Markov parameters of system (4.44) using $a_i, i = 1, \cdots, M_i$ first, and then use the ERA for the state space realization. If the Markov parameters of system (4.44) are $\hat{a}_i, i = 1, \cdots, M_i$, then $\hat{a}_1 = C_n B_n = a_1, \hat{a}_2 = C_n(A_n + B_n C_n)B_n = a_2 + a_1 a_1, \cdots$. As we explained before, for a WSS process with rational power spectrum, from [92] , we can always find a stable realization $(A_n + B_n C_n, B_n, C_n)$.*

By using the Cholesky decomposition, we can find a unique lower triangular matrix P such that: $\Omega_r = PP^*$. If w_k is white noise with distribution $N(0, 1)$, then Pw_k would be white noise with distribution $N(0, \Omega_r)$. Thus, the innovations model

we construct that has the same statistics as the unknown input system (4.2) is:

$$\eta_k = (A_n + B_n C_n)\eta_{k-1} + B_n P w_{k-1},$$
$$u_k = C_n \eta_k + P w_k, \tag{4.45}$$

where w_k is a randomly white noise with standard normal distribution.

Under assumption A4, the following result holds.

Lemma 3 *Denote $\hat{R}_{uu}(m)$ as the input autocorrelations recovered from the measurements, then $\hat{R}_{uu}(m)$ can be reconstructed exactly by using the innovations model (4.45), i.e., $\tilde{R}_{uu}(m) = \hat{R}_{uu}(m)$, where $\tilde{R}_{uu}(m)$ is the input autocorrelations of the realization of system (4.45).*

From Theorem 9 and Lemma 3, under the same assumptions, the following corollary immediately follows.

Corollary 4 *Denote u_k as the actual unknown input process, and $R_{uu}(m)$ as the actual input autocorrelation function. Then $\|\tilde{R}_{uu}(m) - R_{uu}(m)\| \leq k_a \delta$, where k_a is some constant, when δ is small enough. System (4.45) is an innovations model for the unknown input u_k.*

The procedure of constructing the innovations model is summarized in Algorithm 7.

Remark 8 *The AR model based unknown input realization technique we proposed can also be used when the unknown inputs affect both the states and outputs, i.e.,*

$$x_{k+1} = Ax_k + Bu_k,$$
$$y_k = Cx_k + Du_k + v_k, \tag{4.46}$$

Algorithm 7 AR model based unknown input realization technique

1. Choose a finite number N_o, compute output autocorrelation function $R_{yy}(m)$ by using measurements y_k, $|m| \leq N_o$.
2. Choose a finite number M, construct the coefficient matrix C_{yu} from (4.16).
3. Choose a finite number N_i, solve the least squares problem (4.17) for unknown input autocorrelation function $R_{uu}(m)$, $|m| \leq N_i$.
4. Construct an AR model for the unknown input $u(k) = \sum_{i=1}^{M_i} a_i u(k-i) + \epsilon(k)$, find the coefficient matrices $a_i, i = 1, 2, \cdots M_i$ by solving the modified Yule-Walker equation (4.41).
5. Find the covariance Ω_r of $\epsilon(k)$ by solving (4.42).
6. Construct the state space representation (4.43) for the AR model using ERA.
7. Find a unique lower triangular matrix P such that $\Omega_r = PP^*$, and construct an innovations model as in (4.45).

where u_k is the stochastic unknown input, v_k is the measurement noise. The solution y_k can be written as:

$$y_k = \sum_{i=1}^{M} h_i u_{k-i} + Du_k + v_k, \tag{4.47}$$

and the relationship between output autocorrelations and input autocorrelations is:

$$\begin{aligned} vec(\hat{R}_{yy}(m)) = &\sum_{i=1}^{M}\sum_{j=1}^{M} \bar{h}_j \otimes h_i vec(R_{uu}(m+i-j)) + \sum_{i=1}^{M} \bar{D} \otimes h_i vec(R_{uu}(m+i)) \\ &+ \sum_{i=1}^{M} \bar{h}_j \otimes D vec(R_{uu}(m-j)) + \bar{D} \otimes D vec(R_{uu}(m)) \end{aligned} \tag{4.48}$$

where $\hat{R}_{yy}(m) = R_{yy}(m) - R_{vv}(m)$. Notice that the first term is the same as (4.16), and the last three terms correspond to the perturbation of the unknown inputs in the

outputs. It can also be formulated as a least squares problem (4.17), and an unknown input system may be realized following the same procedure as in Algorithm 7.

Remark 9 *For real valued system, we can save the computation by using the properties of autocorrelation functions:*

$$\begin{aligned} R_{u_i u_i}(-m) &= R_{u_i u_i}(m), \\ R_{u_i u_j}(-m) &= R_{u_j u_i}(m), i \neq j \end{aligned} \tag{4.49}$$

Thus, we only need to collect $N_o + 1$ output autocorrelations and have $p^2(N_o + 1)$ equations with $q^2(N_i + 1)$ unknowns in (4.17).

4.3.3 Extension to Estimate Unknown Input Locations

The AR model based unknown input realization technique we proposed can also be applied to estimate the locations of unknown inputs.

Consider system (4.1), where both B and u_k are unknown, but the locations of the unknown inputs are known to be in some region Ω, for example, the unknown inputs are known to be on boundaries, we can estimate the matrix B as well. There is an augmented matrix $\hat{B} \in R^{n \times \hat{p}}$, which includes all the possible locations of the unknown inputs, and without losing generality, we assume that rank $CA\hat{B}$ = rank $\hat{B} = \hat{p}$.

Compared with system (4.1), the augmented system is:

$$\begin{aligned} x_{k+1} &= Ax_k + \hat{B}\hat{u}_k, \\ y_k &= Cx_k + v_k, \end{aligned} \tag{4.50}$$

where $A, \hat{B}, C$ are known and $\hat{u}_k \in R^{\hat{p}}$ is the augmented unknown inputs. Following

the same procedure as in Algorithm 7, we can reconstruct the unknown input correlations. If the unknown input autocorrelations $\hat{R}_{u_i u_i}(m) = 0$, which means the i^{th} entry of $\hat{u}_k$ is zero, i.e., there is no input at the corresponding location in the actual system. Thus, B is estimated by using all the columns of $\hat{B}$ where the corresponding unknown inputs are not identically zero.

Notice that in the augmented system, we need more measurements, i.e., $q \geq \hat{p}$, which leads to a larger least-squares problem, and if $\hat{R}_{u_i u_i}(m)$ should be zero but not due to the numerical errors of solving least-squares problem, it may also affect the accuracy of the estimation of B matrix.

4.4 Augmented State Kalman Filter and Model Reduction

After we construct an innovations model for the unknown inputs, we apply the standard Kalman filter on the augmented system with states augmented by the unknown input states. A ROM based filter is also constructed using RPOD* for reducing the computational cost of the resulting filter.

4.4.1 Augmented State Kalman Filter

The full order system can be represented by augmenting the states of the original system as:

$$\begin{pmatrix} x_{k+1} \\ \eta_{k+1} \end{pmatrix} = \begin{pmatrix} A & BC_n \\ 0 & A_n + B_n C_n \end{pmatrix} \begin{pmatrix} x_k \\ \eta_k \end{pmatrix} + \begin{pmatrix} BP \\ B_n P \end{pmatrix} w_k,$$
$$y_k = \begin{pmatrix} C & 0 \end{pmatrix} \begin{pmatrix} x_k \\ \eta_k \end{pmatrix} + v_k, \tag{4.51}$$

where w_k is white noise with standard normal distribution. v_k is white noise with known covariance.

Remark 10 *The augmented state system (4.51) is stable and detectable. The eigenvalues of the augmented system (4.51) are the eigenvalues of A and the eigenvalues of $A_n + B_n C_n$. From assumption A1, A is stable, from Remark 7, $A_n + B_n C_n$ is stable, and hence, the augmented system (4.51) is stable. From assumption A1, system (4.1) is detectable, and from the asymptotic stability of matrix $A_n + B_n C_n$, (4.44) is also detectable, therefore, all the unobservable modes in (4.51) are asymptotically stable, which implies that (4.51) is detectable. Thus, we may now use the standard Kalman filter for state estimation of the augmented system (4.51).*

4.4.2 Unknown Input Estimation Using Model Reduction

For large scale systems as we would like to consider, in particular, the systems governed by partial differential equations (suitably discretized), we can use model reduction technique such as RPOD* to construct a ROM first, and then extract the input autocorrelations from the reduced order model. We apply the Kalman filter to the ROM to reduce the computational cost.

The ROM system is extracted from the full order system using the RPOD* and is denoted by:

$$\begin{aligned} a_k &= A_r a_{k-1} + B_r u_{k-1}, \\ y_k &= C_r a_k + v_k. \end{aligned} \tag{4.52}$$

Let $\hat{h}_i = C_r A_r^{i-1} B_r, i = 1, 2, \cdots, M$ be the Markov parameters of the ROM. Then the relationship between input autocorrelations and output autocorrelations can be written as:

$$\hat{R}_{yy}(m) = \sum_{i=1}^{M} \sum_{j=1}^{M} \hat{h}_i R_{uu}(m + i - j) \hat{h}_j^*. \tag{4.53}$$

Following the same procedure as in Algorithm 7, we can now recover the input autocorrelations, and construct an innovations model which can generate the same statistics as the unknown inputs.

The advantage of using model reduction is that for a large scale system, computing $\hat{h}_i = C_r A_r^{i-1} B_r$ is much faster than computing $h_i = CA^{i-1}B$ because of the reduction in the size of A. Also, the order of the ROM is much smaller than the order of the full order system, and thus the computational cost of using the Kalman filter is much reduced. Hence, even with the augmented states, the standard Kalman filter remains computationally tractable.

Remark 11 *To reduce the computational cost of the augmented states in Kalman filter, we can also use the existing optimal two-stage or three-stage Kalman filtering technique [54, 56], which decouple the augmented filter into two parallel reduced order filters. These techniques are preferable when the order of the innovations model is high, while the RPOD based ROM filter is preferable when the order of the dynamic system is high.*

4.5 Computational Results

We test the AR model based algorithm on several advection-diffusion equations. First, we compare the simulations results using full order model and the ROM constructed by RPOD* algorithm. We check the results by comparing the autocorrelation functions of the inputs, outputs and the states. And we show the state estimation using Kalman filter. Also, we compare the performance of AR model based algorithm with OTSKF and UMV algorithms

Denote the average root mean square error(ARMSE) as:

$$ARMSE = \frac{1}{n}\sum_{i=1}^{n}\sqrt{\frac{\sum_{k=1}^{n}(\hat{x}_i(k) - x_i(k))^2}{n}}, \tag{4.54}$$

where $\hat{x}_i(k)$ is the state estimate $\hat{x}_i$ at time t_k, and $x_i(k)$ is the true state x_i at time t_k, where i denotes the i^{th} component of the state vector.

Suppose at the state component x_i, the measurement noise v_k is a white noise with zero mean and covariance Ω_i. We define a noise to signal ratio (NSR):

$$NSR = \sqrt{\frac{|\Omega_i|}{(E[x_i x_i^*])}}. \tag{4.55}$$

4.5.1 Heat Problem

The heat transformation along a slab is given by the partial differential equation:

$$\begin{aligned} &\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} + f, \\ &T|_{x=0} = 0, \frac{\partial T}{\partial x}|_{x=L} = 0, \end{aligned} \tag{4.56}$$

where α is the thermal diffusivity, $L = 1m$, and f is the unknown forcing. There are two point sources located at $x = 0.5m$ and $x = 0.6m$.

The system is discretized using finite difference approach, and there are 50 grids which are equally spaced.

Measurement Model. To satisfy the observer matching condition in the UMV algorithm, we take two measurements at $x = 0.5m$, $x = 0.6m$. The measurement noise is white noise with covariance $0.1I_{2\times 2}$.

Unknown Input Model. In the simulation, the unknown inputs are generated

by a second order model:

$$\xi_k = \begin{pmatrix} 0.3 & 0.5 \\ 0.4 & 0.2 \end{pmatrix} \xi_{k-1} + I_{2\times 2}\mu_k,$$
$$u_k = I_{2\times 2}\xi_k, \tag{4.57}$$

and $\mu_k \sim N(0, 10I_{2\times 2})$.

Choosing Design Parameters. The design parameters $M = 4000$, $N_i = 200$, $N_o = 2000$ are chosen as follows. M is chosen so that the Markov parameters $\|h_i\| \approx 0, i > M$. N_i and N_o are chosen by trial and error. First, we randomly choose a suitable N_i and N_o, where $N_i \leq N_o$. Then we follow the AR based unknown input realization procedure, and construct the augmented state system (4.51). Given the white noise processes w_k, v_k perturbing the system, we check the output statistics of the augmented state system (4.51). If the errors are small enough, we stop, otherwise, we increase the values of N_i and N_o, and repeat the same procedure until the errors are negligible. Notice that increasing M, N_i, N_o would increase the accuracy of the input statistics we can recover, but also increases the computational cost.

First, in Figure 4.1, we show the comparison of the input correlations we recover with the actual input correlations. Since there are two inputs, thus, the cross-correlation function between input 1 and input 2 are also included.

It can be seen that the statistics of the unknown inputs can be recovered almost perfectly, and given the system perturbed by the unknown inputs innovations model we constructed, the statistics of the outputs and the states are almost the same as well.

Next, we compare the performance of the unknown inputs constructed using the ROM with the full order system. The full order system has 50 states, and the ROM

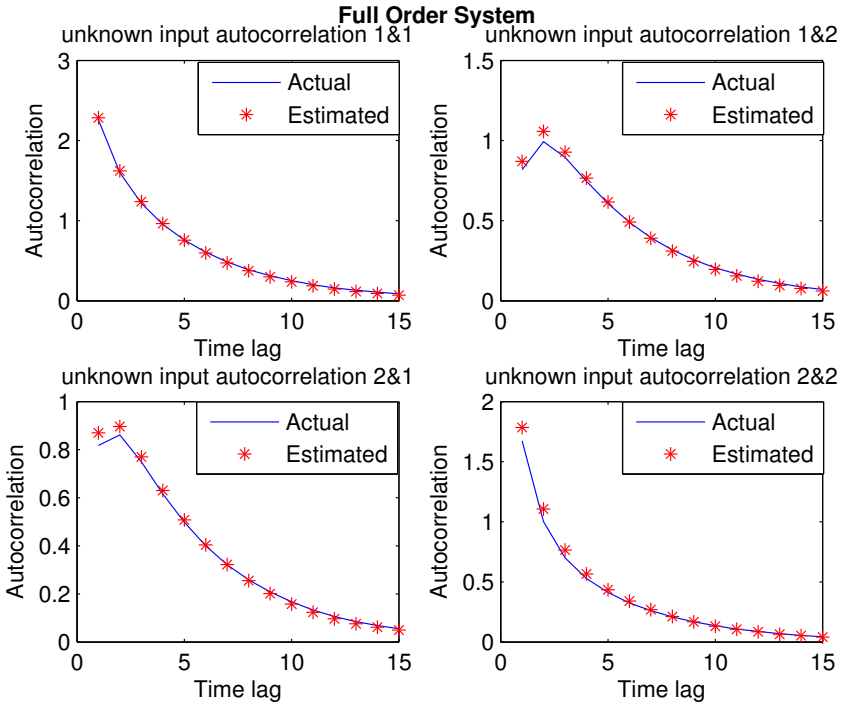


Figure 4.1: Comparison of the recovered input autocorrelations with actual unknown input autocorrelations for heat problem.

has 20 states. The relative error of the input correlation is shown in Figure 4.2.

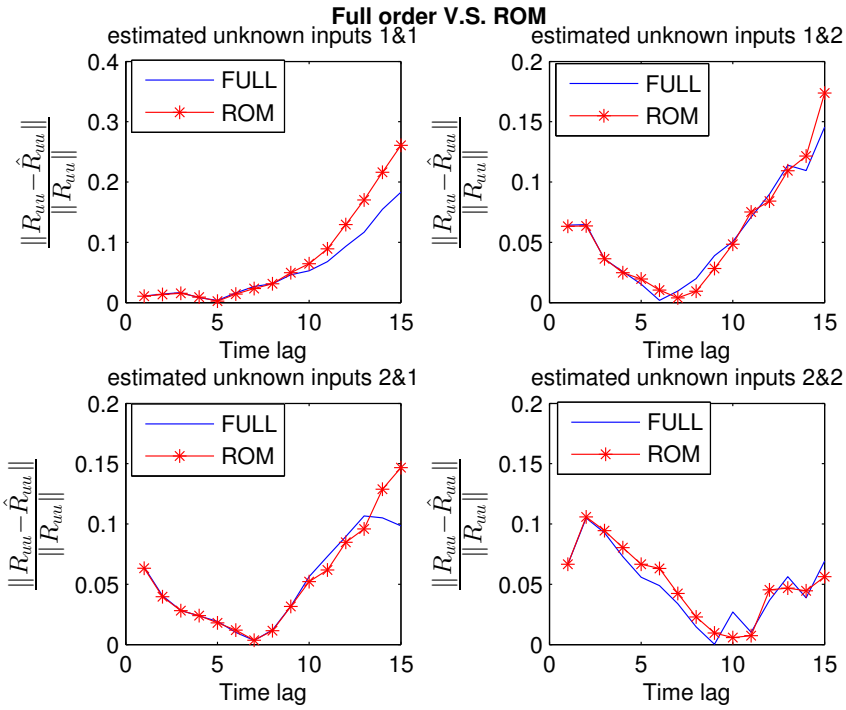


Figure 4.2: Comparison of input autocorrelation relative error using full order model with ROM for heat problem.

We can see that the statistics reconstructed by using the ROM is not as accurate as using the full order system, however, the relative error is on the same scale, and hence, the computational cost is reduced without losing much accuracy.

The state estimation using ROM is shown in Figure 4.3. We randomly choose two states and show the comparison of the actual state with the estimated states. The state estimation error and 3σ bounds are shown. It can be seen that the Kalman filter using the ROM performs well, and hence, for a large scale system, the computational complexity of ASKF can be reduced by using the RPOD.

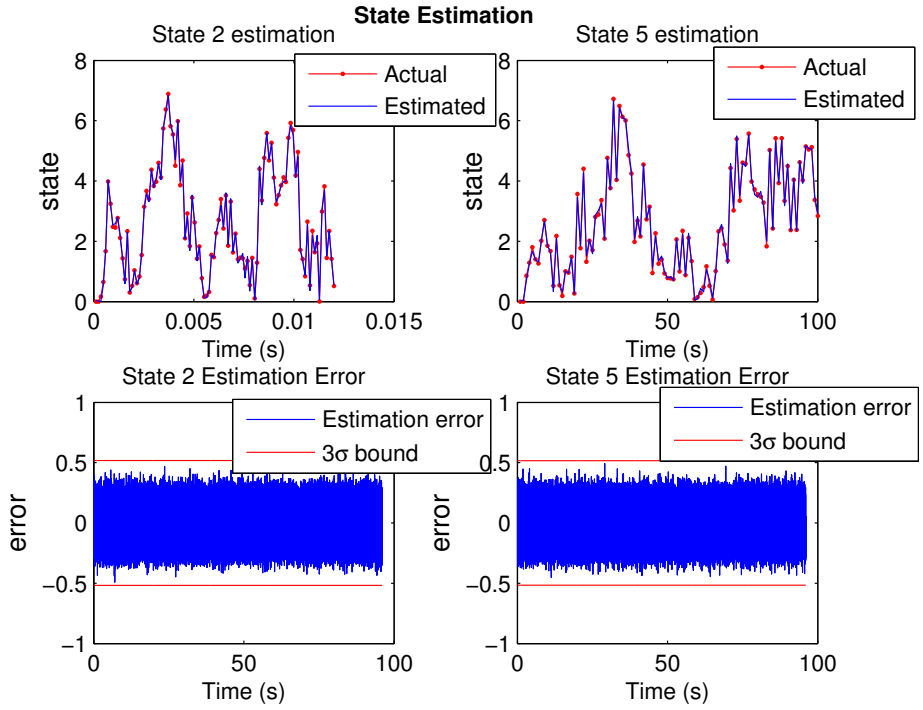


Figure 4.3: Unknown input filtering for heat problem using ROM. State estimation error and 3σ bounds for two randomly chosen states.

Next, we compare the performances of the AR model based algorithm with OTSKF and UMV algorithms. The OTSKF and UMV algorithms we use are summarized in Appendix C and Appendix D.

OTSKF model. The assumed unknown input model used in the OTSKF is not the same as the true model, in particular, the system matrices of the input system are perturbed from the true values, the model used for OTSKF is:

$$\eta_{k+1} = A_o \eta_k + v_k = \begin{pmatrix} 0.4569 & 0.2768 \\ 0.2214 & 0.4016 \end{pmatrix} \eta_k + v_k, \tag{4.58}$$

where $v_k \sim N(0, 10I_{2\times 2})$. Here, A_o is chosen as follows. The eigenvalues of A_e in (4.57) are $0.7, -0.2$. We perturb the eigenvalues of A_e with randomly generated numbers between $[-0.3, 0.3]$ and $[-0.8, 0.8]$ with uniform distribution respectively, and keep the eigenvectors same as the eigenvectors of A_e. The perturbed eigenvalues are $0.6783, 0.1802$. We calculate the output statistics of (4.57) and (4.58), and we can see that the unknown input statistics used in OTSKF are perturbed by 5% about the true value.

The estimation of the initial state $\bar{x}_0$ and covariance $\bar{P}_0$ in three algorithms are the same.

We vary the measurement noise covariance Ω_i, and for each Ω_i, a Monte Carlo simulation of 10 runs is performed to compare the magnitude of the ARMSE using AR model based algorithm with the OTSKF and UMV algorithms in Table 4.2.

The comparison is shown in Figure 4.4. It can be seen that the AR model based method performs the best. Note that when the assumed unknown input model used in OTSKF is not accurate, the performance of AR model based algorithm is much better while with increase in the sensor noise, the performance of the AR model based

Table 4.1: Performances of the AR model based algorithm, OTSKF and UMV for Heat Problem

NSR	AR model based	OTSKF	UMV
0.2215%	0.0036	0.0111	0.0033
6.8704%	0.0832	0.2418	0.0874
13.5171%	0.1309	0.3955	0.1528
20.3456%	0.3810	0.6516	0.4332
26.9467%	0.4190	0.7141	0.5112

algorithm gets better than the UMV algorithm. It should also be noted that when the sensors and the unknown inputs are non-collocated, the "observer matching" condition is not satisfied, and hence, the UMV algorithm can not be used, while the OTSKF and the AR model based algorithm are not affected.

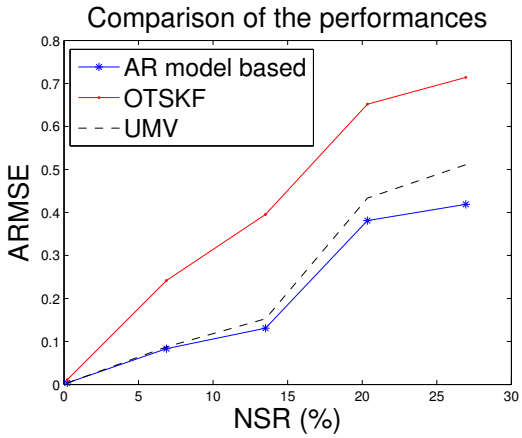


Figure 4.4: Comparison of the performances of AR model based algorithm with OTSKF and UMV algorithms for heat transfer problem. The ARMSE is plotted as a function of NSR.

4.5.2 Stochastically Perturbed Laminar Flow

Consider the three-dimensional flow between two infinite plates (at $y = \pm 1$) driven by a pressure gradient in the streamwise x direction with periodic boundary condition in x- and z-directions. The equations are given as follows.

$$\begin{aligned}
&[(\frac{\partial}{\partial t} + U\frac{\partial}{\partial x})\nabla^2 - U''\frac{\partial}{\partial x} - \frac{1}{Re}\nabla^4]v = 0 \\
&[\frac{\partial}{\partial t} + U\frac{\partial}{\partial x} - \frac{1}{Re}\nabla^2]\eta = -U'\frac{\partial v}{\partial z}
\end{aligned} \tag{4.59}$$

The mean velocity profile is given by $U(y) = 1 - y^2$. At each wavenumber pair $(\alpha, \beta)_{mn}$, the wall-normal velocity $v(x, y, z, t)$ and wall-normal vorticity $\eta(x, y, z, t)$ are:

$$\begin{aligned}
v(x, y, z, t) &= \hat{v}_{mn}(y, t)e^{i(\alpha x + \beta z)}, \\
\eta(x, y, z, t) &= \hat{\eta}_{mn}(y, t)e^{i(\alpha x + \beta z)}.
\end{aligned} \tag{4.60}$$

Denote

$$\hat{q}_{mn}(y, t) = \begin{pmatrix} \hat{v}_{mn}(y, t) \\ \hat{\eta}_{mn}(y, t) \end{pmatrix}, \tag{4.61}$$

where $\hat{(.)}$ denotes the Fourier transformed variable, and $(.)_{mn}$ denotes the wavenumber pair $(\alpha, \beta)_{mn}$.

The evolution of the flow in Fourier domain can be written as:

$$\frac{d}{dt}M\hat{q}_{mn} + L\hat{q}_{mn} = \underbrace{\sum_{k+l=m, l+j=n} N(\hat{q}_{kl}, \hat{q}_{ij})}_{\text{nonilinear coupling}} + \underbrace{\hat{e}_{mn}(y, t)}_{external forcing}, \tag{4.62}$$

where

$$M = \begin{pmatrix} -\Delta & 0 \\ 0 & I \end{pmatrix}, \tag{4.63}$$

$$L = \begin{pmatrix} -i\alpha U\Delta + i\alpha U'' + \Delta^2/Re & 0 \\ i\beta U' & i\alpha U - \Delta/Re \end{pmatrix}. \tag{4.64}$$

The Laplacian operator is denoted as $\Delta = D^2 - k^2$, where D and D^2 represent first order and second order differentiation operators in the wall-normal direction, and $k^2 = \alpha^2 + \beta^2$.

The boundary conditions on v and η correspond to no-slip solid walls

$$v(\pm 1) = Dv(\pm 1) = \eta(\pm 1) = 0. \tag{4.65}$$

The flow model can be written as

$$\frac{d}{dt} M\hat{q}_{mn} + L\hat{q}_{mn} = Tf(y,t). \tag{4.66}$$

Operator T transforms the forcing $f = (f_1, f_2, f_3)^T$ on the evolution equation for the velocity vector $(u, v, w)^T$ into an equivalent forcing on the $(v, \eta)^T$ system, where

$$T = \begin{pmatrix} i\alpha D & k^2 & i\beta D \\ i\beta & 0 & -i\alpha \end{pmatrix}, \tag{4.67}$$

The derivation of T can be found in [16]. The forcing $f(y,t)$ accounts for the nonlinear terms and the external disturbances via an unknown stochastic model and

is assumed to be colored in time, i.e.,

$$\dot{\xi} = A_e\xi + B_e\nu,$$
$$f = C_e\xi + \mu \tag{4.68}$$

where ν, μ are uncorrelated white noise processes.

System (4.62) can be discretized using Chebyshev polynomials, and in the simulation, we assume there are two unknown inputs and two measurements.

Unknown input model. The unknown input f is assumed to be a colored noise generated by a third order linear complex system (4.2). The process noise ν_k is assumed to be white noise with zero mean and covariance $10I_{3\times 3}$.

The realization of the unknown inputs is a second order system. The measurement noise v_k is white noise with covariance $10I_{2\times 2}$.

Choosing design parameters. In the simulation, the design parameters $M = 1000$, $N_i = N_o = 100$ are chosen as follows. M is chosen so that the Markov parameters $\|h_i\| \approx 0, i > M$. N_i and N_o are chosen by trial and error. First, we randomly choose a suitable N_i and N_o, where $N_i \leq N_o$. Then we follow the AR based unknown input realization procedure, and construct the augmented state system (4.51). Given the white noise processes w_k, v_k perturbing the system, we check the output statistics of the augmented state system (4.51). If the errors are small enough, we stop, otherwise, we increase the values of N_i and N_o, and repeat the same procedure until the errors are negligible. Notice that increasing M, N_i, N_o would increase the accuracy of the input statistics we can recover, but also increases the computational cost.

First, in Fig. 4.5, we show the comparison of the input correlations we recover with the actual input correlations in complex plane. Since there are two inputs, thus,

the cross-correlation function between input 1 and input 2 are also included.

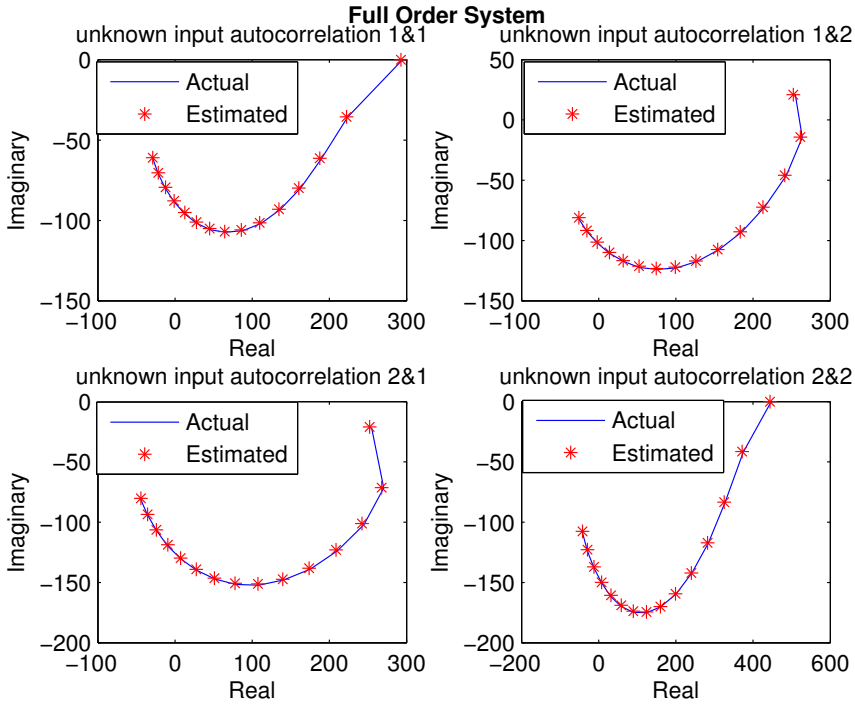


Figure 4.5: Comparison of recovered input autocorrelations with actual unknown input autocorrelations for stochastically perturbed laminar flow.

It can be seen that the statistics of the unknown inputs can be recovered almost perfectly, and given the system perturbed by the unknown inputs innovations model we constructed, the statistics of the outputs and the states are almost the same as well. The comparison of the statistics of outputs and states are omitted here.

Next, we compare the performance of the unknown inputs constructed using the ROM with the full order system. The full order system has 30 states, and the ROM has 15 states. The relative error of the input correlation is shown in Fig. 4.6, and the comparison of the relative error of output correlations and the state correlations are omitted here.

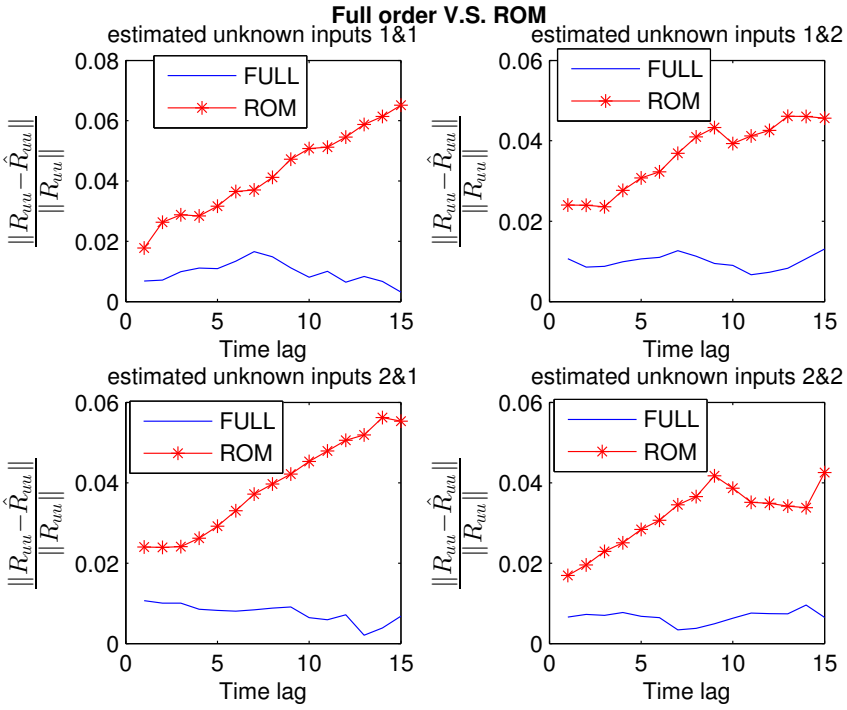


Figure 4.6: Comparison of input autocorrelation relative error using full order model and ROM for stochastically perturbed laminar flow.

We can see that the statistics reconstructed by using the ROM is not as accurate as using the full order system, however, the relative error is on the same scale, and hence, the computational cost is reduced without losing too much accuracy.

The state estimation using ROM is shown in Fig. 4.7. We randomly choose two states and show the comparison of the actual state with the estimated states. The state estimation error and 3σ bounds are shown. Since the error is complex valued, only the absolute value of the error is shown. The state estimation using full order system is a little better, and is omitted here due to the page limits. It can be seen that the Kalman filter using the ROM perform well, and hence, for a large scale system, the computational complexity of ASKF can be reduced by using the RPOD*.

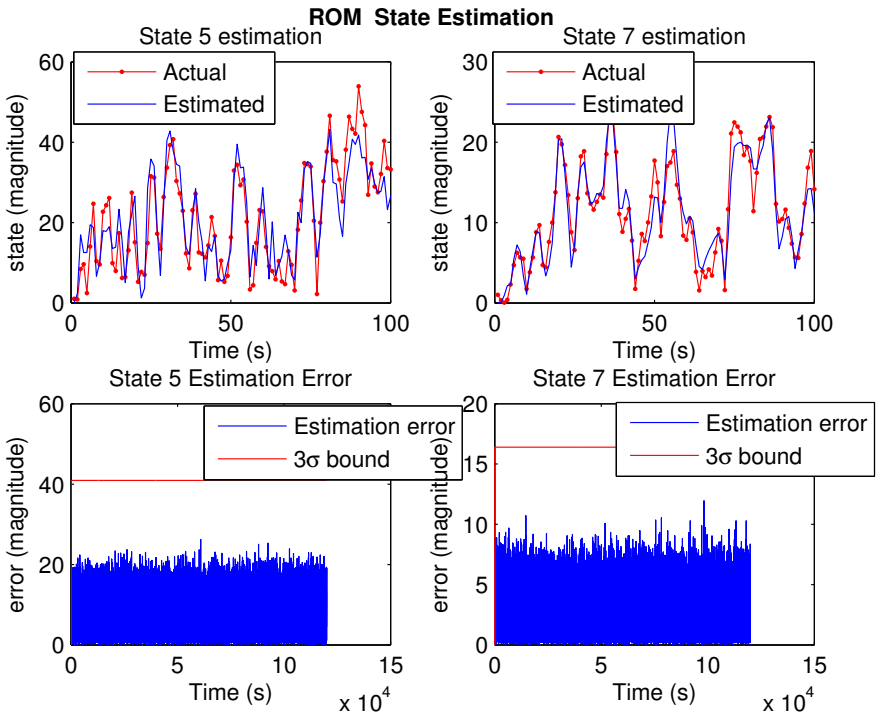


Figure 4.7: Unknown input filtering for stochastically perturbed laminar flow using ROM. State estimation errors and 3σ bounds for two randomly chosen states.

Next, we compare the performances of the AR model based algorithm with OTSKF and UMV algorithms. The assumed unknown input model used in OTSKF is the same as the true model, but the noise covariance is $6I_{3\times3}$. A Monte Carlo simulation of 10 runs is performed and we compare the magnitude of the ARMSE of three randomly chosen states in Table 4.2.

Table 4.2: Performances of the AR model based algorithm, UMV and OTSKF for stochastically perturbed laminar flow

ARMSE	AR model based	UMV	OTSKF
state 5	7.8637	9.9903	9.0868
state 7	3.1601	3.8012	3.3794
state 9	7.8377	10.0178	9.1438

We can see that when the assumed unknown input model used in OTSKF is not accurate, the performance of AR model based algorithm is the best of three algorithms, and when the sensor noise is not large, the performance of UMV is better than OTSKF. With the increase of the sensor noise, the performance of the UMV algorithm gets worse, and in Table 4.2, we can see that the performance of UMV is even worse than OTSKF.

4.6 Summary

In this section, we propose a balanced unknown input realization method for the state estimation of systems with unknown inputs that can be treated as a wide sense stationary process. We recover the unknown inputs statistics from the output data using a least squares procedure and then construct a balanced minimal realization of the unknown input using an AR model and the ERA technique. The recovered innovations model is used for the state estimation, and the standard Kalman filter is

applied to the augmented system. We compare the performances of the AR model based algorithm with the OTSKF and UMV algorithms, and we see that AR model based method performs the best.

The advantages of the AR model based algorithm we propose are summarized as follows. First, the performance of the AR model based algorithm is better than the ASKF, OTSKF and UMV algorithms when the unknown inputs can be treated as WSS processes with rational PSDs. The AR model based algorithm we propose constructs one particular realization of the true unknown input model, and the performance of the AR model based algorithm is the same as OTSKF when the assumed unknown input model used in OTSKF is accurate, and is better than UMV algorithm in the sense that the error covariances are smaller. With the increase of the sensor noise, we have seen that the performance of AR model based algorithm gets much better than the UMV algorithm. Second, a milder assumption than the "observer matching" condition needs to be satisfied. Also, to reduce the computational cost of the ASKF, we apply the RPOD* technique to construct a ROM for filtering.

5. GAUSSIAN PROCESS (GP) FOR STATE ESTIMATION

5.1 Introduction

Gaussian Process (GP), which is also known as Kriging interpolation technique [96], has been widely used to estimate and predict spatial phenomena. GP is a non-parametric generalization of linear regression algorithm, which only assumes that the spatial phenomena has a Gaussian distribution, while no prior information about the underlying dynamics of the phenomena needs to be known. As a Gaussian distribution is fully described by its mean and covariance matrix, a GP model is fully described by its mean function and covariance function, which can be represented using a small set of parameters (hyperparameters). For example, in [59], the temperature measurements in two-dimensional space are assumed to be spatially correlated, and can be modeled by a GP. A GP model is learned using sensor readings, and is used to estimate the temperature at arbitrary locations. How to place a limited number of sensors to generate a GP model is formulated as a sensor placement problem and has been studied [59, 60].

In this section, we consider the state estimation of spatio-temporal phenomena using GP model. In Section 5.2, we briefly review the GP, and in Section 5.3, we estimate the phenomena using spatial GP model. We show that predicting the spatio-temporal phenomena using spatial GP model is not accurate, and results in large estimation errors using two examples. We are aware that currently there is research focuses on constructing spatio-temporal GP models for application in field estimation. Therefore, in future work, we would like to compare the state estimation using spatio-temporal GP models with ROM based algorithm.

5.2 Preliminaries on GP

A Gaussian Process [63] is a collection of random variables, which any finite linear combinations of samples has a joint Gaussian distribution. As mentioned, GP can be used for estimating the spatial phenomena. In this section, first, we briefly review the GP, and then we introduce GP sensor placement problem.

Denote input $x = [x_1, x_2, \cdots, x_m]$ as a set of locations, and output $y = [y_1, y_2, \cdots, y_m]$ as the corresponding measurements at these locations. A Gaussian process $f(x)$ is fully described by the mean function $m(x)$, and covariance function $k(x_p, x_q)$, and can be written as: $f \sim GP(m(x), k(x_p, x_q))$. The covariance function is also known as kernel function.

Assume the sensor measurement has an additive independent identically distributed zero-mean Gaussian noise with variance σ_n^2, i.e.,

$$y = f(x) + \mu, \tag{5.1}$$

where $\mu \sim N(0, \sigma_n^2 I)$. Then the prior on the noisy observations becomes:

$$k_y(y_p, y_q) = k(x_p, x_q) + \sigma_n^2 \delta_{pq}, \tag{5.2}$$

where δ_{pq} is a Kronecker delta.

Given the measurements $Y_S = [y_1, \cdots, y_n]$ at locations $X_S = [x_1, \cdots, x_n]$, the GP can be used to predict the measurements Y_U at unobserved locations X_U. The posterior distribution conditioned on the measurements is given by:

$$\begin{aligned} m_{U|S} &= m_U + K(X_U, X_S)(K(X_S, X_S) + \sigma_n^2 I)^{-1}(Y_S - m_S), \\ K_{U|S} &= K(X_U, X_U) - K(X_U, X_S)(K(X_S, X_S) + \sigma_n^2 I)^{-1} K(X_S, X_U), \end{aligned} \tag{5.3}$$

where m_S, m_U are the mean vectors of Y_S and Y_U respectively, $K(.,.)$ is covariance matrix, and $K(X_S, X_S)$ denotes the $n \times n$ covariances evaluated at X_S, $K(X_U, X_S)$ denotes the cross covariances evaluated at all pairs of X_U and X_S, and $K(X_U, X_S) = K(X_S, X_U)$.

It can be seen that if GP model $f \sim GP(m(x), k(x_p, x_q))$ is known, then given measurements at y_S, the conditional distribution at unobserved locations can be predicted. This process is described as the Gaussian regression approach. Notice that the covariance $K_{U|S}$ does not depend on the actual measurements, and the inverse of an $n \times n$ matrix involves when calculating the posterior distribution, and as the number of measurements increases, the computational complexity of state estimation also increases.

Now the problem is reduced to finding the mean and covariance functions $m(x)$, $k(x_p, x_q)$. Normally, the mean and covariance functions are assumed to be some functions with a few free parameters, which are called the hyperparameters. For example, a commonly used covariance function in one dimension has the form:

$$k_y(x_p, x_q) = \sigma_f^2 exp(-\frac{1}{2l^2}(x_p - x_q)^2) + \sigma_n^2 \delta_{pq}, \tag{5.4}$$

where $\theta = (l, \sigma_f, \sigma_n)$ are the hyperparameters, and are determined by solving an optimization problem using training data.

Clearly, different choices of covariance functions, and how well the hyperparameters are selected would affect the accuracy of the prediction.

5.3 State Estimation Using Spatial GP Model

In this section, we show a one-dimensional and a two-dimensional heat problem to illustrate the state estimation of spatio-temporal system using GP model and its existing issues.

The state estimation problem we are interested is formulated as follows. We partition the spatial field into discrete grids, and take measurements at all possible locations at time t_0, where t_0 is large enough, and learn a GP model using these training data. Then the GP model is used to predict the states of the system after time t_0 using Kalman Filter. We also construct ROM for the system using RPOD* algorithm, and apply KF on the ROM for state estimation. At each time step, we place two sensors randomly in the field to take new measurements. Denote the prior mean and covariance at time t is $m_{t|t-1}$ and $K_{t|t-1}$, the posterior mean and covariance as $m_{t|t}, K_{t|t}$ respectively. Compared with the standard KF, the state estimation using the GP regression model $m(x), K(x, x')$ can be written as:

$$\begin{aligned} m_{t|t-1} &= m_{t-1|t-1}, \\ K_{t|t-1} &= K_{t-1|t-1}, \end{aligned} \tag{5.5}$$

and

$$\begin{aligned} m_{t|t} &= m_{t|t-1} + K_{t|t-1}C_t^T(C_tP_{t|t-1}C_t^T + R)^{-1}(y_t - C_tm_{t|t-1}), \\ K_{t|t} &= (I - K_{t|t-1}C_t^T(C_tP_{t|t-1}C_t^T + R)^{-1}C_t)K_{t|t-1}, \end{aligned} \tag{5.6}$$

with initial condition $m_{0|0} = m(x), K_{0|0} = K(x, x')$.

5.3.1 Computational Results: 1D Heat Problem

First, we consider the heat transfer in one-dimension space.

$$\frac{\partial T}{\partial t} = c\frac{\partial^2 T}{\partial x^2} + f, \tag{5.7}$$

where $T(x,t)$ is the temperature at location x and time t. The thermal diffusivity $c = 0.1$, and the spatial field is $x \in [0,1]$, and $f = 0$.

We partition x into 100 equi-spaced points. To learn a GP model, we take measurements at these 100 locations, and the mean function is chosen to be:

$$m(x) = a_0 + a_1 x + a_2 x^2. \tag{5.8}$$

The covariance function is chosen to be:

$$k(x,x') = b_0 + b_1^2 exp(-\frac{1}{2b_2^2}(x-x)^2), \tag{5.9}$$

Therefore, the hyperparameters needs to be estimated are $\{a_0, a_1, a_2, b_0, b_1, b_2\}$, and we estimate the hypeparameters using codes from [63]. In Fig. 5.1, we show the GP model with the training data.

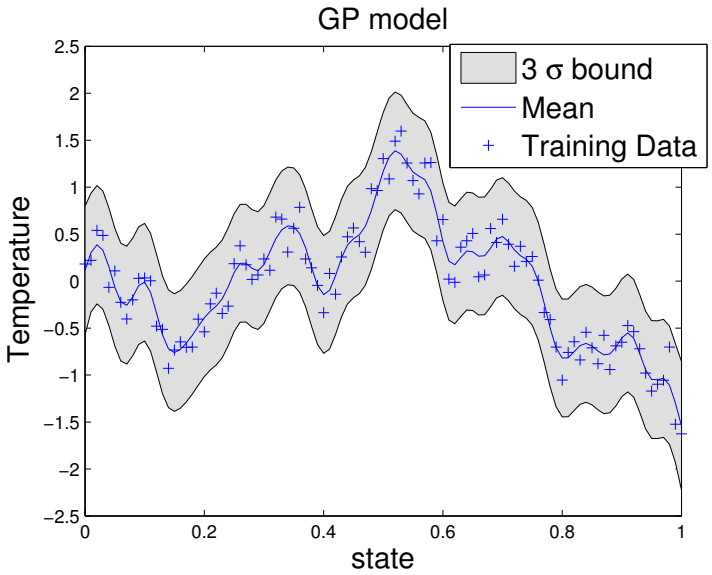


Figure 5.1: GP model learned from training data at time t_0 for 1D heat problem.

At time $t \geq t_0$, we predict the states of the system using new measurements. At each time step, we place two random moving sensors C_t to take measurements y_t. In Fig. 5.2, we compare the estimation error using KF on the ROM constructed using RPOD* and GP model.

5.3.2 Computational Results: 2D Heat Problem

We consider the prediction of the temperature in a two-dimensional space. The 2D heat transfer is described as follows.

$$\frac{\partial T}{\partial t} = c(\frac{\partial^2 T}{\partial x_1^2} + \frac{\partial^2 T}{\partial x_2^2}) + f, \tag{5.10}$$

where $T(x_1, x_2, t)$ is the temperature at position $x = (x_1, x_2)'$ and time t, the thermal diffusivity $c = 0.1$. The spatial field is $x_1 \in [0, 40]$, $x_2 \in [-10, 30]$, and $f = 5$ is a constant source at $(20, 10)$.

We partition x_1 and x_2 directions into 20 equi-spaced points, and take measurements at these 400 locations. The mean function is chosen to be:

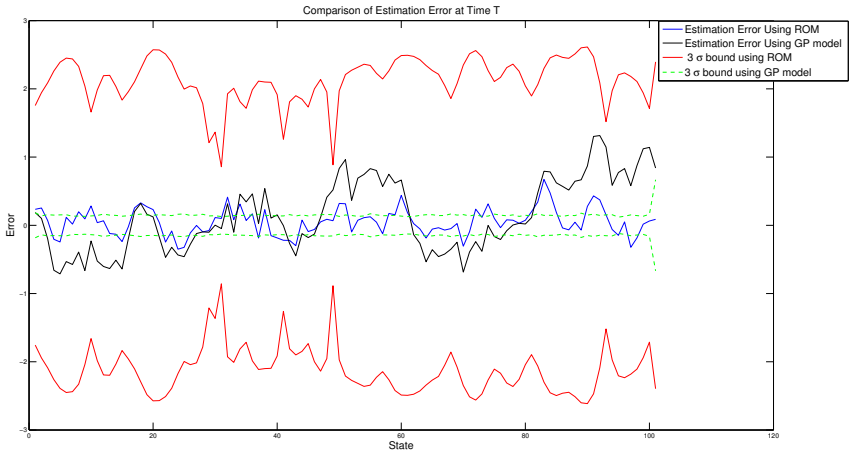
$$m(x) = a_0 + a_1 x_1 + a_2 x_1^2 + a_3 x_2 + a_4 x_2^2, \tag{5.11}$$

and kernel function is:

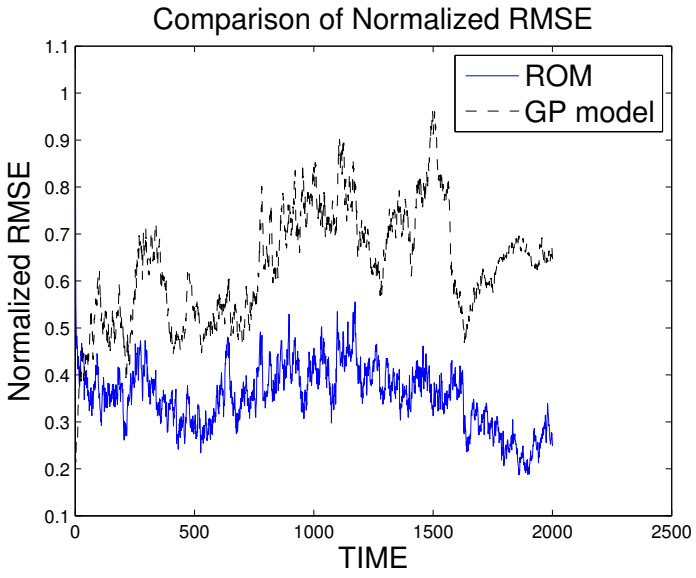
$$k(x, x') = b_0 + b_1^2 exp(-(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} x_1' \\ x_2' \end{pmatrix})' \begin{pmatrix} b_2 & \\ & b_3 \end{pmatrix}^{-1} (\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} x_1' \\ x_2' \end{pmatrix})/2), \tag{5.12}$$

Therefore, the hyperparameters needs to be estimated are $\{a_0, a_1, a_2, a_3, a_4, b_0, b_1, b_2, b_3\}$, and we estimate the hypeparameters using codes from [63].

The training data is plotted in Fig. 5.3(a). In Fig. 5.3(b), we plot the GP regression model, which is the mean function plus the 3σ bounds.

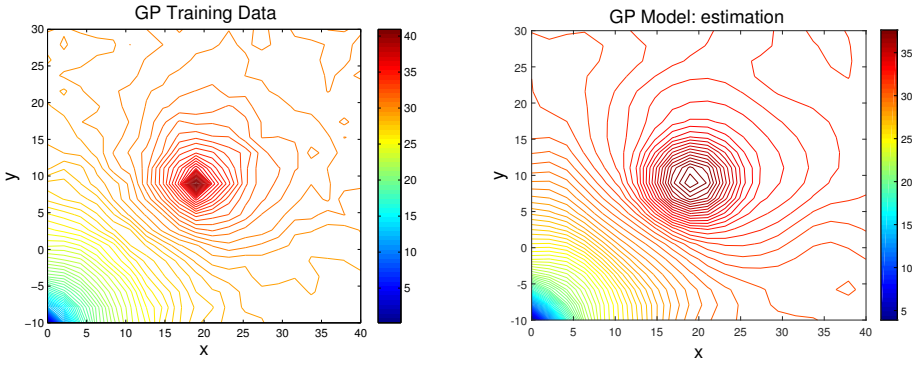


(a)



(b)

Figure 5.2: Comparison of state estimation at $t \geq t_0$ using GP model and ROM for 1D heat problem. (a) Comparison of state estimation error and 3σ bounds. (b) Comparison of normalized RMSE.



(a) (b)

Figure 5.3: GP model learned from training data at t_0 for 2D heat problem. (a) Training Data (b) GP model.

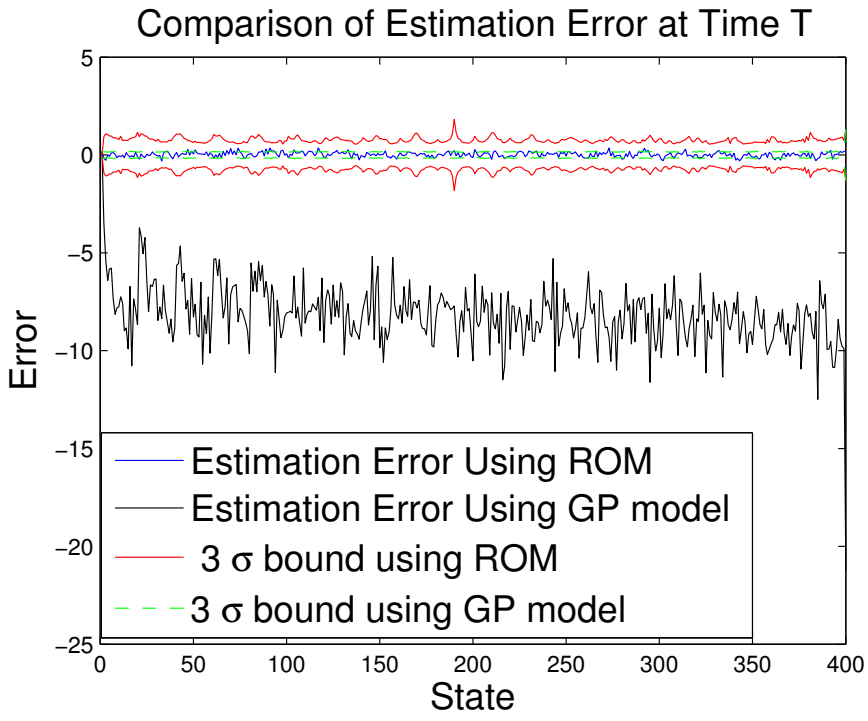
At time $t \geq t_0$, we predict the states of the system using new measurements. At each time step, we place two random moving sensors C_t to take measurements y_t. In Fig. 5.4, we compare the estimation error using KF on the ROM constructed using RPOD* and GP model.

It can be seen that GP estimation error is not within the 3σ bound, which is because GP covariance matrix converges independent of the actual measurements, and is not updated with the system dynamics. Therefore, as shown in Fig. 5.4(b), as time increases, the RMSE using GP model keeps increasing.

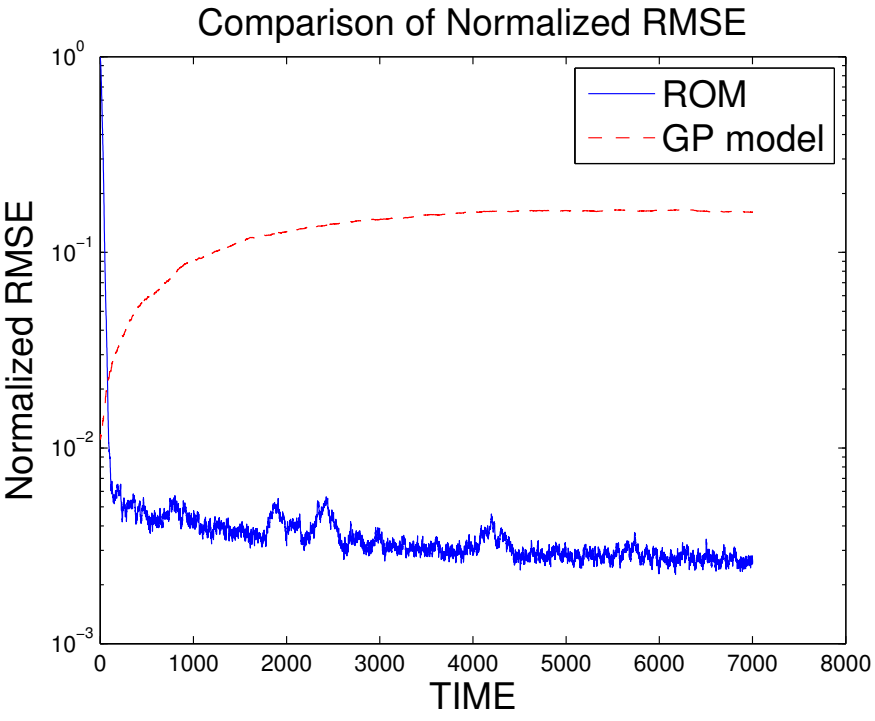
5.4 Summary

From the analysis, and the simulation results shown in this section, we can see that estimation of the states using GP models is not as accurate as desired.

The disadvantages of GPs for estimating spatio-temporal phenomena are summarized as follows. 1) The mean functions m and kernels needs to be specified, mean functions and kernels are chosen from experience, or from trial and error. 2) Fitting GP models in high dimensional spaces might be difficult, and it may give



(a)



(b)

Figure 5.4: Comparison of state estimation at $t \geq t_0$ using GP model and ROM for 2D heat problem. (a) Comparison of state estimation error and 3σ bounds. (b) Comparison of normalized RMSE.

poor performance. 3) The estimation does not consider the dynamics of the system.

6. SENSOR SCHEDULING FOR SPATIO-TEMPORAL PHENOMENA

6.1 Introduction

The monitoring and prediction of spatio-temporal phenomena, such as pollution concentration, surface temperature, natural disaster, or power grids [67–69, 97] has been studied for decades. A limited number of sensors are placed in the spatial field, and a sensor scheduling problem is formulated to optimize the performance of the estimator.

In this section, we consider the sensor scheduling problem for spatio-temporal phenomena which can be modeled by PDEs. We present a three-step framework to place multiple mobile sensors which can optimize the performance of KF. First, the sensor scheduling problem is formally stated in Section 6.2. An ROM based sensor placement approach is provided in Section 6.3, which restricts the sensors to the most informative locations. In Section 6.4, two sensor scheduling methods are applied to maximize the long-term KF performance. The three-step framework is summarized in Section 6.5. Finally, in Section 6.6, simulation results are presented to demonstrate the efficiency of the proposed approach.

6.2 Problem Formulation

Consider the discrete-time linear time-invariant (LTI) system which is constructed by discretizing a PDE:

$$x_k = Ax_{k-1} + Bu_k + w_k, \tag{6.1}$$

where $x_k \in \Re^N, u_k \in \Re^p$ are the states and inputs at time instant t_k respectively. The process noise w_k is assumed to be Gaussian white noise with zero mean and

covariance Q.

At each time step, we use q sensors to take measurements, and without loss of generality, we assume that all the states are measurable. Denote $S = \{1, 2, \cdots, N\}$ as the set of sensor locations, and $|S| = N$, where $|S|$ denotes the size of the set S. The measurements are given by

$$y_k = C_k x_k + v_k, \tag{6.2}$$

where $C_k \in \Re^{q \times N}$, and v_k is the measurement noise, which is assumed to be Gaussian white noise with zero mean and covariance R.

We make the following assumptions.

Assumption 5 *The system is stable and diagonalizable.*

Assumption 6 *Each sensor can measure one state of the system, i.e., C_k is a (0,1)-matrix with one non-zero element in each row, and $C_k(i,j) = 1$ if the i^{th} sensor is placed to measure state j, $i = 1, \cdots, q, j = 1, \cdots, N$.*

Assumption 7 *The initial state x_0, the process noise w_k and the sensor noise v_k are mutually independent.*

The optimal estimate in the minimum mean square sense is given by a Kalman filter. At time t_k, denote the predicted state estimate as $\hat{x}_{k|k-1}$, and prediction error is

$$e_k = x_k - \hat{x}_{k|k-1}. \tag{6.3}$$

The prediction error covariance $P_k = E[e_k e_k']$ evolves according to the Riccati recur-

sion

$$P_{k+1} = AP_kA' + Q - AP_kC_k'(C_kP_kC_k' + R)^{-1}C_kP_kA', \tag{6.4}$$

with initial condition P_0.

Define the T-step sensor placement sequence $\sigma = \{C_1, \cdots, C_T\}$. Given initial condition P_0, and a sensor placement sequence σ, the prediction error covariances are denoted by the sample path $\chi = \{P_1, \cdots, P_T\}$. Further, define a T-step averaged reward as:

$$J(\chi, \sigma, P_0) = \frac{1}{T}\sum_{i=1}^{T} r(P_i), \tag{6.5}$$

where $r(.)$ is the reward function, for instance, this could be the trace of the error covariance.

The sensor scheduling problem is formulated as follows.

Find a T-step sensor schedule $\sigma^* = \{C_1^*, \cdots, C_T^*\}$ such that

$$\sigma^* = \arg \max_{C_1, \cdots, C_T \in S} J(\chi, C_1, \cdots, C_T). \tag{6.6}$$

For the sensor scheduling problem with $|S| = N$, the brute-force search takes time $O((N^q)^T)$, which is known to be an NP-hard problem. In addition, the dimension N is large due to the discretization of PDEs. Consequently, solving the sensor scheduling problem is not computationally feasible using existing algorithms. Therefore, we would like to find a subset $S^* \subseteq S$ of sensor locations, such that the sensor scheduling problem considered in this work is:

Sensor Scheduling Problem:

$$\sigma_r^* = \arg \max_{C_1,\cdots,C_T \in S^*} J(\chi, C_1, \cdots, C_T), \tag{6.7}$$

where $|S^*| = M$ is a subset of S, M is the number of locations in the reduced subset, and $M \ll N$.

The three-step sensor scheduling framework is summarized as follows.

- Step 1. Construct an ROM (A_r, B_r, C_r) (Section 3).
- Step 2. Find a reduced subset of sensor locations S^* (Section 6.3).
- Step 3. Find a T-step sensor schedule σ_r^* (Section 6.4)
    1. using the Information Space Receding Horizon Control approach (Section 6.4.2),
    2. using the modified Monte Carlo Tree Search approach (Section 6.4.4).

6.3 ROM Based Sensor Placement

State estimation and sensor scheduling of the spatio-temporal models suffer from large computational demands due to the high dimensionality: the state estimation using Kalman filter involves $N \times N$ matrix multiplications at each time step, and T-step sensor scheduling takes time $O((N^q)^T)$, where N is the dimension of discretized PDEs. In Section 2 and Section 3, we construct ROMs using RPOD and RPOD* which address the first problem of large N. To address the second problem, in this section, an optimal sensor placement problem is formulated to find the subset $S^* \subseteq S$ of sensor locations, such that $V(S^*)$ is close to $V(S)$, where $V(.)$ is an optimization criterion.

Various optimization criteria are used for optimal sensor placement problem [98]. For the open loop sensor placement problem, the most frequently used criteria are maximizing the observability gramian [99], spatial H_2 norm [100], and modal observability [67, 101, 102].

Modal observability is a standard optimization criterion for spatially distributed systems. The optimization problem is motivated from the idea that for each mode, the locations which give the best observability should be kept in the sensor location subset S^*. In the following, first we define the modal observability and then relate it with the ROM.

6.3.1 ROM and Modal Observability

Under assumption that the system is stable and diagonalizable, the measure of observability for each eigenmode is defined as follows.

Definition 4 *For an LTI system with output matrix $C \in \Re^{q\times N}$, define a measure matrix $M_o = CV \in \Re^{q\times N}$, where V denotes the right eigenvectors of the system, and $M_o(i,j)$ is the element in the i^{th} row, j^{th} column. Then $0 \leq |M_o(i,j)| \leq 1$ indicates the observability of the eigenmode j from the i^{th} sensor. $|M_o(i,j)| = 0$ implies that the j^{th} mode is not observable from the i^{th} sensor, and $|M_o(i,j)| = 1$ implies the best observability.*

The output y_k at time t_k with zero initial condition is related to the controllable and observable modes as follows.

$$y_k = \sum_{i=1}^{k} CV_{co}\Lambda_{co}^{i-1}U_{co}'(Bu_{k-i} + w_{k-i}) + v_k, \tag{6.8}$$

where $(\Lambda_{co}, V_{co}, U_{co})$ are the controllable and observable modes defined in Definition 1, and we assume that $w_k \ll u_k$.

Therefore, we are interested in construct an ROM which can capture the input-output behavior of the full order system by retaining only the controllable and observable modes in the ROM. From the analysis in Section 2 and Section 3, we see that RPOD and RPOD* algorithms can be used.

Recall that the ROM constructed using RPOD* under Assumption 3 is:

$$\begin{cases} A_r = \Lambda_{co}, \\ B_r = U'_{co}B, \\ C_r = CV_{co}, \end{cases} \tag{6.9}$$

where from Definition 4, $C_r = CV_{co}$ is exactly the measure matrix for all the controllable and observable modes. Thus, the RPOD* ROM is directly related to the modal observability, and the optimization problem is solved as follows.

6.3.2 ROM Based Sensor Placement Optimization Problem

An ROM is constructed following the RPOD* algorithm given in Algorithm 4 with $C = I_{N\times N}$. The ROM $C_r = CV_{co} = V_{co}$ is the measurement matrix for all the controllable and observable modes with N sensors.

The subset of sensor locations S^* is constructed by keeping the locations that give the best observability of each controllable and observable mode, and the optimization problem is to find the local maxima for each mode.

Suppose the ROM has rank l. Define design parameters $\delta > 0$, and $n_l \leq l$, which are chosen a priori, and denote $|.|_m$ is the Euclidean metric. The sensor placement algorithm is given in Algorithm 8.

Remark 12 *Effects of δ and n_l. The subset S^* constructed using Algorithm 8 contains local maxima for first n_l controllable and observable modes. More local maxima*

Algorithm 8 $S^* =$ SP (C_r, n_l, δ)

1. For the j^{th} eigenmode, $j = 1, \cdots n_l, n_l \leq l$, find the local maxima

$$I^* = \{i^*_{(j)} : |C_r(i^*, j)| > |C_r(i, j)|, \forall |i - i^*|_m \leq \delta\}, \tag{6.10}$$

2. $S^* = \{i^*_{(j)}, \forall j\}$.

would be preserved in the subset S^* if we choose a small δ and a large n_l, while the problem size is increased. Hence, there is a trade off between efficiency and accuracy.

Remark 13 *Local maxima V.S. Threshold. In [67], the subset of sensor locations which can maximize the modal observability is constructed as follows. For each eigenmode, locations where the modal observability is above a threshold ϵ should be kept. So the optimization problem becomes:*

For the j^{th} eigenmode, $j = 1, \cdots l$,

$$i^* = \{i^* : |C_r(i^*, j)| > \epsilon\}, \tag{6.11}$$

where ϵ is a design parameter.

The comparison of the solution to optimization problem (6.10) and optimization problem (6.11) is shown in Fig. 6.1 for a simple eigenmode.

The subset S^* constructed using Algorithm 8 includes five local maxima. If the threshold ϵ is chosen as in Fig. 6.1, then the subset S^*_t constructed using threshold algorithm includes the global maximum/minimum and some extra locations. The local maxima could be found if ϵ is small enough, but will result in a larger S^*_t. Therefore, constructing the subset S^* using local maxima is more efficient in this work.

Also, it should be noticed that using the modal observability criterion does not

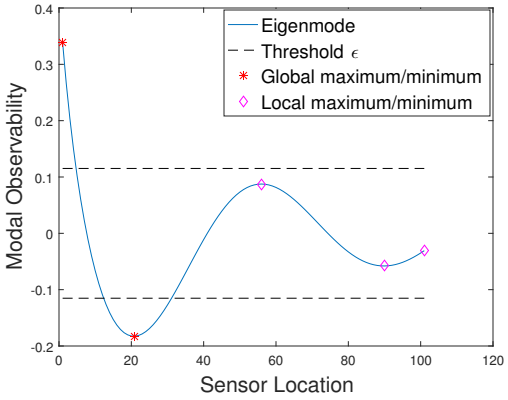


Figure 6.1: This figure illustrates the differences between constructing subset sensor locations S^* using local maxima criterion and threshold ϵ.

guarantee the optimal schedule in S is preserved in the subset S^*. However, in practice, we've seen that the performance of sensor scheduling problem with a small lookahead time period is actually improved by using the subset S^*. In Section 6.6.1, we show an example comparing the sensor scheduling performance using the full set S, and the reduced subset S^*.

6.4 ROM Based Sensor Scheduling

Given the subset S^*, with $|S^*| = M$, in this section, we solve the sensor scheduling problem:

$$\sigma_r^* = \arg \max_{C_1,\cdots,C_T \in S^*} J(\chi, C_1, \cdots, C_T), \tag{6.12}$$

where $J(\chi, C_1, \cdots, C_T) = \frac{1}{T}\sum_{i=1}^{T} r(P_i)$, and $r(.)$ is the reward function.

Several non-myopic sensor scheduling frameworks and optimization techniques have been developed. For example, in [103], suboptimal sliding window and threshold methods were proposed to increase search efficiency in non-myopic sensor scheduling, and in [74], a random selection approach was proposed.

In the following, we start with a discussion on the sensor scheduling problem considered in this work, and then provide two efficient sensor scheduling methods to solve the problem.

6.4.1 Discussion on the Sensor Scheduling Problem

The sensor scheduling problem considered in this work is a non-convex optimization problem, and the non-convexity is from the following two aspects.

First, consider the evolution of prediction error covariance given in (6.4):

$$P_{k+1} = \underbrace{AP_kA' + Q - AP_kC_k'(C_kP_kC_k' + R)^{-1}C_kP_kA'}_{f(P_k,C_k)}. \tag{6.13}$$

Function $f(P_k, C_k)$ has several well-known properties.

It is proved in [74,104] that $f(P_k, C_k)$ is monotone and concave in P_k provided P_k is positive semidefinite and R is positive definite, i.e., for any $P_1, P_2 \geq 0$, $c \in [0, 1]$, and fixed C_k,

$$\begin{aligned} &P_1 \leq P_2 \implies f(P_1, C_k) \leq f(P_2, C_k), \\ &f(cP_1 + (1-c)P_2, C_k) \geq cf(P_1, C_k) + (1-c)f(P_2, C_k). \end{aligned} \tag{6.14}$$

However, $f(P_k, C_k)$ is a nonlinear function of C_k and hence, leads to a non-convex optimization problem.

Also, the reward function $r(.)$ could be a non-convex function.

There are several choices of the reward function $r(.)$. A-optimality minimizes the trace of error covariance, i.e., $r(P_k) = tr(P_k)$, E-optimality minimizes the largest eigenvalue of P_k, $r(P_k) = \lambda_{max}(P_k)$, and D-optimality minimizes the determinant of error covariance, i.e., $r(P_k) = lndet(P_k)$. In this problem, as N increases, the deter-

minant of P_k is large, and the changes in the determinant are also small, similarly, the changes in the trace placing sensors at different locations are small. Hence, in this work, we use E-optimality, or the condition number, as optimization criterion, and change the problem into a maximization problem. The condition number gives insight as to how a perturbation in the input, the initial state vector, perturbs the output, the measurements. It also measures the sensitivity in taking the inverse of the matrix. We maximize $r(P_k) = \frac{\sigma_N}{\sigma_1}$ where σ_1 is the largest singular value of P_k and σ_N is the smallest singular value of P_k. The SVD ratio (condition number) is quasiconvex (i.e., has convex level sets).

In this dissertation, the Information Space Receding Horizon Control (I-RHC) algorithm [105] and the Monte Carlo Tree Search (MCTS) algorithm [106] are utilized to solve the sensor scheduling problem. To further reduce the computations of the MCTS algorithm, we propose a modified MCTS algorithm, which is a combination of I-RHC algorithm and MCTS algorithm. First, we briefly review the I-RHC algorithm and MCTS algorithm.

6.4.2 Preliminaries: Information Space Receding Horizon Control (I-RHC)

The I-RHC algorithm proposed in [105] starts from a stochastic relaxation of the sensor scheduling problem (6.12).

We start from placing one sensor at each time step. Denote the sensor locations as $S^* = \{1, 2, \cdots, M\}$, and the control action at time step k as u_k, where $u_k = j$ implies that at time step k, the sensor is placed at location j. At each time step k, instead of taking a particular control action, I-RHC takes control action $u_k = j$ with probability $\pi_{k,j}$, where $0 \leq \pi_{k,j} \leq 1$, and $\sum_{j=1}^{M} \pi_{k,j} = 1$ for all k.

Define a randomized policy $\Pi = \{\pi_1, \cdots, \pi_T\} \in \Re^{T \times M}$, where π_k is a row vector with element $\pi_{k,j}$, $j = 1, \cdots, M$. Then the T-step total expected reward with

stochastic policy Π could be written as follows:

$$E(J(\chi,\Pi)) = \sum_{u_1=1}^{M} \cdots \sum_{u_T=1}^{M} J(\chi, u_1, \cdots, u_T)\pi_{1,u_1} \cdots \pi_{T,u_T}. \tag{6.15}$$

And hence, the optimization problem we want to solve is:

$$\begin{aligned} \arg\max_{\Pi} \quad & E(J(\chi,\Pi)) \\ \text{subject to} \quad & \sum_{j=1}^{M} \pi_{k,j} = 1, k = 1, \cdots, T, \\ & 0 \le \pi_{k,j} \le 1, k = 1, \cdots, T, j = 1, \cdots, M, \end{aligned}$$

given some initial state χ_0.

A standard approach to solve the optimization problem (6.4.2) is to use gradient descent. The total expected reward can be written as:

$$E(J(\chi,\Pi)) = \sum_{u_1,\cdots,u_T} (\sum_{j=1}^{M} J(\chi, u_1, \cdots, u_t = j, \cdots, u_T)\pi_{t,j})\pi_{1,u_1} \cdots \pi_{T,u_T} \tag{6.16}$$

and hence,

$$\begin{aligned} \frac{\partial E(J(\chi,\Pi))}{\partial \pi_{t,j}} &= \sum_{u_1,\cdots,u_T} J(\chi, u_1, \cdots, u_t = j, \cdots, u_T)\pi_{1,u_1} \cdots \pi_{T,u_T} \\ &= E(J(\chi,\Pi|u_t = j)). \end{aligned} \tag{6.17}$$

From the gradient descent method, the policy Π can be improved by ascending along the gradient $\frac{\partial J(\chi,\Pi)}{\partial \Pi}$, and we can adjust the policy at iteration n as follows:

$$\Pi_{n+1} = \Theta(\Pi_n + \epsilon_n \frac{\partial J(\chi,\Pi)}{\partial \Pi}|_{\Pi=\Pi_n}), \tag{6.18}$$

where ϵ_n is a small step size, and Θ is a projection operator which project the new policy onto the probability space. Evaluating the gradient $\frac{\partial J(\chi,\Pi)}{\partial \Pi}$ exactly would require a large number of simulations, which is not tractable. Hence, a noisy estimate of the gradient using a single sample path is used as follows:

$$\frac{\partial \hat{J}(\chi,\Pi)}{\partial \pi_{t,j}} = \begin{cases} \frac{J(\omega)}{\pi_{t,j}}, & \text{if } u_t(\omega) = j, \\ 0, & \text{o.w,} \end{cases} \tag{6.19}$$

where $\omega = \{\chi_1(\omega), u_1(\omega), \cdots, \chi_T(\omega), u_T(\omega)\}$ denotes a sample sample path. Therefore, given some initial guess Π_0, the optimum of $J(\chi,\Pi)$ with respect to the stochastic policy Π could be found by updating the policy using (6.19).

The I-RHC algorithm proposed in [105] is to solve a T-step stochastic optimization problem at each time step, and apply the control action in a receding horizon fashion. Suppose at time $k = 0$, the initial state is P_0 and initial guess is Π_0. The optimum policy could be found using (6.18) and (6.19), and suppose the policy converge to $\Pi^* = \{\pi_1^*, \cdots, \pi_T^*\}$. Then as a standard procedure in RHC, we apply the control u_1 according to π_1^*, and update the state to P_1. Then at the next time step, the procedure above is repeated.

Now, consider the problem to place q sensors at each time step. We place the sensors sequentially using I-RHC, i.e., we start from placing one sensor, and then repeat the I-RHC procedure q times.

Suppose the initial state error covariance at time $k = 0$ is P_0, the lookahead horizon is H, the step size is ϵ_n and the initial policy is Π_0. The I-RHC Algorithm is summarized in Algorithm 9.

Remark 14 *Computation Complexity Analysis*

Assume that the maximum number of iterations in I-RHC algorithm is N_l. In

Algorithm 9 I-RHC $\sigma_r^* =$ I-RHC (H, ϵ_n)

1. For each time step k, let $\chi_0 = P_k$, $C_k = \emptyset$.
   (a) For $i = 1$ to q, let $n = 0$, with Π_0
   - while Π_n does not converge, do
     - Generate the sample path $\{\chi_0, \cdots, \chi_H\}$, with initial state χ_0, and policy Π_n.
     - Update the policy with Equation (6.18) and (6.19).
   - Output converged H-step policy $\Pi^* = \{\pi_1^*, \cdots, \pi_H^*\}$, add the i^{th} sensor schedule $C_k = \{C_k, C_j\}$ with probability $\pi_{1,j}^*$, $j = 1, \cdots, M$.

   (b) Update $P_{k+1} = f(P_k, C_k)$, go to Step 1.
2. Output optimal sensor schedule $\sigma_r^* = \{C_1, \cdots, C_T\}$.

each iteration, system propagates H times in one sample path, where system propagation takes time $O(l^3)$, l is the size of the ROM. Policy update takes time $O(HM)$. Therefore, the total computation complexity for a T-step sensor scheduling problem is $O(qTHN_l(l^3 + M))$, which linearly increases with number of sensors q, sensor scheduling time T and search space M. Here, $l, M \ll N$, and H is a small lookahead region, N_l could also been tuned by varying step size ϵ_n.

Remark 15 *Discussion on implementation issues.*

In the following, we discuss how to improve the I-RHC performance in practice.

- *Parallelization of I-RHC: The stochastic gradient descent technique based on one sample path is noisy. To reduce the variance of the estimates, we can draw multiple sample paths at each iteration parallelly, and update the policy. Moreover, at each time step k, multiple I-RHC algorithm could be run in parallel. The converged policies might be different, so we can choose the best policy.*
- *Reduce computation time using a stopping criterion. The policy $\pi_k, k = 1, \cdots, H$*

would converge to only a few choices in the first several iterations, but it takes a long time for π_k to converge to a single location. Therefore, we can use a stopping criterion N_s, such that when the number of the non-zero elements in Π is smaller than N_s, we stop the gradient ascent and find the best action using exhaustive search.

Remark 16 *Comparison with stochastic selection algorithm in [74].*

In [74], a stochastic sensor selection algorithm is proposed, which also assigns a probability $\hat{\pi}_i, i = 1, \cdots, M$ as the probability to use the i^{th} sensor, and the optimization is to minimize the steady state prediction error covariance. Therefore, we compare the I-RHC algorithm with the stochastic sensor selection algorithm proposed in [74].

The major differences between the two algorithms are summarized as follows.

First, implementation of two algorithms are different. For I-RHC algorithm, at each time step, the policy converges to one sensor location with probability 1, and hence, the implementation of I-RHC algorithm is deterministic. For stochastic sensor selection algorithm in [74], at each time step, the i^{th} sensor is selected with probability $\hat{\pi}_i, i = 1, \cdots, M$, which results in a random sensor schedule.

Second, the solution using both algorithms is an approximation to the true optimal solution. In [74], the upper bound of the objective function is used for the optimization problem. In I-RHC algorithm, the exact objective function is used, while the gradient is approximated using one sample path.

6.4.3 Preliminaries: Monte Carlo Tree Search (MCTS)

The MCTS is a heuristic algorithm, which expands the search tree based on the random sampling of the search space. Recently, the MCTS has been widely applied in game play, such as the real-time computer Go [107].

There are four basic steps in one search iteration, and until the stopping criterion, typically a limit on time or memory has been reached, MCTS repeats the four steps, and build a search tree iteratively. The best performing root action is returned. The four steps of MCTS are summarized in Fig. 6.3.

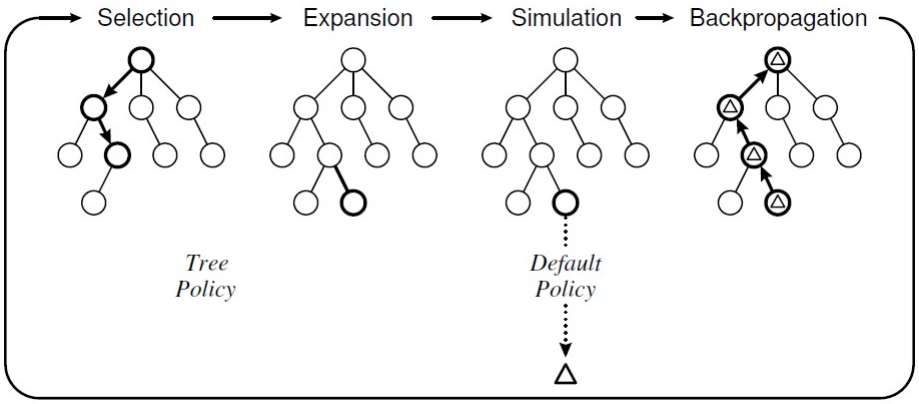


Figure 6.2: Outline of MCTS approach [1]

In the following, we briefly review the MCTS algorithm.

- Selection: Starting from the root node, a child selection policy is recursively applied through the tree until the most urgent expandable node is reached. A node is expandable if it represents a nonterminal state and has unvisited (i.e., unexpanded) children.

  The selection policy balances the exploration and exploitation of the tree search. One commonly used selection policy is the Upper Confidence Bounds for Trees(UCT). A child node i is selected to maximize

$$UCT = \frac{J_i}{n_i} + 2c_p\sqrt{\frac{2lnn_t}{n_i}}, \tag{6.20}$$

where J_i is the total reward after the i^{th} move, and n_i is the number of times child node i has been visited, and hence, $\frac{J_i}{n_i}$ denotes the estimated reward of node j. n_t is the number of times that the current node's parent has been visited, and $c_p > 0$ is a constant, which is the exploration parameter.

- Expansion: one or more child nodes are added to expand the tree, according to the available actions.

- Simulation: a simulation is run from the new nodes according to the default policy to produce an outcome. The default simulation policy is to apply the possible actions randomly until the end of the game, which is also called a random rollout.

- Backpropagation: the simulation result is backpropagated through the selected nodes to update their statistics. In the backpropagation step, the selected nodes are updated by adding the reward of the simulation to the current node, and increment the number of visited times to the current node.

The advantages of using MCTS is that as shown in [106], the MCTS using UCT converges to the minimax tree and is thus optimal, and it can be implemented in real-time.

6.4.3.1 Direct Application of MCTS

The direct application of MCTS has the following steps. At each time instant k, we solve an H-step optimization problem by constructing a Monte Carlo Tree with depth H, and select the first move in a receding horizon fashion. For system with multiple sensors, we select the sensors sequentially. However, there are some implementation issues with the direct application of MCTS.

First, for a 19×19 board Go game, at each game state, a player is faced with a choice of about 250 possible moves and a typical game in Go might last for 150 moves. For a sensor scheduling problem, for example, to place one sensor in a 3D field, the possible choices at each state could be 10^6, and the process could last for thousands steps. Hence, it is a more complex problem than Go.

Second, the performance of MCTS is highly dependent on the rollout policy. A carefully selected rollout policy rather than the uniform policy can significant improve the performance.

Third, although there are several parallelization approaches, in general, MCTS is not easy to implement in parallel, and the performance of the parallel MCTS is typically worse than running serial MCTS for equivalent number of simulations.

The main parallelization approaches for MCTS are shown in Figure 6.3, as described by [2].

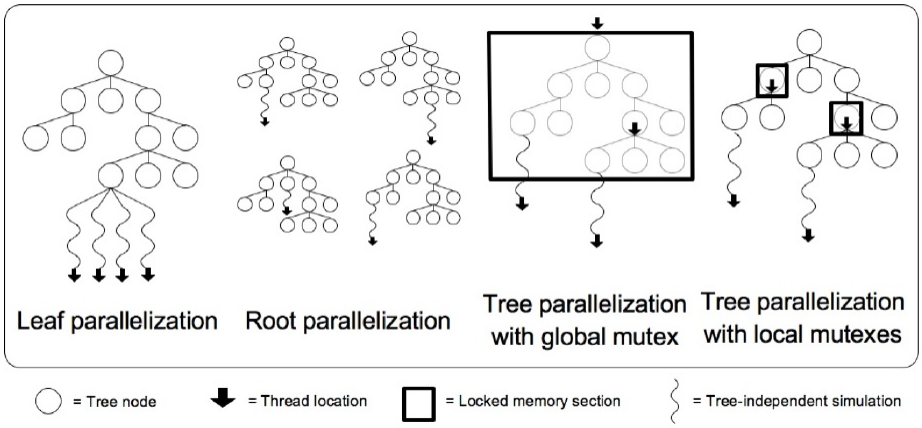


Figure 6.3: Parallelization approaches for MCTS [2]

6.4.4 Modified MCTS

To reduce the computational cost and improve the performance of MCTS algorithm, we propose a modified MCTS algorithm which combines the I-RHC and MCTS algorithm.

Using the I-RHC method in Section 6.4.2, we can see that for a tree with depth H, the probability that any node at level k takes action j is $\pi_{k,j}$. Furthermore, in each round of MCTS, one sample path is generated, and hence, can be used to update the policy gradient. Therefore, for the modified MCTS algorithm, instead of the naive uniform policy, we perform the rollout with the I-RHC policy.

We start with an empty tree, and an initial policy $\Pi \in \Re^{H \times M}$, with a lookahead horizon H. Following the same procedure as MCTS, we select the best child node to expand using UCT. In expansion step, the available actions is chosen using policy Π, and as Π converges, the number of available actions is reduced. In simulation step, we rollout with policy Π. If the current level is k, then we rollout $H - k$ steps. In the backpropagation step, the tree and the policy are updated as in standard MCTS. The modified MCTS algorithm is summarized in Algorithm 10.

The major improvement is that in modified MCTS, we do not rollout randomly. We rollout with policy Π which can provide a better approximation of the cost-to-go than a naive random rollout. Also, the computational cost is reduced by expanding the possible actions with Π. As Π converges, the number of branches is reduced.

6.5 Three-Step Sensor Scheduling Framework

Now, we can solve the sensor scheduling problem using a three-step approach summarized in Algorithm 11.

Comparison of I-RHC and modified MCTS. The difference between I-RHC and modified MCTS is how to choose the best node to explore. For I-RHC algo-

Algorithm 10 σ_r^* = Modified MCTS (H, c_p)

1. For each time step k, let $\chi_0 = P_k$, $C_k = \emptyset$.
   (a) For $i = 1$ to q, let $n = 0, d = 0$, with Π_0.
   - While $d \leq H$
     - Search tree with selection policy (6.20) until node i at level d.
     - Expand the j^{th} child of i with probability $\pi_{d,j}$.
     - Rollout $H - d$ steps with probability Π.
     - Backpropagation.
     - Update $\Pi_{n+1} = \Pi_n + \frac{\partial J}{\partial \Pi}$ as I-RHC.
   - Search tree from root with tree search policy and $c_p = 0$ for the first action $C_i = j$. Add $C_k = \{C_k, C_i\}$.

   (b) Update $P_{k+1} = f(P_k, C_k)$, go to Step 1.
2. Output optimal sensor schedule $\sigma_r^* = \{C_1, \cdots, C_T\}$.

Algorithm 11 Sensor Scheduling Algorithm

1. Choose l and construct ROM using RPOD* algorithm: (A_r, B_r, C_r) = RPOD* (A, B, C, l).
2. Choose δ, n_l and construct subset: $S^* =$ SP (C_r, n_l, δ).
3. Choose H, β, ϵ_n and find the optimal schedule σ_r^* = I-RHC(H, ϵ_n),
   or choose H, c_p, n_h and find the optimal schedule σ_r^* = Modified MCTS (H, c_p).

rithm, in each iteration, the best sample path is chosen simultaneously according to the policy Π. For modified MCTS, in each iteration, the best node to be explored is chosen with UCT criterion, which is equivalent to choose the node with the largest expected reward with a variance after generating several sample paths. As the number of iterations increases, the expected reward is estimated more accurately, and hence, modified MCTS is more accurate.

However, modified MCTS expands a large search tree, which in the worst case scenario, has M^H brunches in one simulation, and hence, the computational complexity is $O(qTHM^H(l^3+M))$, while the computational complexity of I-RHC algorithm as analyzed before, is $O(qTHN_l(l^3+M))$, which is order of M. Moreover, I-RHC algorithm is easy to implement in parallel.

Therefore, for a suitable sized problem, for example, $M = 10$, using stochastic MCTS performs better and converges faster. For a large scale problem, for example, $M \geq 100$, the I-RHC algorithm should be used.

6.6 Computational Results

In this section, first, we show a one-dimensional heat example to compare the sensor scheduling performance using the full set S and the reduced subset S^*. Then we show the simulation results of placing three moving sensors for atmospheric dispersion problem. We start from a toy problem, which aims to compare the performances of the I-RHC algorithm and Modified MCTS algorithm with the optimal schedule for a lookahead horizon $H = 5$. Then we solve the sensor scheduling problem for a large scale three-dimensional system.

6.6.1 Comparison of the sensor scheduling performance using the full set and the reduced subset

Consider the one-dimensional heat example shown in (5.7).

The full order system has 100 DOFs, the ROM is constructed using RPOD*, and has 20 DOFs. The subset S^* is constructed by keeping the peaks of first n_l eigenmodes, $n_l = 1, \cdots, 20$.

At each time step, we place one mobile sensor in the field, and the optimal sensor scheduling problem is solved using exhaustive search in receding horizon fashion. At each time step, we look ahead for 3 steps, and find the best action using exhaustive search. Therefore, at each time step, we perform we perform $|S|^3$ and $|S^*|^3$ simulations to find the optimal action using the full set S and reduced subset S^*, respectively.

In Fig. 6.4, we compare the optimal rewards for placing one mobile sensor using S^* and S. Since the performance of the sensor scheduling on the subset S^* depends on the design parameters n_l and δ. Therefore, the optimal reward is plotted as a function of n_l and δ. From the simulation results, we can see that when $n_l \geq 6$, $\delta \leq 50$, $J(\chi, \sigma_r^*, P_0) \geq (\chi, \sigma^*, P_0)$, i.e., the performance of sensor scheduling using subset S^* is better than using full set S. This is because the sensor scheduling problem is solved using a small lookahead region, the best action at time step k using full set S is not guaranteed to be the optimal action, and hence, restricting the choices to the most controllable and observable locations might eliminate the bad locations, and give a better performance. In this simulation, the best performance is achieved when $n_l = 10, \delta = 30$, and the size of the resulting subset S^* is 38.

6.6.2 Sensor scheduling for 2D atmospheric dispersion problem

We start from a small scale 2D atmospheric dispersion example, where the exhaustive search is computationally tractable.

The evolution of the air pollutant concentration is described in (3.74), and we assume that $\frac{\partial c}{\partial z} = 0, \frac{\partial^2 c}{\partial z^2} = 0$.

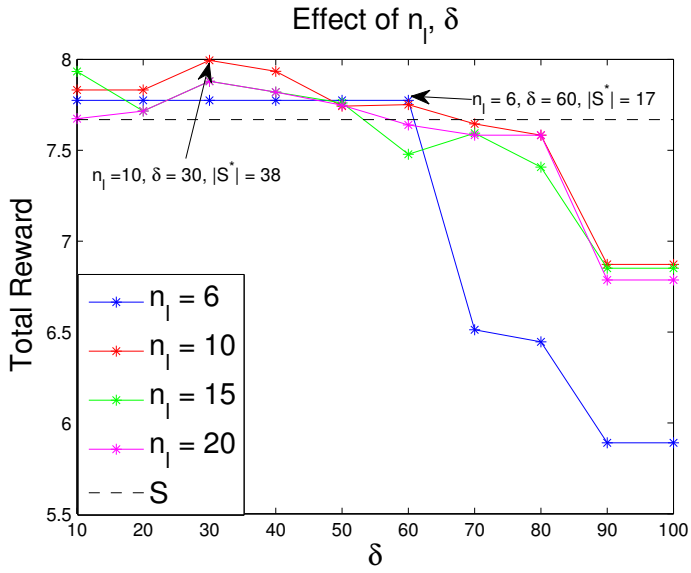


Figure 6.4: Comparison of optimal reward using S^* and S for 1D heat problem. The optimal reward using S^* is plotted as a function of design parameters n_l and δ.

The system is discretized using finite difference method, and there are 10×10 grids which are equally spaced, so the size of the full order system is 100. The ROM (A_r, B_r, C_r) is constructed using RPOD* algorithm, and the size of ROM is $l = 10$.

The subset S^* is constructed by finding the local maxima of each column of C_r, and $|S^*| = 17$. The simulation is performed for 100 time steps. At each time step, we lookahead for $H = 5$ steps. We perform five independent simulations using I-RHC algorithm and modified MCTS algorithms.

In Fig. 6.5, we compare the total rewards using exhaustive search, I-RHC, modified MCTS and myopic algorithms. The averaged computation time is also shown in the figure. It can be seen that myopic algorithm is the fastest, and exhaustive search is the slowest. The time reduction using modified MCTS is 98.5%, and using I-RHC is 96% compared with exhaustive search. The performance improvement using modified MCTS and I-RHC is 23% compared with myopic algorithm.

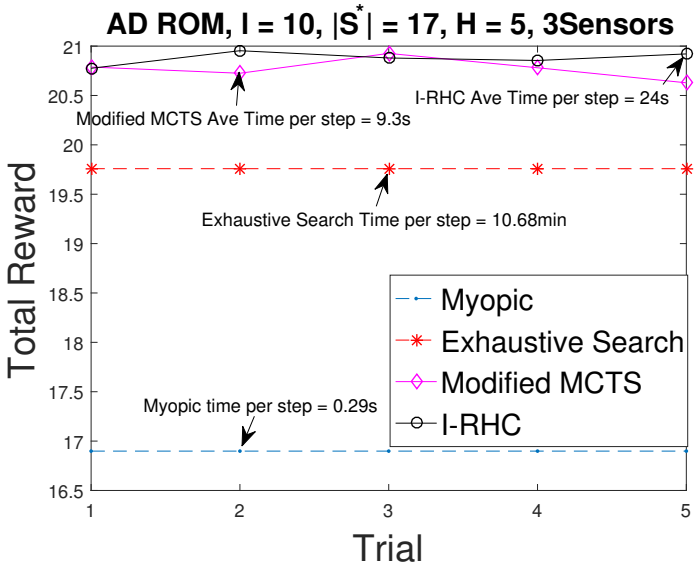


Figure 6.5: Comparison of I-RHC, modified MCTS, exhaustive search and myopic approaches for 2D atmospheric dispersion problem.

The comparison of state estimation error and 3σ bounds are shown in Fig. 6.6.

We can see that the state estimation errors using three approaches are close, and all within the 3σ bounds.

6.6.3 Sensor scheduling for 3D atmospheric dispersion problem

In the following, we place three mobile sensors for the 3D atmospheric dispersion problem introduced in (3.74). The dimension of the system after discretization is 10^5. The ROM is constructed using RPOD* algorithm, and the order of reduced model is 30.

We start from $|S| = N = 10^5$, and reduce the feasible sensor locations to the local maxima of the 30 modes, such that the subset $|S^*| = 309$. We perform the simulation for 100 time steps, i.e., $T = 100$. At each time step, we lookahead $H = 5$ steps. In this example, the exhaustive search and modified MCTS are not feasible.

In Fig. 6.7, we compare the total rewards using I-RHC algorithm with myopic

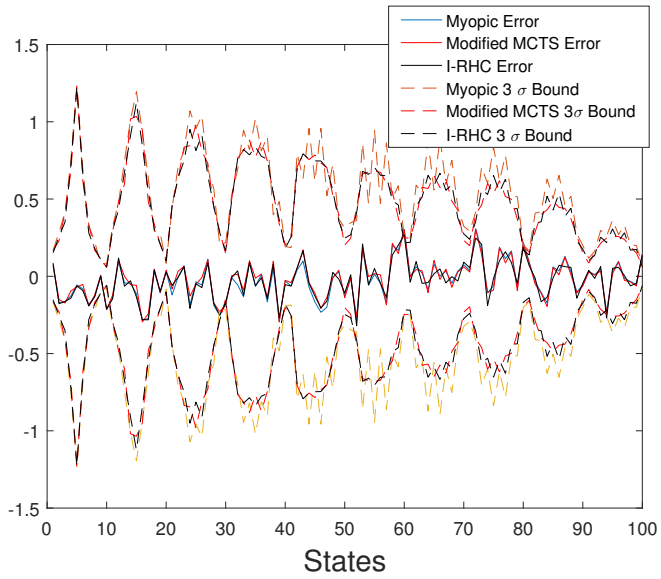


Figure 6.6: State estimation using I-RHC, modified MCTS and myopic approaches.

algorithm. We run the simulation five times using I-RHC algorithm.

The average performance improvement using I-RHC is about 7% compared with myopic algorithm. In this example, using modified MCTS approach takes about twice time than the I-RHC approach, which is not efficient. Therefore, the modified MCTS approach is not utilized for this example.

6.7 Summary

In this section, we propose a three-step sensor scheduling framework, which can place multiple mobile sensors to monitor spatio-temporal phenomena. First, we construct an ROM using RPOD* algorithm, wherein the ROM is directly related to the modal observability. Next, we reduce the feasible sensor locations by solving an optimization problem that maximizes the observability of the modes of the ROM. The I-RHC algorithm, and a modified MCTS algorithm are proposed for the resulting sensor scheduling problem. Both algorithms are applied in the receding horizon

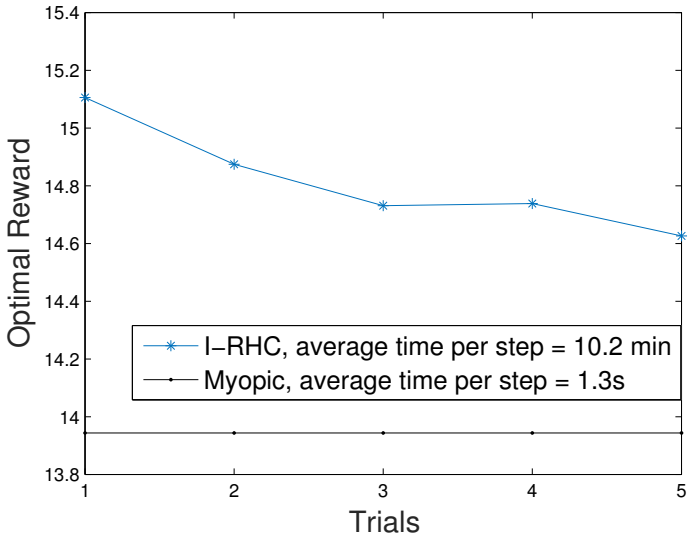


Figure 6.7: Comparison of I-RHC and myopic approaches for 3D atmospheric dispersion problem.

fashion, and are used to optimize the long-term average reward.

From the simulation results shown in this section, we see that for a moderate size system, the I-RHC algorithm and modified MCTS algorithm can solve the sensor scheduling problem fast, and the performance improvement using I-RHC algorithm and modified MCTS algorithms is much larger than myopic algorithm, while the computational time saved is more than 90% compared with exhaustive search. As is evident from the results, the implementation of I-RHC and MCTS is not efficient enough for large scale problems, and has to be explored in the future.

7. CONCLUSION AND FUTURE WORK

This dissertation tackles three problems associated with the state estimation of spatio-temporal phenomena. First, for the spatio-temporal phenomena, which can be modeled by PDEs, the dimension of the system is large due to the discretization of the spatial field. Therefore, state estimation using KF is not computationally tractable. Also, in general, exact modeling of the spatio-temporal phenomena is difficult. Therefore, the unmodeled dynamics, nonlinear terms and external disturbances can be treated as unknown inputs. The state estimation with unknown inputs has been an active research area. Furthermore, the performance of the KF could be improved by placing sensors at feasible locations.

In Section 2 and Section 3, we present two model reduction algorithms: RPOD and RPOD* algorithm, which construct the ROMs for state estimation. In Section 4, we propose an AR model based unknown input filtering algorithm, which can be used to estimate the stochastic unknown inputs using measurements. Section 5 illustrates the issues of using GP for estimating the spatio-temporal phenomena. In Section 6, we present a three-step sensor scheduling framework to place multiple mobile sensors to monitor the spatio-temporal phenomena. The conclusions and proposed future work are discussed as follows.

Contributions on Model Reduction: The RPOD and RPOD* algorithms proposed in this work construct ROMs, which can capture the input-output behavior of the full order systems and are used for estimating the states of the original system. It has been demonstrated that state estimation using the ROMs constructed by both algorithms are accurate enough. The major contribution of the proposed model reduction algorithms is that the computational cost and the storage requirement

are dramatically reduced compared with BPOD/output projection algorithm. More importantly, the RPOD* algorithm could be implemented in real-time, which we believe, is the most computationally efficient algorithm. Moreover, we provide a new perspective to understand the relationship between the dynamical system and its ROM. The RPOD* algorithm is easy to implement, and the implementation issues are discussed in details.

Contributions on Unknown Input Filtering: The AR model based unknown input realization approach proposed in this work could be used when the unknown inputs are wide sense stationary with rational power spectrum. The algorithm is based on standard system identification techniques, and is more accurate than OT-SKF and UMV algorithms, and can tolerate more sensor noise. Moreover, the "observer matching" condition needs not to be satisfied, and the proposed approach can be generalized to estimate the locations of the unknown inputs as well.

Contributions on Sensor Scheduling: In Section 6, we propose a three-step framework to solve the sensor scheduling problem. As discussed in Section 1, to monitor the spatio-temporal phenomena, most of the existing algorithms focus on placing stationary sensors, or place mobile sensors that can optimize the performance at the current time instant, using myopic actions. On the other hand, non-myopic algorithms outperform myopic algorithms drastically. Therefore, the major contribution of the proposed framework is that two non-myopic sensor scheduling algorithms are proposed for the problem. The ROM constructed via RPOD* algorithm could be used directly for finding the feasible subset of sensor locations. The simulation results demonstrate that at least for a moderate sized problem, the proposed framework can be used to solve the sensor scheduling problem efficiently, while significantly improving performance over a myopic policy.

Future Work on Gaussian Process: The GP has been widely used for state

estimation when the dynamical system is unknown. As discussed in Section 5, the state estimation using a spatial GP model is not accurate, and a lot of current research focuses on developing suitable spatio-temporal GP models. Further study in constructing spatio-temporal GP model has to be done, and a careful comparison between state estimation using spatio-temporal GP model and that using physics based ROMs may be of considerable interest.

Future Work on Model Reduction: The RPOD and RPOD* algorithms proposed in this dissertation are limited to linear stable systems. However, more general applications of RPOD* could be studied in the future work. The RPOD* algorithm can be implemented in real-time, even for a large scale system with field measurements. Therefore, the applications of RPOD* for time-varying system is of interest. Moreover, the dynamic mode decomposition (DMD) model reduction approach has attracted lot of attention recently, and a comparison between RPOD* and DMD might help us improve the RPOD* algorithm as well.

Future Work on Sensor Scheduling: In this dissertation, we have applied the I-RHC algorithm and modified MCTS algorithm for solving the sensor scheduling problem for large scale systems. However, the performance of I-RHC and modified MCTS does not improve significantly compared with a myopic algorithm in large scale systems due to implementation issues. Hence, in future work, improving the performance of I-RHC algorithm and MCTS algorithm has to be studied for large-scale systems, in particular, via parallel implementation of the I-RHC and MCTS methods.

REFERENCES

[1] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck, "Monte-Carlo Tree Search: A New Framework for Game AI," in *Proceeding of Artificial Intelligence and Interactive Digital Entertainment Conference*, 2008, pp. 216–217.

[2] G. M.J-B.Chaslot, M. H. M. Winands, and H. J. van den Herrik, "Parallel Monte-Carlo Tree Search," in *Proceeding of Computers and Games*, 2008, pp. 60–71.

[3] C. Yoo, J. B. Valdes, and G. R. North, "Stochastic Modeling of Multidimensional Precipitation Fields Considering Spectral Structure," *Water Resources Research*, vol. 32(7), pp. 2175–2187, 1996.

[4] J. C. McWilliams, "Modeling the Oceanic General Circulation," *Annual review of fluid mechanics*, vol. 28, pp. 215–248, 1986.

[5] G. Evensen, "Inverse methods and data assimilation in nonlinear ocean models," *Physica D*, vol. 77, pp. 108–129, 1994.

[6] J. M. Stockie, "The Mathematics of Atmospheric Dispersion Modeling," *SIAM Review*, vol. 53, No.2, pp. 349–372, 2011.

[7] M. Ersoy, "Dimension Reduction for Compressible Pipe Flows Including Friction," *Asymptotic Analysis, IOS Press*, vol. 98(3), pp. 237–255, 2016.

[8] R. Vasquez and M. Krstic, *Control of Turbulent and Magnetohydrodynamic Channel Flows, Systems and Control: Foundations and Applications.* Boston, MA: Birkhauser, 2007.

[9] A. Smyshlyaev, M. Kristic, N. Chaturvedi, J. Ahmed, and A. Kojic, "PDE Model for Thermal Dynamics of a Large Li-Ion Battery Pack," in *Proceeding of 2011 American Control Conference*, 2011, pp. 959–964.

[10] B. Perthame, *Parabolic Equations in Biology: Growth, reaction, movement and diffusion.* Springer, 2015.

[11] Y.-F. Huang, S. Werner, J. Huang, N. Kashyap, and V. Gupta, "State Estimation in Electric Power Grids," *IEEE Signal Processing Magazine*, pp. 33–43, 2012.

[12] G. Evensen, "The Ensemble Kalman Filter: theoretical formulation and practical implementation," *Ocean Dynamics*, vol. 53, pp. 343–367, 2003.

[13] J. A. Burns, E. Cliff, and C. Rautenberg, "A distributed parameter control approach to optimal filtering and smoothing with mobile sensor networks," in *Proceeding of Mediterrenean Control Conference*, 2009.

[14] H. T. Banks and K. Kunisch, *Estimation Techniques for Distributed Parameter Systems, Systems and Control: Foundations and applications.* Boston, MA: Birkhauser, 1989.

[15] M. Krstic and A. Smyshlaev, *Boundary Controls of PDEs, Advances in Design and Control.* Philadelphia, PA: SIAM, 2008.

[16] M.Jovanovic and B. Bamieh, "Modeling flow statistics using the linearized navier-stokes equations," in *Proceedings of the 40th IEEE conference on decision and control*, 2001, pp. 4944–4949.

[17] J. Hepffner, M. Chevalier, T. R. Bewley, and D. S. Henningson, "State estimation in wall-bounded flow systems. Part 1. Perturbed laminar flows," *Journal of Fluid Mechanics*, vol. 534, pp. 263–294, 2005.

[18] R. K. Mehra, "On the Identification of Variances and Adaptive Kalman Filtering," *IEEE Transactions on Automatic Control*, vol. 15, no. 2, pp. 175–184, 1970.

[19] J. Dunik and M. Simandl, "Estimation of state and measurement noise covariance matrices by multi-step prediction," in *Proceedings of the 17th IFAC World Congress*, 2008, pp. 3689–3694.

[20] D. D. Arianandа and G. Leus, "Compressive Wideband Power Spectrum Estimation," *IEEE Transactions on signal processing*, vol. 60, no. 9, pp. 4775–4789, 2012.

[21] D. Ucinski, *Optimal Measurement Methods for Distributed Parameter System Identification*. CRC Press, 2004.

[22] G. Berkooz, P. Holmes, and J. L. Lumley, "The proper orthogonal decomposition in the analysis of turbulent flows," *Annual review of fluid mechanics*, vol. 25, pp. 539–575, 1993.

[23] K. C. Hall, J. P. Thomas, and E. H. Dowell, "Proper Orthogonal Decomposition Technique for Transonic Unsteady Aerodynamic Flows," *AIAA Journal*, vol. 38, no. 10, pp. 1853–1862, 2000.

[24] L. Sirovich, "Turbulence and the dynamics of coherent structures. Part 1: Coherent Structures," *Quarterly of Applied Mathematics*, vol. 45, pp. 561–571, 1987.

[25] M. Rathinam and L. R. Petzold, "A new look at proper orthogonal decomposition," *SIAM Journal on Numerical Analysis*, vol. 41, no. 5, pp. 1893–1925, 2003.

[26] T. Smith, J. Moehlis, and P. Holmes, "Low-dimensional models for turbulent plane couette flow in a minimal flow unit," *Journal of Fluid Mechanics*, vol. 538, pp. 71–110, 2005.

[27] S. G. Siegel *et al.*, "Low-dimensional modeling of a transient cylinder wake using double proper orthogonal decomposition," *Journal of Fluid Mechanics*, vol. 610, pp. 1–42, 2008.

[28] B. C. Moore, "Principal Component Analysis in Linear Systems: Controllability, Observability and Model Reduction," *IEEE Transactions on Automatic Control*, vol. 26, No.1, pp. 17–32, 1981.

[29] K. Willcox and J. Peraire, "Balanced Model Reduction via the Proper Orthogonal Decomposition," *AIAA Journal*, vol. 40, pp. 2323–2330, 2002.

[30] C. W. Rowley, "Model Reduction for Fluids using Balanced Proper Orthogonal Decomposition," *International Journal of Bifurcation and Chaos*, vol. 15, pp. 997–1013, 2005.

[31] S. Kung, "A New Identification method and Model Reduction Algorithm Via Singular Value Decomposition," *12th Asilomar Conference on Circuits, Systems and Computers*, pp. 705–714, Nov. 1978.

[32] J.-N. Juang and R. S. Pappa, "An Eigensystem Realization Algorithm for Model Parameter Identification and Model Reduction," *Journal of Guidance, Control, and Dynamics*, vol. 8, No.5, pp. 620–627, 1985.

[33] Z. Ma, S. Ahuja, and C. W. Rowley, "Reduced Order Models for control of fluids using the eigensystem realization algorithm," *Theoretical and Computational Fluid Dynamics*, vol. 25, pp. 233–247, 2011.

[34] P. J. Schmid, "Dynamic Mode Decomposition of Numerical and Experimental Data," *Journal of Fluid Mechanics*, vol. 656, pp. 5–28, 2010.

[35] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, "On dynamic mode decomposition: Theory and applications," *Journal of Computational Dynamics*, vol. 1, Issue 2, pp. 391–421, 2014.

[36] J. H. Tu and C. W. Rowley, "An improved algorithms for balanced POD through an analytic treatment of impulse response tails," *Journal of Computational Physics*, vol. 231, pp. 5317–5333, June, 2012.

[37] N. Halko, P. Martinsson, and J. Tropp, "Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 52(2), pp. 217–288, 2011.

[38] P. Drineas *et al.*, "Fast Monte Carlo Algorithms for Matrices II: Computing a low rank approximation to a matrix," *SIAM Journal on Computing*, vol. 36, pp. 158–183, 2006.

[39] M. W. Mahnoey, "Randomized algorithms for matrices and data," *Foundations and Trends in Machine Learning*, vol. 3(2), pp. 123–224, 2011.

[40] G. Calafiore and M. Campi, "The Scenario Approach to Robust Control Design," *IEEE Transactions on Automatic Control*, vol. 51, pp. 742–753, 2006.

[41] M. C. Campi, S. Garatti, and M. Prandini, "The Scenario Approach for Systems and Control Design," *Annual Reviews in Control*, vol. 33, pp. 149–157, 2009.

[42] S.-H. Wang, E.J.Davison, and P. Dorato, "Observing the states of systems with unmeasurable disturbances," *IEEE Transactions on Automatic Control*, vol. 20, no. 5, pp. 716–717, 1975.

[43] S. Bhattacharyya, "Observer design for linear systems with unknown inputs," *IEEE Transactions on Automatic Control*, vol. AC-23, no. 3, pp. 483–484, 1978.

[44] P. Kudva, N.Viswanadham, and A. Ramakrishna, "Observers for linear systems with unknown inputs," *IEEE Transactions on Automatic Control*, vol. 25, no. 1, pp. 113–115, 1980.

[45] M. Hou and P. Muller, "Design of observers for linear systems with unknown inputs," *IEEE Transactions on Automatic Control*, vol. 37, no. 6, pp. 871–875, 1992.

[46] S. Hui and S. H. Zak, "Low-order state estimators and compensators for dynamical systems with unknown inputs," *Systems & Control Letters*, vol. 21, No.6, pp. 493–502, 1993.

[47] M. Darouach, M. Zasadzinski, and S. Xu, "Full-order observers for linear systems with unknown inputs," *IEEE Transactions on Automatic Control*, vol. 39, no. 3, pp. 606–609, 1994.

[48] S. K. Spurgeon, "Sliding mode observers: a survey," *International Journal of Systems Science*, vol. 39, no. 8, pp. 751–764, 2008.

[49] K. Kalsi, J. Lian, S. Hui, and S. H. Zak, "Sliding-mode observers for systems with unknown inputs: A high-gain approach," *Automatica*, vol. 46, Issue 2, pp. 347–353, 2010.

[50] M. Darouach, M. Zasadzinski, A. B. Onana, and S. Nowakowski, "Kalman filtering with unknown inputs via optimal state estimation of singular systems," *International Journal of Systems Science*, vol. 26(10), pp. 2015–2028, 1995.

[51] M. Hou and R. J. Patton, "Optimal Filtering for Systems with Unknown Inputs," *IEEE Transactions on Automatic Control*, vol. 43, no. 3, pp. 445–449,

1998.

[52] D. Koenig and S. Mammar, "Reduced order unknown input kalman filter: application for vehicle lateral control," in *Proceedings of American Control Conference*, 2003, pp. 4353–4358.

[53] C.-S. Hsieh, "A Unified Framework for State Estimation of Nonlinear Stochastic Systems with Unknown Inputs," in *Proceedings of 9th IEEE Asian Control Conference*, 2013.

[54] C.-S. Hsieh and F.-C. Chen, "Optimal Solution of the Two-Stage Kalman Estimator," *IEEE Transactions on Automatic Control*, vol. 44, pp. 194–199, 1999.

[55] S. Kanev and M. Verhaegen, "Two-Stage Kalman Filtering via Structured Square-Root," *Communications in Information and Systems*, vol. 5, no. 2, pp. 143–168, 2005.

[56] F. B. Hmida, K. Khemiri, J. Ragot, and M. Gossa, "Three-stage Kalman filter for state and fault estimation of linear stochastic systems with unknown inputs," *Journal of the Franklin Institute*, vol. 349, pp. 2369–2388, 2012.

[57] S. Gillijns and B. D. Moor, "Unbiased minimum-variance input and state estimation for linear discrete-time systems," *Automatica*, vol. 43, pp. 111–116, 2007.

[58] C.-S. Hsieh, "Extension of unbiased minimum-variance input and state estimation for systems with unknown inputs," *Automatica*, vol. 45, pp. 2149–2153, 2009.

[59] A. Krause, A. Singh, and C. Guestrin, "Near-Optimal Sensor Placements in Gaussian Processes: Theory, Effects Algorithms and Empirical Studies," *Jour-*

nal of Machine Learning Research, vol. 9, pp. 235 – 284, 2008.

[60] Y. Xu, J. Choi, and S. Oh, "Mobile Sensor Network Navigation using Gaussian Processes with Truncated Observations," *IEEE Transactions on Robotics*, vol. 27(6), pp. 1118–1131, 2011.

[61] N. Stubbs and S. Park, "Optimal Sensor Placement for Mode Shapes via Shannon's Sampling Theorem," *Microcomputers in Civil Engineering*, vol. 11, pp. 411–419, 1996.

[62] M. Papadopoulos and E. Garcia, "Sensor Placement Methodologies for Dynamic Testing," *AIAA Journal*, vol. 36(2), pp. 256–263, 1998.

[63] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[64] Y. Xu and J. Choi, "Adaptive Sampling for Learning Gaussian Processes Using Mobile Sensor Networks," *Sensors*, vol. 11(3), pp. 3051–3066, 2011.

[65] V. Roy, A. Simonetto, and G. Leus, "Spatio-Temporal Sensor Management for Environmental Field Estimation," *Signal Processing*, vol. 128, pp. 369–381, 2016.

[66] S. Garg and N. Ayanian, "Persistent Monitoring of Stochastic Spatio-temporal Phenomena with a Small Team of Robots," in *Proceeding of Robotics: Science and Systems*, 2014.

[67] P. Wolf, S. Moura, and M. Kristic, "On optimizing Sensor Placement for Spatio-Temporal Temperature Estimation in Large Battery Packs," in *51st IEEE Conference on Decision and Control*, 2012, pp. 973–978.

[68] B.Yildirim, C. Chryssostomidis, and G. Karniadakis, "Efficient Sensor Placement for Ocean Measurements Using Low-dimensional Concepts," *Ocean Modeling*, vol. 27, pp. 160–173, 2009.

[69] Z. Zhang and X. Y. ang Guang Lin, "POD-Based Constrained Sensor Placement and Field Reconstruction from Noisy Wind Measurements: A Perturbation Study," *Mathematics*, vol. 4(2), 26, 2016.

[70] P. Mokhasi and D. Rempfer, "Optimized sensor placement for urban flow measurement," *Physics of Fluids*, vol. 16, pp. 1758–1764, 2004.

[71] M. P. Vitus, W. Zhang, A. Abate, J. Hu, and C. J. Tomlin, "On efficient sensor scheduling for linear dynamical systems," *Automatica*, vol. 48, pp. 2482 – 2493, 2012.

[72] S. T. Jawaid and S. L. Smith, "A Complete Algorithm for the Infinite Horizon Sensor Scheduling Problem," in *Proceedings of American Control Conference*, 2014, pp. 437–442.

[73] S. Joshi and S. Boyd, "Sensor Selection via Convex Optimization," *IEEE Transactions on Signal Processing*, vol. 57(2), pp. 451 – 462, February, 2009.

[74] V. Gupta, T. H. Chung, B. Hassibi, and R. M.Murray, "On a Stochastic Sensor Selection Algorithm with Applications in Sensor Scheduling and Sensor Coverage," *Automatica*, vol. 42(2), pp. 251–260, 2006.

[75] D. Yu and S. Chakravorty, "A Randomized Iterative Proper Orthogonal Decomposition Technique with Application to Filtering of PDEs," in *Proceedings of American Control Conference*, 2012, pp. 4363–4368.

[76] ——, "An Iterative Proper Orthogonal Decomposition (I-POD) Technique with Application to the Filtering of Partial Differential Equations," *Journal of the*

Astronautical Sciences, vol. 60(3-4), pp. 468–493, 2013.

[77] ——, "A Randomized Proper Orthogonal Decomposition Technique," in *Proceedings of American Control Conference*, 2015, pp. 1137–1142.

[78] ——, "A Computationally Optimal Randomized Proper Orthogonal Decomposition Technique," in *Proceedings of American Control Conference*, 2016, pp. 3310 – 3315.

[79] ——, "An Autoregressive (AR) Model Based Stochastic Unknown Input Realization and Filtering Technique," in *Proceedings of American Control Conference*, 2015, pp. 1499–1504.

[80] ——, "A stochastic unknown input realization and filtering technique," *Automatica*, vol. 63, pp. 26–33, 2016.

[81] R. Pinnau, *Model Order Reduction: Theory, Research Aspects And Applications, Chapter 5.* Springer, 2008.

[82] S. Lall, P. Krysl, and J. E. Marsden, "Structure-preserving Model Reduction for Mechanical Systems," *Physica D*, vol. 184(1-4), pp. 30–318, 2003.

[83] L. Sirovich, "Turbulence and the Dynamics of Coherent Structures. Part 1: Coherent Structures," *Quarterly of Applied Mathematics*, vol. 45(3), pp. 561–571, 1987.

[84] B. Moore, "Principal Component Analysis in Linear Systems: Controllability, Observability, and Model Reduction," *IEEE Transactions on Automatic Control*, vol. 26(1), pp. 17–32, 1981.

[85] S. Lall, J. Marsden, and S. Glavaski, "Empirical Model Reduction of Controlled Nonlinear Systems," in *Proceeding of the IFAC World Congress, Vol. F, International Federation of Automatic Control*, 1999, pp. 473 – 478.

[86] T. Soderstrom, "Perturbation Results for Singular Values," Department of Information Technology at Uppsala University, Tech. Rep., 1999.

[87] B. Friedlander and B. Porat, "First-Order Perturbation Analysis of Singular Vectors in Singular Value Decomposition," *IEEE Transactions on Signal Processing*, vol. 56, Issue 7, pp. 3044–3049, 2008.

[88] T. Kato, *Perturbation Theory for Linear Operators*. New York: Springer-Verlag, 1995.

[89] M. Ilak and C. W. Rowley, "Modeling of transitional channel flow using balanced proper orthogonal decomposition," *Physics of Fluids*, vol. 20, 2008.

[90] W. E. Roth, "On direct product matrices," *Bulletin of the American Mathematical Society*, vol. 40, pp. 461–468, 1934.

[91] V. Faber and T. Manteuffel, "Necessary and Sufficient Conditions for the Existence of a Conjugate Gradient Method," *SIAM Journal on Numerical Analysis*, vol. 21, pp. 352–362, 1984.

[92] E. Wong and B. Hajek, *Stochastic Processes in Engineering Systems*. New York: Springer-Verlag, 1985.

[93] M. West and J. Harrison, *Bayesian Forecasting and Dynamic Models*. New York: Springer-Verlag, 1989.

[94] B. Friedlander and B. Porat, "The modified Yule-Walker method of ARMA spectral estimation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-20, no. 2, pp. 158–173, 1984.

[95] J.-N. Juang, *Applied System Identification*. Englewood Cliffs, NJ: Prentice Hall, 1994.

[96] N. Cressie, *Statistics for Spatial Data*. Wiley, 1991.

[97] X. Li, A. Scaglione, and T.-H. Chang, "Optimal sensor placement for hybrid state estimation in smart grid," in *IEEE International Conference on Acoustic, Speech and Signal Processing*, 2013, pp. 5253–5257.

[98] V. Gupta, M. Sharma, and N. Thakur, "Optimization Criteria for Optimal Placement of Piezoelectric Sensors and Actuators on a Smart Structure: A Technical Review," *Journal of Intelligent Material Systems and Structures*, vol. 21, pp. 1227 – 1243, 2010.

[99] P. Mokhasi and D. Rempfer, "Optimized sensor placement for urban flow measurement," *Physics of Fluids*, vol. 16(5), pp. 1758–1764, 2004.

[100] K. K. Chen and C. W. Rowely, "Fluid flow control applications of h_2 optimal actuator and sensor placement," in *Proceedings of American Control Conference*, 2014, pp. 4044 – 4049.

[101] P. U. Sik, C. J. Weon, Y. Wan-Suk, L. M. Hyung, S. Kwon, L. J. Myung, L. M. Choel, and H. S. Hyun, "Optimal Placement of Sensors and Actuators using Measures of Modal Controllability and Observability in a Balanced Coordinate," *KSME International Journal*, vol. 17(1), pp. 11 –22, 2003.

[102] A. Barbagallo, D. Sipp, and P. Schmid, "Input-output measures for model reduction and closed-loop control: Application to global modes," *Journal of Fluid Mechanics*, vol. 685, pp. 23–53, 2011.

[103] V. Gupta, T. Chung, B. Hassibi, and R. M. Murray, "Sensor Scheduling Algorithms Requiring Limited Computation," in *Proceeding of IEEE ICASSP*, 2004, pp. 17–21.

[104] B. Sinopoli and etc, "Kalman Filtering with Intermittent Observations ," *IEEE transactions on Automatic Control*, vol. 49(9), pp. 1453–1464, 2004.

[105] Z. Sunberg, S. Chakravorty, and R. S. Erwin, "Information Space Receding Horizon Control ," *IEEE transactions on cybernetics*, vol. 43(6), pp. 2255–2260, 2013.

[106] E. P. Cameron Browne and D. Whitehouse, "A Survey of Monte Carlo Tree Search Methods," *IEEE transactions on Computational Intelligence and AI IN Games*, vol. 4(1), pp. 1– 49, 2012.

[107] S. Gelly, L. Kocsis, M. Schoenauer, M. Sebag, D. Silver, C. Szepesvari, and O. Teytaud, "The grand challenge of computer Go: Monte Carlo tree search and extensions," *Communications of the ACM*, vol. 55(3), pp. 106 – 113, 2012.

[108] A.M.King, U. Desai, and R.E.Skelton, "A generalized approach to q-markov covariance equivalent realizations for discrete systems," *Automatica*, vol. 24, no. 4, pp. 507–515, 1988.

APPENDIX A

KALMAN FILTER (KF)

Consider the system:

$$\begin{aligned} x_k &= A_k x_{k-1} + B_k u_k + w_k, \\ z_k &= C_k x_k + v_k, \end{aligned} \tag{A.1}$$

where $w_k \sim N(0, Q_k), v_k \sim N(0, R_k)$.

Kalman Filter uses a series of measurements observed to estimate the states of the system. There are two steps involved in the Kalman Filter: prediction and update.

In a prediction step, Kalman Filter produces an estimate of the mean and covariance of the states at the current time step. The prediction step is:

$$\hat{x}_{k|k-1} = A_k \hat{x}_{k-1|k-1} + B_k u_k, \tag{A.2}$$

$$P_{k|k-1} = A_k P_{k-1|k-1} A_k' + Q_k, \tag{A.3}$$

where $\hat{x}_{k|k-1}$ and $P_{k|k-1}$ are known as the prior state estimate and error covariance at time step k.

Given the measurements at time step k, Kalman Filter improves the estimates in the update step, and the posterior mean and error covariance are estimated as

follows:

$$\tilde{y}_k = z_k - C_k \hat{x}_{k|k-1}, \tag{A.4}$$

$$S_k = C_k P_{k|k-1} C_k' + R_k, \tag{A.5}$$

$$K_k = P_{k|k-1} C_k' S_k^{-1}, \tag{A.6}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k, \tag{A.7}$$

$$P_{k|k} = (I - K_k C_k) P_{k|k-1}. \tag{A.8}$$

APPENDIX B

EIGENSYSTEM REALIZATION ALGORITHM (ERA)

Consider the system:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k, \\ y_k &= Cx_k + Du_k, \end{aligned} \tag{B.1}$$

where $A \in \Re^{N\times N}, B \in \Re^{N\times p}, C \in \Re^{q\times N}$.

The Markov parameters are denoted as:

$$Y_0 = D, Y_1 = CB, Y_2 = CAB, \cdots, Y_k = CA^{k-1}B, \tag{B.2}$$

and can be computated from impule-response simulatons or experiments.

A Hankel matrix is constructed using the Markov parameters as:

$$H_0 = \begin{pmatrix} Y_1 & Y_2 & \cdots & Y_\beta \\ Y_2 & Y_3 & \cdots & Y_{1+\beta} \\ \vdots & \vdots & \vdots & \vdots \\ Y_\alpha & Y_{1+\alpha} & \cdots & Y_{\alpha+\beta-1} \end{pmatrix}, \tag{B.3}$$

where $\alpha \geq N$, $\beta \geq N$.

ERA solves the singular value decomposition,

$$H_0 = R\Sigma S', \tag{B.4}$$

where

$$\Sigma = \begin{pmatrix} \Sigma_N & 0 \\ 0 & 0 \end{pmatrix}, \tag{B.5}$$

with $\Sigma_N = diag\{\sigma_1, \cdots, \sigma_N\}$ are ordered as $\sigma_1 \geq \sigma_2 \geq \sigma_N \geq 0$. Let R_N, S_N be the matrices formed by the first N columns of R and S respectively, and hence,

$$H_0 = R_N \Sigma_N S_N', \tag{B.6}$$

where $R_N' R_N = I_N = S_N' S_N$.

Then the system realization for ERA is:

$$\begin{aligned} \hat{A} &= \Sigma_N^{-1/2} R_N' H_1 S_N \Sigma_N^{-1/2}, \\ \hat{B} &= \text{ first p columns of } \Sigma_N^{1/2} S_N', \\ \hat{C} &= \text{ first q rows of } R_N \Sigma_N^{1/2}, \end{aligned} \tag{B.7}$$

where

$$H_1 = \begin{pmatrix} Y_2 & Y_3 & \cdots & Y_{1+\beta} \\ Y_3 & Y_4 & \cdots & Y_{2+\beta} \\ \vdots & \vdots & \vdots & \vdots \\ Y_{1+\alpha} & Y_{2+\alpha} & \cdots & Y_{\alpha+\beta} \end{pmatrix}, \tag{B.8}$$

APPENDIX C

OPTIMAL TWO-STAGE KALMAN FILTER (OTSKF)

Consider the system:

$$\begin{aligned} x_{k+1} &= A_k x_k + E_k d_k + w_k^x, \\ y_k &= H_k x_k + \eta_k, \end{aligned} \tag{C.1}$$

where $x_k \in \Re^n$ is the state vector, $y_k \in \Re^m$ the observation vector, $d_k \in \Re^p$ the unknown inputs. The process noise w_k and the measurement noise η_k are zero-mean uncorrelated white noise sequences with covariances Q_k, R_k respectively.

Assume the dynamics of d_k is:

$$d_{k+1} = d_k + w_k^d, \tag{C.2}$$

where w_k^d is a zero-mean white noise sequence with covariances Q_k^d, and assume $E\{w_k(w_l^d)'\} = Q_k^{xd}\delta_{kl}$.

Treating x_k and d_k as the augmented state, an augmented state Kalman filter can be used to produce the optimal state estimate, or two parallel reduced-order filter can be used.

$$x_{k|k} = \bar{x}_{k|k} + V_k d_{k|k}, \tag{C.3}$$

$$P_{k|k}^x = P_{k|k}^{\bar{x}} + V_k P_{k|k}^d V_k', \tag{C.4}$$

where $\bar{x}_{k|k}$ is given by:

$$\bar{x}_{k|k-1} = A_{k-1}\bar{x}_{k-1|k-1} + \bar{u}_{k-1}, \tag{C.5}$$

$$\bar{x}_{k|k} = \bar{x}_{k|k-1} + K_k^{\bar{x}}(y_k - H_k\bar{x}_{k|k-1}), \tag{C.6}$$

$$P_{k|k-1}^{\bar{x}} = A_{k-1}P_{k-1|k-1}^{\bar{x}}A_{k-1}' + \bar{Q}_{k-1}, \tag{C.7}$$

$$K_k^{\bar{x}} = P_{k|k-1}^{\bar{x}}H_k'(H_kP_{k|k-1}^{\bar{x}}H_k' + R_k)^{-1}, \tag{C.8}$$

$$P_{k|k}^{\bar{x}} = (I - K_k^{\bar{x}}H_k)P_{k|k-1}^{\bar{x}}. \tag{C.9}$$

$d_{k|k}$ is given by

$$d_{k|k-1} = d_{k-1|k-1}, \tag{C.10}$$

$$d_{k|k} = d_{k|k-1} + K_k^d(y_k - H_k\bar{x}_{k|k-1} - S_kd_{k|k-1}), \tag{C.11}$$

$$P_{k|k-1}^d = P_{k-1|k-1}^d + Q_{k-1}^d, \tag{C.12}$$

$$K_k^d = P_{k|k-1}^dS_k'(H_kP_{k|k-1}^{\bar{x}}H_k' + R_k + S_kP_{k|k-1}^dS_k')^{-1}, \tag{C.13}$$

$$P_{k|k}^d = (I - K_k^dS_k)P_{k|k-1}^d, \tag{C.14}$$

with coupling equations

$$\bar{u}_k = (\bar{U}_{k+1} - U_{k+1})d_{k|k}, \tag{C.15}$$

$$\bar{Q}_k = Q_k - Q_k^{xd}\bar{U}_{k+1}' - U_{k+1}(Q_k^{xd} - \bar{U}_{k+1}Q_k^d)', \tag{C.16}$$

$$\bar{U}_k = A_{k-1}V_{k-1} + E_{k-1}, \tag{C.17}$$

$$S_k = H_kU_k, \tag{C.18}$$

$$U_k = \bar{U}_k + (Q_{k-1}^{xd} - \bar{U}_kQ_{k-1}^d)(P_{k|k-1}^d)^{-1}, \tag{C.19}$$

$$V_k = U_k - K_k^{\bar{x}}S_k. \tag{C.20}$$

APPENDIX D

UNBIASED MINIMUM-VARIANCE FILTER (UMV)

Consider the system:

$$\begin{aligned} x_{k+1} &= A_k x_k + E_k d_k + w_k^x, \\ y_k &= H_k x_k + \eta_k, \end{aligned} \tag{D.1}$$

where $x_k \in \Re^n$ is the state vector, $y_k \in \Re^m$ the observation vector, $d_k \in \Re^p$ the unknown inputs. The process noise w_k and the measurement noise η_k are zero-mean uncorrelated white noise sequences with covariances Q_k, R_k respectively. Assme rank $(E_k) = p$, rank $(H_k) = m(\geq p)$,

The observer matching condition needs to be satisfied is:

$$\text{rank } (H_k E_{k-1}) = p. \tag{D.2}$$

The unbiased minimum-variance filter is given as follows.

$$x_{k|k} = x_{k|k-1} + L_k(y_k - H_k x_{k|k-1}), \tag{D.3}$$

$$L_k = K_k + (I - K_k H_k)E_{k-1}(E'_{k-1}H'_k C_k^{-1} H_k E_{k-1})^{-1} E'_{k-1} H'_k C_k^{-1}, \tag{D.4}$$

$$P^x_{k|k} = (I - L_k H_k)P^x_{k|k-1}(I - L_k H_k)' + L_k R_k L'_k, \tag{D.5}$$

where

$$x_{k|k-1} = A_{k-1}x_{k-1|k-1}, \tag{D.6}$$

$$P^x_{k|k-1} = A_{k-1}P^x_{k-1|k-1}A'_{k-1} + Q_{k-1}, \tag{D.7}$$

$$K_k = P^x_{k|k-1}H'_kC_k^{-1}, \tag{D.8}$$

$$C_k = H_kP^x_{k|k-1}H'_k + R_k. \tag{D.9}$$

APPENDIX E

COMPARISON OF Q-MARKOV COVARIANCE EQUIVALENT REALIZATION (Q-MARKOV COVER) AND ERA

A q-Markov covariance equivalent realization (q-Markov COVER) [108] is a state realization which matches exactly a finite portion of the impulse response sequence (Markov parameters) and the output autocorrelation sequence of a linear system. In this section, first, we review the general approach to construct all minimal stable q-Markov COVERs, and then we compare the q-Markov COVERs with the ERA.

E.1 Problem Statement

Given a stable linear discrete time system,

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k), \\ y(k) &= Cx(k) + Du(k), \end{aligned} \tag{E.1}$$

where $x(k) \in \Re^N, y(k) \in \Re^{n_o}, u(k) \in \Re^{n_i}$ are the state, output vector, and input vector respectively. Denote the impulse response sequence as:

$$\{h_i\}_0^\infty = \{h_0, h_1, h_2, \cdots\}, \tag{E.2}$$

Perturb the system with zero mean unit covariance white noise process, and denote the output autocorrelation sequence as:

$$\{r_i\}_0^\infty = \{r_0, r_1, r_2, \cdots\}, \tag{E.3}$$

where $r_i = \lim_{k\to\infty} Ey(k+i)y(k)'$, and $(.)'$ denotes the transpose of $(.)$.

A q-Markov COVER of the system generating the data sequence $\{h_i\}_0^\infty$ and $\{r_i\}_0^\infty$ is a stable state space realization:

$$\begin{aligned}\hat{x}(k+1) &= \hat{A}x(k) + \hat{B}u(k),\\ \hat{y}(k) &= \hat{C}\hat{x}(k) + \hat{D}u(k),\end{aligned} \tag{E.4}$$

with Markov parameters

$$\hat{h}_0 = \hat{D}, \hat{h}_i = \hat{C}\hat{A}^{i-1}\hat{B}, i = 1, 2, \cdots, \tag{E.5}$$

and covariance parameters

$$\hat{r}_i = \hat{C}\hat{A}^i\hat{X}\hat{C}' + \hat{h}_i\hat{D}', i = 0, 1, 2, \cdots. \tag{E.6}$$

such that:

$$\begin{aligned}\hat{h}_i &= h_i, i = 0, 1, 2, \cdots, q,\\ \hat{r}_i &= r_i, i = 0, 1, 2, \cdots, q-1.\end{aligned} \tag{E.7}$$

Here, the state covariance matrix is

$$\hat{X} = \lim_{k\to\infty} E\hat{x}(k)\hat{x}(k)', \tag{E.8}$$

and satisfies the Liapunov equation:

$$\hat{X} = \hat{A}\hat{X}\hat{A}' + \hat{B}\hat{B}', \hat{X} > 0. \tag{E.9}$$

The realization problem is that given the finite data sequence $\{h_i\}_0^q$ and $\{r_i\}_0^{q-1}$, find all minimal stable q-Markov COVERs. The model reduction problem is that given a full order model $\{A, B, C, D, X\}$ with dimension N, find all minimal stable COVERs of reduced dimension.

E.2 Review of the q-Markov COVER Algorithm

Consider the sytem (E.1), it can be seen that:

$$\underbrace{\begin{pmatrix} y(k) \\ y(k+1) \\ \vdots \\ y(k+q-1) \end{pmatrix}}_{Y_q(k)} = \underbrace{\begin{pmatrix} C \\ CA \\ \vdots \\ CA^{q-1} \end{pmatrix}}_{O_q} x(k) + \underbrace{\begin{pmatrix} h_0 & 0 & \cdots & 0 & 0 \\ h_1 & h_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ h_{q-1} & h_{q-2} & \cdots & h_1 & h_0 \end{pmatrix}}_{H_1} \underbrace{\begin{pmatrix} u(k) \\ u(k+1) \\ \vdots \\ u_{k+q-1} \end{pmatrix}}_{U_q(k)}, \tag{E.10}$$

Perturb the system (E.1) with zero mean, unit covariance white noise, then the output autocorrelation matrix:

$$R_q = \lim_{k\to} EY_q(k)Y_q(k)' = \begin{pmatrix} r_0 & r_1' & \cdots & r_{q-1}' \\ r_1 & r_0 & \cdots & r_{q-2}' \\ \vdots & \vdots & \cdots & \vdots \\ r_{q-1} & r_{q-2} & \cdots & r_0 \end{pmatrix}, \tag{E.11}$$

and from (E.10),

$$R_q = O_q X O_q' + H_1 H_1'. \tag{E.12}$$

Therefore, define the first data matrix:

$$D_q = O_q X O_q' = R_q - H_1 H_1'. \tag{E.13}$$

Similarly, denote

$$\bar{H}_1 = \begin{pmatrix} h_1 & h_0 & 0 & \cdots & 0 & 0 \\ h_2 & h_1 & h_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ h_q & h_{q-1} & h_{q-2} & \cdots & h_1 & h_0 \end{pmatrix}, \tag{E.14}$$

then

$$Y_q(k+1) = O_q A x(k) + \bar{H}_1 U_{q+1}(k). \tag{E.15}$$

Hence,

$$R_q = O_q A X A' O_q' + \bar{H}_1 \bar{H}_1', \tag{E.16}$$

and define the second data matrix:

$$\bar{D}_q = O_q A X A' O_q' = R_q - \bar{H}_1 \bar{H}_1'. \tag{E.17}$$

The q-Markov COVER algorithm has the following steps.

First, generate a full rank factorization of the first data matrix:

$$\begin{aligned} &D_q = P \Lambda P', \Lambda > 0, \\ &rank(D_q) = rank(P) = rank(\Lambda) = l, \end{aligned} \tag{E.18}$$

where $P \in \Re^{n_o q \times l}$, and $\Lambda \in \Re^{l \times l}$.

Partition P into blocks, and define two new matrices

$$P = \begin{pmatrix} P_0 \\ P_1 \\ \vdots \\ P_{q-1} \end{pmatrix}, F = \begin{pmatrix} P_1 \\ \vdots \\ P_{q-1} \end{pmatrix}, \bar{P} = \begin{pmatrix} F \\ G \end{pmatrix}, \tag{E.19}$$

where G is an $n_o \times l$ matrix to be determined such that

$$\bar{D}_q = \bar{P}\Lambda\bar{P}'. \tag{E.20}$$

All the matrices G can be expressed as:

$$\begin{aligned} G &= G_1 + G_2 V * G_3, \\ G_1 &= \bar{d}_q'(F^+)'\Lambda^{-1}(I - (I_F^+ F) \times ((I - F^+F)\Lambda^{-1}(I - F^+F))^+\Lambda^{-1}), \\ G_2 &= (\bar{D}_{qq} - \bar{d}_q'\bar{D}_{q-1}^+\bar{d}_q)^{1/2}U, \\ G_3 &= W^*((I - F^+F)\Lambda^{-1}(I - F^+F))^{+/2}\Lambda^{-1}, \end{aligned} \tag{E.21}$$

where $(.)^{+/2}$ is the Moore-Penrose inverse of $(.)^{1/2}$,

$$\bar{D}_q = \begin{pmatrix} \bar{D}_{q-1} & \bar{d}_q \\ \bar{d}_q' & \bar{D}_{qq} \end{pmatrix}, \tag{E.22}$$

U is a column unitary matrix defines the range space of $(\bar{D}_{qq} - \bar{d}_q\bar{D}_{q-1}^+\bar{d}_q)$, W is a column unitary matrix defines the null space of F, and V is an arbitrary column unitary matrix.

Special Case 1. If $\Lambda = I$, then the final expression for G is :

$$G = \bar{d}_q'(F^+)' + (\bar{D}_{qq} - \bar{d}_q'\bar{D}_{q-1}^+\bar{d}_q)^{1/2}UV'W'(I - F^+F). \tag{E.23}$$

A q-Markov COVER is:

$$A_q = P^+\bar{P}, B_q = P^+M_q = P^+\begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_q \end{pmatrix}, C_q = P_0, D_q = h_0, \hat{X} = \Lambda, \tag{E.24}$$

with dimension $n_x = rank(D_q)$.

E.3 Comparison with ERA

Now, reconsider the data matrices $D_q, \bar{D}_q$ in the q-Markov COVER. If define the Hankel matrices constructed in ERA as:

$$H_q = \begin{pmatrix} h_1 & h_2 & \cdots & h_{t_{ss}-1} \\ h_2 & h_3 & \cdots & h_{t_{ss}} \\ \vdots & \vdots & \cdots & \vdots \\ h_q & h_{q+1} & \cdots & h_{q+t_{ss}-1} \end{pmatrix}, H_{q+1} = \begin{pmatrix} h_2 & h_3 & \cdots & h_{t_{ss}} \\ h_3 & h_4 & \cdots & h_{t_{ss}+1} \\ \vdots & \vdots & \cdots & \vdots \\ h_{q+1} & h_{q+2} & \cdots & h_{q+t_{ss}} \end{pmatrix} \tag{E.25}$$

where t_{ss} is a finite constant such that $\|A^{t_{ss}}\| \approx 0$. Then the following results hold.

Proposition 1 *The first and second data matrices in q-Markov COVER are related to the Hankel matrices constructed in ERA:* $D_q = H_qH_q', \bar{D}_q = H_{q+1}H_{q+1}'$.

Proposition 2 *The system* (A_b, B_b, C_b, D_b) *constructed using ERA is a special case of the q-Markov COVER.*

The difference is q-Markov COVER algorithm collects the first $q-1$ output auto-correlations $\{r_i\}_0^{q-1}$. To preserve the same information, ERA needs to collect more Markov parameters, i.e., ERA needs to collect $\{h_i\}_0^{t_{ss}+q}$.

E.3.1 Proof of Proposition 1

We prove the above results in two ways. First, by definition:

$$D_q = O_q X O_q'. \tag{E.26}$$

Consider the state $x(k)$ with white noise perturbation:

$$x(k) = \sum_{i=1}^{\infty} A^{i-1} B u(k-i), \tag{E.27}$$

as $k \to \infty$, suppose there exists a constant t_{ss}, such that

$$x_k = \underbrace{\begin{pmatrix} B & AB & \cdots & A^{t_{ss}}B \end{pmatrix}}_{X_b} \underbrace{\begin{pmatrix} u_{k-1} \\ u_{k-2} \\ \vdots \\ u_{k-t_{ss}-1} \end{pmatrix}}_{U_k} \tag{E.28}$$

Then

$$X = \lim_{k\to\infty} E(x_k x_k') = X_b E(U_k U_k') X_b' = X_b X_b'. \tag{E.29}$$

Hence,

$$D_q = (O_q' X_b)(O_q' X_b)' = H_q H_q'. \tag{E.30}$$

Similarily, by definition, $\bar{D}_q$:

$$\bar{D}_q = O_q A X A' O_q' = \begin{pmatrix} CA \\ CA^2 \\ \vdots \\ CA^q \end{pmatrix} X_b E(U_k U_k') X_b' A' O_q' = H_{q+1} H_{q+1}'. \tag{E.31}$$

Note that H_q contains first qn_o rows of the full Hankel matrix constructed using ERA/BPOD.

The second way is to expand R_q and H_1.

$$R_q = \lim_{k\to\infty} E Y_q(k) Y_q(k)'. \tag{E.32}$$

The output $y(k)$ is:

$$y(k) = \sum_{i=0}^{t_{ss}} h_i u(k-i) = C X_b U_k + D u(k). \tag{E.33}$$

Therefore,

$$Y_q(k) = \begin{pmatrix} y(k) \\ y(k+1) \\ \vdots \\ y(k+q-1) \end{pmatrix} = \underbrace{\begin{pmatrix} CX_bU_k \\ CX_bU_{k+1} \\ \vdots \\ CX_bU_{k+q-1} \end{pmatrix}}_{T_1} + \underbrace{\begin{pmatrix} Du(k) \\ Du(k+1) \\ \vdots \\ Du(k+q-1) \end{pmatrix}}_{T_2}, \tag{E.34}$$

Notice that

$$U_k u(k)' = \begin{pmatrix} u(k-1) \\ u(k-2) \\ \vdots \\ u(k-t_{ss}-1) \end{pmatrix} u(k)' = 0. \tag{E.35}$$

Therefore, it can be proved that $T_1 T_2' = T_2' T_1 = 0$.

$$T_2 T_2' = \begin{pmatrix} D^2 & & & \\ & D^2 & & \\ & & \cdots & \\ & & & D^2 \end{pmatrix} \tag{E.36}$$

So

$$T_1 T_1' = \begin{pmatrix} CX_b U_k U_k' X_b' C' & CX_b U_k U_{k+1}' X_b' C' & \cdots & CX_b U_k U_{k+q-1}' X_b' C' \\ \vdots & \vdots & \cdots & \vdots \\ CX_b U_{k+q-1}' U_k X_b' C' & CX_b U_{k+q-1}' U_{k+1}' X_b' C' & \cdots & CX_b U_{k+q-1}' U_{k+q-1}' X_b' C' \end{pmatrix} \tag{E.37}$$

Expectation of the diagonal terms are the same

$$\begin{aligned} E(CX_b U_k U_k' X_b' C') = \cdots = E(CX_b U_{k+q-1} U_{k+q-1}' X_b' C') = CX_b X_b' C' \\ = \begin{pmatrix} h_1 & h_2 & \cdots & h_{t_{ss}} \end{pmatrix} \begin{pmatrix} h_1' \\ h_2' \\ \vdots \\ h_{t_{ss}}' \end{pmatrix} = \sum_{i=1}^{t_{ss}} h_i h_i'. \end{aligned} \tag{E.38}$$

Therefore,

$$D_q = R_q - H_1 H_1' = \lim_{k\to\infty} EY_q(k)Y_q(k)' - H_1 H_1'$$
$$= E(T_1 T_1' + T_2 T_2') - H_1 H_1' = \begin{pmatrix} \sum_{i=1}^{t_{ss}} h_i h_i' + h_0 h_0' - h_0 h_0' & \cdots & \cdots \\ & & \\ & & \end{pmatrix}, \quad \text{(E.39)}$$

If expand all the terms, it can be proved that $D_q = H_q H_q'$.

E.3.2 Proof of Proposition 2

Suppose the SVD of the Hankel matrix H_q is:

$$H_q = U_q \Sigma_q V_q'. \quad \text{(E.40)}$$

Then

$$D_q = H_q H_q' = U_q \Sigma_q^2 U_q' = P \Lambda P' \quad \text{(E.41)}$$

Consider a special case when $\Lambda = \Sigma_q$.

$$D_q = \underbrace{U_q \Sigma_q^{1/2}}_{P} \underbrace{\Sigma_q}_{\Lambda} \underbrace{\Sigma_q^{1/2} U_q'}_{P'}. \quad \text{(E.42)}$$

Therefore,

$$P = U_q \Sigma_q^{1/2} \rightarrow C_q = \text{the first } n_o \text{ rows of } U_q \Sigma_q^{1/2} = C_b \quad \text{(E.43)}$$

Consider $B_q = P^+ M_q = \Sigma_q^{-1/2} U_q' O_q B$

In ERA:

$$H_q = \underbrace{U_q \Sigma_q^{1/2}}_{P_\alpha} \underbrace{\Sigma_q^{1/2} V_q'}_{Q_\beta}, \tag{E.44}$$

$$\begin{aligned}
&\begin{pmatrix} B & AB & \cdots & A^{t_{ss}}B \end{pmatrix} = Q_\beta = (P_\alpha)^+ H_q \\
&= \Sigma_q^{-1/2} U_q' \begin{pmatrix} CB & CAB & \cdots & CA^{t_{ss}}B \\ CAB & CA^2B & \cdots & CA^{t_{ss}+1}B \\ \vdots & \vdots & \cdots & \vdots \\ CA^{q-1}B & CA^qB & \cdots & CA^{q+t_{ss}-1}B \end{pmatrix},
\end{aligned} \tag{E.45}$$

where ERA ROM B_b are the first n_i columns of Q_β:

$$B_b = \Sigma_q^{-1/2} U_q' \begin{pmatrix} CB \\ CAB \\ \vdots \\ CA^{q-1}B \end{pmatrix} = \Sigma_q^{-1/2} U_q' M_q = P^+ M_q = B_q. \tag{E.46}$$

Now if we choose $\bar{P}$:

$$\bar{P} = H_{q+1} V_q \Sigma_q^{-1/2}, \tag{E.47}$$

Then

$$A_q = P^+ \bar{P} = \Sigma_q^{-1/2} U_q' H_{q+1} V_q \Sigma_q^{-1/2} = A_b. \tag{E.48}$$

We only need to verify $\bar{P}$ is one of the solution given in the q-Markov general

form, and we verify

1) $\bar{P}, \Lambda$ are SVD of $\bar{D}_q$,

$$\bar{P}\Lambda\bar{P}' = H_{q+1}V_q\Sigma_q^{-1/2}\Sigma_q\Sigma_q^{-1/2}V_q'H_{q+1}' = H_{q+1}H_{q+1}' = \bar{D}_q \tag{E.49}$$

2) $\bar{P} = PA$,

$$H_{q+1} = P_\alpha A Q_\beta \rightarrow \bar{P} = H_{q+1}(Q_\beta^+) = P_\alpha A = PA. \tag{E.50}$$

3)$\hat{X} = \Sigma_q$ satisfy $\hat{X} = A_q\hat{X}A_q' + B_qB_q'$.

The consistent with the Liapunov equation is proved in the paper [108].

Summary

The ERA is one special case of q-Markov COVER, where the controllability matrix and ovservability matrix are balanced, and in order to construct the same realization as ERA, we should choose $\Lambda = \Sigma_q, \bar{P} = H_{q+1}V_q'\Sigma_q^{-1/2}$ in q-Markov COVER algorithm.