UNIVERSITY OF CALIFORNIA SAN DIEGO

Model Reduction and Iterative LQR for Control of High-Dimensional Nonlinear Systems

A thesis submitted in partial satisfaction of the
requirements for the degree of Master of Science

in

Computer Science

by

Yizhe Huang

Committee in charge:

Professor Boris Kramer, Chair
Professor Chung Kuan Cheng, Co-Chair
Professor Sicun Gao

2020

The thesis of Yizhe Huang is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Co-Chair

Chair

University of California San Diego

2020

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT OF THE THESIS

Model Reduction and Iterative LQR for Control of High-Dimensional Nonlinear Systems

by

Yizhe Huang

Master of Science in Computer Science

University of California San Diego, 2020

Professor Boris Kramer, Chair
Professor Chung Kuan Cheng, Co-Chair

The control of high-dimensional nonlinear systems remains a challenge in control theory. We propose a framework to design controllers for high-dimensional nonlinear systems. The controller is designed by the use of the iterative linear quadratic regulator (ILQR) algorithm, an algorithm that computes control by iteratively applying the linear quadratic control (LQR) algorithm on the local linearizations of the system at each time step. The high dimensionality is addressed by model reduction, which constructs reduced-order models (ROMs) that approximate the dynamics of the original full-order models (FOMs). We apply balanced truncation (BT), a system-theoretic, trajectory-independent

model reduction technique for open-loop systems, and its extension for closed-loop systems, linear quadratic Gaussian balanced truncation (LQG-BT). Numerical experiments of this framework are performed on an 1D Burgers' equation, where the performances of ROMs of different orders, as well as the choice between BT and LQG-BT, are compared and discussed. We find that the ILQR algorithm produces good control on ROMs constructed either by BT or LQG-BT. While BT outperforms LQG-BT in terms of accuracy in open-loop simulations, LQG-BT results in better control in the closed-loop system when combined with ILQR.

Chapter 1

Introduction

In this chapter we first introduce the motivation for our work in Section 1.1. We then review the works related to the ILQR algorithm, model reduction, and control of Burgers' equation in Section 1.2. Finally, we summarize our contribution in this work in Section 1.3.

1.1 Motivation

Systems in science and engineering are inherently nonlinear, and their control and operation remains at the heart of modern computational science and engineering. Control theory provides us with a framework to design a controller, which applies a control input to a system to drive it to the desired target state. The performance of the controller is evaluated by a cost function, which usually consists of the distance between the system state and the target state, as well as the energy cost of the control input. A good controller drives the system state close to target using a low (or potentially optimal) energy for the control.

A well-studied class of control problems exists for linear time-invariant (LTI) systems, in which the derivative of the system state with respect to time (or, in the discrete-time case, the next system state) depends linearly on the current state and control input. Feedback control algorithms such as the linear quadratic regulator (LQR) and the linear

quadratic Gaussian regulator (LQG) provide optimal solutions, which can be computed with reasonable computational effort even for large-scale systems. However, systems in real life are usually nonlinear, and controlling these systems requires case-by-case development of control strategies, often only works locally, and requires much more computational effort than in the linear case. One obvious approach is to find linear approximations to these systems and apply techniques like LQR, but depending on the extent of the nonlinearity, the control found may be very far from optimal. Therefore to get a good control on these systems we have to adopt fully nonlinear controllers, which are often more costly in computational power.

Many systems in engineering are modeled with systems of a few degrees of freedom, such as in classical robotics where position and velocity are used for all rigid components, or in six-DOF airplane models, and others. The computational cost of the algorithm is usually not a concern when dealing with these problems. However, certain problems can have a very high number of dimensions, even on the scale of millions. Those problems are typical when dealing with discretized partial differential equations (PDEs). For example, if we want to represent the temperature of a plate, where space is continuous, we can use a spatial discretization technique to convert a PDE into a large-scale system of ordinary differential equations (ODEs). Then the temperature evaluated on each of the grid nodes becomes a state of the discretized system. The finer we discretize the spatial variable, the more accurate the simulation is, but at the same time, we have to solve a problem of a larger dimension. On those large systems, the computational cost is a concern, especially on controllers that depend on real-time sensor input.

To address these two problems, which means to design a controller for high-dimensional nonlinear systems, we propose to use a combination of model reduction and the ILQR algorithm. ILQR is a nonlinear controller that works well on low-dimensional nonlinear systems with a quadratic cost function. We detail our contributions in the next section.

1.2 Related Work

We discuss related work on the ILQR algorithm in Section 1.2.1, on model reduction techniques in Section 1.2.2 and on controlling Burgers' equation in Section 1.2.3.

1.2.1 ILQR

The LQR algorithm is a classic and well-known algorithm in the theory of optimal control. It provides an automatic way of finding the state-feedback controller. It also plays an important role in solving the related LQG problem, where the system is influenced by a white Gaussian noise in addition to the control input, and the goal is to minimize the expected cost.

However, real-life control problems often do not fit this setting, since the system might not be linear, and the cost function might not be quadratic. The condition for the system equation to be linear and for the cost to be quadratic is fairly strong and often violated in real-life systems. The control engineer may choose to find the linear and quadratic approximation to the system and then apply the LQR algorithm, but in the case of highly nonlinear systems the controller found will likely be far from optimal.

In 2004, the authors in [13] developed the ILQR algorithm that extends LQR by relaxing one of its conditions. The ILQR algorithm, in contrast to standard LQR, does not require the system equation to be linear, though it still assumes quadratic cost function. The core idea of ILQR is that instead of deriving a fixed linearization of the nonlinear dynamics beforehand, we can compute the local linear approximation by Jacobian whenever needed, which means that the linear approximation will be different at different points on the trajectory. These local linearizations are then used in a modified LQR algorithm to find the nominally optimal controller and the corresponding nominal trajectory. The next iteration would then use the new trajectory to recompute the local linearizations and repeat the process, resulting in the iterative nature of ILQR. While the

algorithm is not guaranteed to converge to the optimal control, on most systems the cost of the given solution decreases across iterations and eventually converges.

The ILQR algorithm significantly extends the range of application of LQR methods, as it can be applied to highly nonlinear systems and the constraint of quadratic cost is often satisfied when the cost is energy-related, which is common in physical systems. In [27] ILQR is applied to an 11-dimensional, highly nonlinear dynamic model for a single quadrotor UAV with a cable-suspended heavy rigid body. The authors in [30] apply ILQR to lower limb exoskeleton and compares its performance with adaptive PD controllers. The authors in [31] uses ILQR to control two quadrotors carrying one cable-suspended payload. However, despite the apparent range of applications, these systems are similar in that the models are usually low-dimensional, not exceeding 11 dimensions. As the running time of each iteration in ILQR increases quadratically regarding the number of dimensions, ILQR lost computationally tractability on high-dimensional systems.

There are also efforts to extend ILQR to other scenarios. An iterative LQG method is developed in [18] and is very similar to the ILQR algorithm. The authors in [23] extend ILQR to systems with nonquadratic cost and apply it to differential-drive robots and quadrotor helicopters in environments with obstacles. The authors in [34] propose a way of learning local linearization of dynamics from image inputs and tests it on problems like planar system (navigation), inverted pendulum, and cart-pole balancing. In addition, [33] proposes constrained iterative LQR to handle the constraints in ILQR and applies it to on-road autonomous driving motion planning. However, while these extensions further expand the range of problems that ILQR can be applied to, none of these extensions address the problem that ILQR is not computationally tractable on high-dimensional systems. To the authors' knowledge, ILQR has not been used in the context of high-dimensional semi-discretized PDE systems to date.

1.2.2 Model Reduction

Large-scale dynamical systems are ubiquitous in control theory, as many problems of practical interest are large-scale. To study a PDE, we often perform space discretization to transform it into an ODE, with each point in space becomes a state dimension. Accurate semi-discretized PDEs then lead to large-scale ODEs. As these large-scale problems are often too computationally costly to study, model reduction techniques are often employed. Model reduction is a category of techniques to construct ROMs that are good approximations to the original FOM. See [15, 25, 32] for an overview of model reduction. Model reduction is appealing, as it offers the possibility to perform analysis and optimization with the ROM, and then apply those findings to the FOM, along with suitable error analysis. In the setting of control theory, this means that we are interested in computing the control input using the ROM, and then use the same input to control the original system.

The most popular model reduction technique for open-loop nonlinear systems is the proper orthogonal decomposition (POD) method [6], which is an extension of the principal component analysis method in statistics. It aims to project the high-dimensional variables into low-dimensional spaces while retaining most of the variations. It has the desired property of minimizing the average squared distance between the original data and the reduced linear representation. POD is a Galerkin-projection-based model reduction technique which has been widely useful in control theory despite its delicate dependence on the training data, for example in [8, 11, 16, 17, 22, 24, 28, 29].

System-theoretic model reduction provides an alternative to POD, as it does not rely on trajectory data from the model, and instead only uses the system representation (matrices). One of the most popular system-theoretic model reduction technique for open-loop control systems is the BT method [2]. It differs from the generic POD method that it is specifically designed for reducing LTI systems in control theory. Its goal is to

transform and truncate the model so that the ROM has both good controllability and good observability, which means that it takes little input to control the system and the state of the system is easily observable from the output. In essence, this means that we discard the components of the system that are hard to control and/or hard to observe. This enables us to find good control using the ROM, which can be then applied to the original system.

LQG-BT [3] is a modification to the standard BT. Standard BT balances the open-loop system. However, if we would like to find a feedback controller, we are constructing a closed-loop system, and LQG-BT produces ROMs that are suitable for closed-loop systems, as shown in [19, 20, 21, 26]. The reason behind this is that the BT method does not take the cost function into account, while the matrices in the quadratic cost function are incorporated in the LQG-BT algorithm. In this study, we use the standard BT and LQG-BT method to reduce the high-dimensional Burgers' equation model.

1.2.3 Control of Burgers' Equation

Burgers' equation is a PDE model that has been used in fluid mechanics and traffic flow modeling and is a popular numerical testbed in the design of methods for nonlinear systems. Burgers' equation has been given considerable attention in control theory, as it plays a central role in the modeling of flow problems. One of the first works to investigate the control of the Burgers' equation is [5], which applies LQR to the linear approximation of the system. Other works include [10] which introduces cubic Neumann boundary feedback control to improve stability, and [12] which adapts to the problem where viscosity is unknown. In these works, the emphasis is given on the theoretical properties, such as stability under certain technical conditions.

In contrast, there are also works that focus more on controlling the Burgers' equation while dealing with concrete problems under different scenarios. In [28], the authors discuss the control of coupled, multiphysics systems by investigating coupled Burgers'

equation. The author in [7] proposes to model turbulence by stochastic Burgers' equation and devises a control approach, which, although theoretically suboptimal, results in significant cost reduction in experiments. Another work,[9], studies nonlinear control of incompressible fluid flow and the application to Burgers' equation. A comparative study is done in [14], which compares the idea of three different approaches (SQP-primal dual method, Primal dual-SQP method, and Semi-smooth Newton method) to control the equation, and conducts numerical experiments comparing performances of those approaches under various parameters.

Burgers' equation is high-dimensional, as the semi-discretized PDE results in a high-dimensional ODE. Therefore, to get an accurate simulation, particularly in the presence of shocks in the solution, one has to use a large number of discretized elements. This often results in a system of order of hundreds to thousands of states. For many nonlinear control algorithms, the system is too large to be computationally tractable. In this work, we address this issue by incorporating model reduction techniques in combination with the ILQR algorithm.

1.3 Contribution

We propose a novel framework to control high-dimensional nonlinear systems. The framework uses a system-theoretic model reduction technique to obtain nonlinear ROMs, and subsequently uses the ILQR algorithm for controller design. ILQR has not been used in the context of high-dimensional semi-discretized PDE systems to date due to its computationally complexity, and our work accomplishes this by incorporating model reduction to make ILQR computationally tractable. In the analysis of the ILQR algorithm, we show that i) it converges in a single iteration when applied on LTI systems, and the control given is identical to the LQR algorithm, which is optimal; ii) it does not converge to the optimal control on a linear system with quadratic control, when the control is initialized

to zero; iii) it may take many iterations to converge on a quadratic systems with linear control, when the control is initialized to zero. In addition, to demonstrate the applicability of this framework to high-dimensional systems, we study a problem of controlling flow through 1D Burgers' equation. In this way, we devise a new way to control the nonlinear Burgers' equation by applying the aforementioned framework.

Chapter 2

Iterative Linear Quadratic Regulator

In this chapter, we first introduce the nonlinear control problem in Section 2.1. Then, in Section 2.2 we introduce the ILQR method, first developed in [13], which extends LQR to nonlinear systems. It iteratively computes a local linearization around a nominal trajectory, and uses the local linearization in a modified LQR algorithm to compute a local optimal control. This control is then used to compute the nominal trajectory in the next iteration, and the process proceeds until convergence. We then discuss the behavior of the algorithm when the system equation is linear in Section 2.3, and when the system equation is quadratic in Section 2.4.

2.1 Nonlinear Control Problem

Without loss of generality, we assume that a nonlinear discrete-time dynamical system of the form

$$x_{k+1} = f(x_k, u_k), \qquad k = 0, 1, \ldots, N-1, \tag{2.1}$$

is given. Here, $x_k \in \mathbb{R}^n$ is the system state at time step k, $u_k \in \mathbb{R}^m$ is the control input at time step k, N is the number of time steps between the initial time and the final time, and $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is a continuous function that determines the state of next time step given the current state and control. If the original system is in continuous time,

time-discretization should be applied. The cost function is of the following form:

$$J(x,u) = (x_N - x^*)^\top Q_f (x_N - x^*) + \sum_{k=0}^{N-1} (x_k^\top Q x_k + u_k^\top R u_k), \tag{2.2}$$

where $x = (x_0, x_1, \ldots, x_N)^\top$, $u = (u_0, u_1, \ldots, u_{N-1})^\top$, N is the total number of time steps, which means that x_N is the final state, and x^* is the target state. The symmetric positive semi-definite matrix $Q_f \in \mathbb{R}^{n\times n}$ defines the final state cost, and the symmetric positive semi-definite matrix $Q \in \mathbb{R}^{n\times n}$ defines the intermediate state cost, while the positive definite matrix $R \in \mathbb{R}^{m\times m}$ defines the control cost. As in the LQR algorithm, the cost function is formulated in quadratic form. For the nonlinear control problem, the goal is to find control u that minimizes cost $J(x,u)$:

$$\min_{x,u} \quad J(x,u) \tag{2.3}$$

$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k), \qquad k = 0, 1, \ldots, N-1. \tag{2.4}$$

2.2 The ILQR Algorithm

The ILQR algorithm provides a suitable solution for the nonlinear control problem in equations (2.3)-(2.4). The algorithm is iterative, and can be set to terminate at some stopping criteria, for example when the difference of cost between two consecutive iterations is small enough comparing to the cost itself, i.e, $|J_{\text{old}} - J|/J_{\text{old}} \leq \text{tol}$. We assume that we have a nominal control sequence $u_0, u_1, \ldots, u_{N-1}$ and corresponding trajectory $x_0, x_1, \ldots, x_N$ at the start of each iteration. For the first iteration we can initialize the control arbitrarily (for example set control to zero) and compute the trajectory. For the following iterations those can be obtained from the output of previous iterations. Then,

we can derive a local linearization around the nominal trajectory in the form of

$$\delta x_{k+1} = A_k \delta x_k + B_k \delta x_k \tag{2.5}$$

where δx_k and δu_k denote a small deviation from the nominal trajectory, while $A_k = D_x f(x_k, u_k)$ and $B_k = D_u f(x_k, u_k)$ are the Jacobian of f with respect to x and u, respectively.

Since we want to change the nominal trajectory to make the cost smaller, we can solve the problem with the following cost function

$$\begin{aligned} J(\delta x, \delta u) &= (x_N + \delta x_N - x^*)^\top Q_f (x_N + \delta x_N - x^*) \\ &+ \sum_{k=0}^{N-1} \Big((x_k + \delta x_k)^\top Q (x_k + \delta x_k) + (u_k + \delta u_k)^\top R (u_k + \delta u_k) \Big). \end{aligned} \tag{2.6}$$

At time step k, the Hamiltonian of the cost function is:

$$\begin{aligned} H_k(\delta x_k, \delta u_k, \delta \lambda_{k+1}) &= (x_k + \delta x_k)^\top Q (x_k + \delta x_k) + (u_k + \delta u_k)^\top R (u_k + \delta u_k) \\ &+ \delta \lambda_{k+1}^\top (A_k \delta x_k + B_k \delta u_k), \end{aligned} \tag{2.7}$$

where $\delta \lambda_{k+1}^\top$ is the Lagrange multiplier. Therefore, the costate equation is

$$\delta \lambda_k = A_k^\top \delta \lambda_{k+1} + Q(\delta x_k + x_k), \tag{2.8}$$

and the stationary condition is

$$0 = R(u_k + \delta u_k) + B_k^\top \delta \lambda_{k+1}, \tag{2.9}$$

while the boundary condition is

$$\delta\lambda_N = Q_f(x_N + \delta x_N - x^*). \tag{2.10}$$

The optimal control improvement δu_k can be obtained by solving the state equation (2.5) and the costate equation (2.7), together with the stationary condition (2.9) and the boundary condition (2.10). We combine these equations and conditions by first solving (2.9), which gives

$$\delta u_k = -R^{-1}B_k^\top \delta\lambda_{k+1} - u_k, \tag{2.11}$$

substituting this equation into (2.5), and combining it with (2.7), we get the following system:

$$\begin{pmatrix} \delta x_{k+1} \\ \delta\lambda_k \end{pmatrix} = \begin{pmatrix} A_k & -B_kR^{-1}B_k^\top \\ Q & A_k^\top \end{pmatrix} \begin{pmatrix} \delta x_k \\ \delta\lambda_{k+1} \end{pmatrix} + \begin{pmatrix} -B_ku_k \\ Qx_k \end{pmatrix}. \tag{2.12}$$

Then, we assume based on the boundary condition that

$$\delta\lambda_k = S_k\delta x_k + v_k, \tag{2.13}$$

for some unknown sequences S_k and v_k. Substituting it into (2.5) yields

$$\delta x_{k+1} = (I + B_kR^{-1}B_k^\top S_{k+1})^{-1}(A_k\delta x_k - B_kR^{-1}B_k^\top v_{k+1} - B_ku_k), \tag{2.14}$$

which is then plugged into (2.7) to yield

$$\begin{aligned} S_k\delta x_k + v_k = {} & Q\delta x_k + A_k^\top S_{k+1}(I + B_kR^{-1}B_k^\top S_{k+1})^{-1} \\ & (A_k\delta x_k - B_kR^{-1}B_k^\top v_{k+1} - B_ku_k) + A_k^\top v_{k+1} + Qx_k. \end{aligned} \tag{2.15}$$

Applying the matrix inversion lemma $(A + BCD)^{-1} = A^{-1} - A^{-1}B(DA^{-1}B +$

$C^{-1})^{-1}DA^{-1}$, we obtain the equation for S_k and v_k:

$$
\begin{aligned}
K &= (B_k^\top S_{k+1} B_k + R)^{-1} B_k^\top S_{k+1} A_k && (2.16)\\
S_k &= A_k^\top S_{k+1}(A_k - B_k K) + Q && (2.17)\\
v_k &= (A_k - B_k K)^\top v_{k+1} - K^\top R u_k + Q x_k. && (2.18)
\end{aligned}
$$

Observe that we can compute S_k and v_k if we know S_{k+1} and v_{k+1}. In addition, based on the original boundary condition (2.10), we can deduce that $S_N = Q_f$, $v_N = Q_f(x_N - x^*)$. Therefore, we can compute the auxiliary variables S_k and v_k backward in time. Furthermore, we can combine (2.11), (2.13), and (2.14) to compute δu_k:

$$\delta u_k = -K\delta x_k - K_v v_{k+1} - K_u u_k, \tag{2.19}$$

where K is defined as the same way previously, and $K_v = (B_k^\top S_{k+1} B_k + R)^{-1} B_k^\top$, while $K_u = (B_k^\top S_{k+1} B_k + R)^{-1} R$. We can then march the control increment δu_k and the new trajectory x_k forward in time. In particular, we first set $\delta x_0 = 0$, since the initial state remains the same regardless of input. Then we use it to compute δu_0 by equation (2.19). We can then use the new state and control in the original system equation to compute the state variable of the next time step, which can then be used to compute the control increment of the next step. This process continues until we reaches the final step, producing the "optimal" control and trajectory based on the local linearizations A_k, B_k. This new trajectory can then be used to derive the next set of local linearizations in the next iteration. As mentioned previously, the algorithm continues until the trajectory converges. Algorithm 1 summarizes the complete ILQR algorithm.

Algorithm 1: The ILQR Algorithm

Input: System $f(x_k, u_k)$, initial state x_0, initial control sequence $u_0, u_1, \ldots, u_{N-1}$, number of steps N, matrices in cost function Q, Q_f, R, convergence threshold tol;

Output: Control u, number of ILQR iterations l;

$l = 0$;

$S_N = Q_f$;

$\delta x_0 = 0$;

$x_{k+1} = f(x_k, u_k)$, $\forall k = 0, 1, \ldots, N-1$;

$J_{\text{old}} = \infty$;

while $|J_{old} - J|/J_{old} > tol$ **do**

  $J_{\text{old}} = J$;

  $v_N = Q_f(x_N - x^*)$;

  $x^{\text{old}} = x$;

  $A_k = D_x f(x_k, u_k)$, $B_k = D_u f(x_k, u_k)$, $\forall k = 0, 1, \ldots, N-1$;

  for $i \leftarrow N-1$ **to** 0 **do**

    $K = (B_k^\top S_{k+1} B_k + R)^{-1} B_k^\top S_{k+1} A_k$;

    $K_v = (B_k^\top S_{k+1} B_k + R)^{-1} B_k^\top$;

    $K_u = (B_k^\top S_{k+1} B_k + R)^{-1} R$;

    $S_k = A_k^\top S_{k+1}(A_k - B_k K) + Q$;

    $v_k = (A_k - B_k K)^\top v_{k+1} - K^\top R u_k + Q x_k$;

  end

  for $i \leftarrow 0$ **to** $N-1$ **do**

    $\delta u_k = -K \delta x_k - K_v v_{k+1} - K_u u_k$;

    $u_k = u_k + \delta u_k$;

    $x_{k+1} = f(x_k, u_k)$;

    $\delta x_{k+1} = x_{k+1}^{\text{old}} - x_{k+1}$;

  end

  $l = l + 1$;

  $J = x_N^\top Q_f x_N$;

  for $i \leftarrow 0$ **to** $N-1$ **do**

    $J = J + x_i^\top Q x_i + u_i^\top R u_i$;

  end

end

2.2.1 Computational Cost of ILQR

We provide an analysis of the time complexity of the ILQR algorithm. In each iteration, Algorithm 1 completes three passes, the first for obtaining the linearizations, the second for computing K, K_v, K_u, S, and v, and the third one for computing the control increment δu and the new trajectory.

The cost of computing the local linearization in step 12 depends on how the problem is formulated. If the analytical form of the Jacobians is known, we can compute the Jacobian directly, resulting in N calls of the Jacobian functions in the whole pass. However, if the system is given as a black box and we can only compute the Jacobian numerically, then we have to march the system equation once for each dimension of state and control. In this case, we have to compute the system equation for $N(n+m)$ times in total.

In the second portion (step 13 to step 19) and third portion (step 20 to step 25) of the algorithm, the time complexity depends on the algorithm for matrix multiplication and inversion. State of art algorithms of those operations have complexities around $O(n^{2.4})$[4], so the complexity of the second and third pass is about $O(N(n^{2.4} + m^{2.4}))$.

The number of iterations of the ILQR algorithm l varies greatly. As shown in Section 2.3 and 2.4, l can be either 1 if the system is linear, or $O(N)$.

Therefore, we can conclude that the time complexity ILQR is about $O(N(n^{2.4} + m^{2.4}))$ with respect to the number of dimensions. Therefore, in the case of very high-dimensional problems, ILQR becomes computationally intractable, which motivates the application of model reduction techniques.

2.3 Linear Case

In this section, we present an analysis of the behavior of ILQR on an LTI system. First, we present a well-known result for the form of the LQR controller for discrete-time

LTI systems. This serves as the basis of comparison in our analysis.

Theorem 2.3.1. *[1] Consider an LTI system*

$$x_{k+1} = Ax_k + Bu_k. \tag{2.20}$$

The standard discrete-time LQR algorithm yields the following solution for $k = 0, 1, \ldots, N-1$*:*

$$\begin{aligned} u_k &= -F_k x_k && (2.21)\\ F_k &= (B^\top (Q + P_{k+1})B + R)^{-1} B^\top (Q + P_{k+1})A && (2.22)\\ P_k &= A^T (Q + P_{k+1})(A - BF_k) && (2.23)\\ P_N &= Q_f && (2.24) \end{aligned}$$

We next consider the ILQR Algorithm 1 when applied to a discrete-time LTI systems.

Proposition 1. *When applied on an LTI system of the form* (2.20)*, the ILQR algorithm converges in a single iteration, and the resulting control is the same as the output of standard LQR as given in Theorem 2.3.1.*

Proof. It is sufficient to prove that the second iteration yields the same output as the first iteration, which implies that it converges in a single iteration. Observe that the Jacobian is constant: $D_x(Ax_k + Bu_k) = A$, $D_u(Ax_k + Bu_k) = B$. This means that the matrices A_k, B_k in the local linearization is always A and B, regardless of k and the trajectory. We can then deduce that the linearization used in any iterations remains the same.

Let $x'_k = x_k + \delta x_k$ and $u'_k = u_k + \delta u_k$, where x_k, u_k represent the input trajectory in each iteration, $\delta x_k, \delta u_k$ represents the state and control increment computed in each iteration, and x'_k, u'_k is the output trajectory in each iteration. Substitute this into the

state equation (2.5), costate equation (2.7), stationary condition (2.9) and boundary condition (2.10), we can transform those equations to equations of x'_k, u'_k. Observe that those equations are uniquely defined by the linearization $A_k = A$ and $B_k = B$. Therefore, since the linearization of linear system always remain the same, the algorithm is solving the same set of four equations (2.5)(2.7)(2.9) (2.10) for all iterations, which means the output trajectories are also always the same. Therefore the algorithm converges in one iteration.

Furthermore, since the trajectory satisfies the stationary condition, it is locally optimal. Since the system is linear, the problem of finding the trajectory of minimal cost is a convex programming problem, with quadratic cost function and linear constraints. Therefore a locally optimal solution is also the globally optimal solution. Since the standard LQR algorithm also yields the globally optimal solution, we conclude that the ILQR algorithm yields the same output as the standard LQR algorithm on linear systems. □

2.4 Quadratic Case

In this section, we provide analysis of the behavior of the ILQR algorithm applied to two particular quadratic systems. We show that i) ILQR is not guaranteed to converge to the globally optimal solution and ii) ILQR is not guaranteed to converge in a few iterations: the number of iterations can be linear in terms of time discretization steps. Therefore we suggest that i) it is important to select a good initial trajectory, or try multiple initializations and pick the best result, and ii) it may be useful to use coarse discretization in the first few iterations, followed by fine-grained discretization later, to accelerate convergence.

2.4.1 Linear System with Quadratic Control

The first system of interest is a system that is quadratic in control:

$$x_{k+1} = Ax_k + u_k^\top Bu_k. \tag{2.25}$$

Suppose that we initialize the trajectory with no control ($u_k = 0 \forall k$). In the local linearization $B_k = D_u(Ax_k + u_k^\top B u_k) = (B + B^\top)u$. Since $u_k = 0 \forall k$, $B_k = 0 \forall k$. This means that $K = K_v = 0 \forall k$, and together with the fact that $u_k = 0 \forall k$, we can deduce that $\delta u_k = 0 \forall k$. Therefore, the output trajectory is exactly the same as the input trajectory, which means that the algorithm converges. However, it is clear that this trajectory is not the optimal one for most x^*.

We have shown that ILQR is not guaranteed to converge to the optimal trajectory for certain initializations. We suggest that it is often better to pick a random initial control than a particular one. Initialize with no control ($u_k = 0 \forall k$) is especially likely to be problematic since it may result in $B_k = 0$ for many systems as the system analyzed above. Alternatively, if computation time is not a concern, we can run ILQR multiple times, on different random initializations, and select the output trajectory with the lowest cost.

2.4.2 Quadratic System with Linear Control

The second system we analyze is a system that is quadratic in state:

$$x_{k+1} = x_k^\top A x_k + B u_k. \tag{2.26}$$

Suppose that $Q = 0$ in the cost equation (2.2), which is common in many problems when we only care about the final state. Suppose the initial state is $x_0 = 0$, and the initial control is given by $u_k = 0 \forall k$.

Proposition 2. *ILQR algorithm takes at least N iterations to converge, where N is the number of time steps.*

Proof. It is easy to observe that for the initial trajectory the system is stationary: $x_k = 0 \forall 0$. In the local linearization $A_k = D_x(x_k^\top A x_k + B u_k) = (A + A^\top)x$, $B_k = D_u(x_k^\top A x_k + B u_k) = B$. Therefore, $A_k = 0$ in all local linearizations.

We first analyze the behavior of ILQR algorithm in the first iterations. First, notice that except S_N, all of S_k is 0, since $A_k = Q = 0$. Therefore, with the exception of step $N-1$, which is the step before final step, $K = 0$, $K_v = R^{-1}B^\top$ and $K_u = I$. Therefore $(A_k - B_k K) = 0 \forall k$, which means that all v_k is 0 except v_N. Therefore, $\delta u_k = 0$ except δu_{N-1}. In particular, $\delta u_{N-1} = (B^\top Q_f B + R)^{-1} B^\top Q_f x^*$. As the result, only u_{N-1} is updated in the first iteration.

Similarly, we can find out that only u_{N-2} and u_{N-1} are updated in the second iteration, and so on. It takes N iterations before u_0 starts to get updated. Therefore, it takes at least N iterations for ILQR algorithm to converge. □

The findings of the previous theorem can be problematic since in each iteration the time complexity is $O(N)$ regarding the number of time steps. If it takes $O(N)$ iterations to converge, the total time complexity is $O(N^2)$, which is unacceptable when the nature of the problem requires fine-grained time discretization for precision. To address this problem, we suggest that it may be helpful to use coarse discretization on the first few iterations, and then use fine-grained discretization, to speed up convergence. For the system shown above, if N initially 10, then all u_k can be updated in 10 iterations, and we can then switch to $N = 1000$, for example. It can also be thought of as using the output from coarse ILQR to initialize fine-grain ILQR. We can also use ROMs to speed up the algorithm since the time need for each iteration is reduced significantly.

Chapter 3

System-Theoretic Model Reduction

We present two model reduction techniques that are based on systems theory, and therefore suitable for control problems. The BT algorithm is introduced in Section 3.1, while the LQG-BT algorithm is detailed in Section 3.2. These algorithms are designed to be applied on continuous-time systems, but we can apply them to discrete-time systems by first transforming the system from discrete-time to continous-time, using for example a Tustin transformation. While BT [2] is the gold standard for open-loop model reduction, LQG balancing [3] balances the closed-loop system. Despite the fact that both methods are designed for LTI systems, in the setting of controlled systems, we keep the system around the equilibrium/linearization point. Therefore, using these methods to obtain the projection subspaces and subsequently compute the nonlinear ROM makes more sense than in the open-loop nonlinear setting. Both methods will subsequently be used to enable ILQR for nonlinear systems.

3.1 Balanced Truncation

We introduce BT, one of the classical model reduction techniques. Consider a continuous-time LTI system

$$\dot{x} = Ax + Bu \tag{3.1}$$

$$y = Cx \tag{3.2}$$

where $x \in \mathbb{R}^n$ is the system state, $u \in \mathbb{R}^m$ is the control input, and $y \in \mathbb{R}^p$ is the output for observation. The system matrices are $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ and $C \in \mathbb{R}^{p\times n}$. Our goal is to have a ROM of (3.1)–(3.2) that has both good controllability and good observability properties (states that are uncontrollable or unobservable are not deemed important in a ROM). The controllability is characterized by its controllability Gramian P, while the observability is characterized by its observability Gramian Q, which is given as

$$P = \int_0^\infty e^{A\tau} B B^\top e^{A^\top \tau} d\tau \tag{3.3}$$

$$Q = \int_0^\infty e^{A^\top \tau} C^\top C e^{A\tau} d\tau. \tag{3.4}$$

We want to transform the system so that the controllability aspect and the observability aspect are balanced, which means that $P = Q$ in the transformed system. We can do this by applying a state transformation $x = T\tilde{x}$, the transformed system becomes

$$\dot{\tilde{x}} = \tilde{A}\tilde{x} + \tilde{B}u \tag{3.5}$$

$$y = \tilde{C}\tilde{x} \tag{3.6}$$

where $\tilde{A} = T^{-1}AT$, $\tilde{B} = T^{-1}B$ and $\tilde{C} = CT$. Meanwhile, the Gramians of the transformed system become $\tilde{P} = T^{-1}PT^{-\top}$ and $\tilde{Q} = T^\top QT$. First, we would like to find

T such that $\tilde{P} = \tilde{Q}$. In this way, this transformation can make the controllability and observability of the transformed system to be equal. Suppose that $\tilde{P} = \tilde{Q} = \Sigma$. Therefore,

$$\Sigma^2 = \tilde{P}\tilde{Q} = T^{-1}PT^{-\top}T^{\top}QT = T^{-1}PQT, \tag{3.7}$$

which means

$$(PQ)T = T\Sigma^2. \tag{3.8}$$

We can observe that T consists of the eigenvector of PQ, while Σ^2 is the the matrix of eigenvalues. Therefore, T can be obtained by computing the eigenvectors of PQ. However, recall that any eigenvector can be scaled and remain an eigenvector, which means there are infinite number of possible T. If we pick a random T out of them, then we are only guaranteeing $\tilde{P}\tilde{Q} = \Sigma^2$, but not $\tilde{P} = \tilde{Q}$. Hence, we want to first find any eigenvector matrix $\hat{T}$, and then scale it so that $\tilde{P} = \tilde{Q}$. To do so, we compute T by

$$T = \hat{T}(\hat{T}^{-1}P\hat{T}^{-\top})^{\frac{1}{4}}(\hat{T}^{\top}Q\hat{T})^{\frac{1}{4}}. \tag{3.9}$$

Hence we obtain a balanced transformation T, which can be rearranged to sort the eigenvalues of PQ in descending order. To get a BT, we divide T into left and right blocks, and discard the right block, as well as the corresponding state vector components, which correspond to small eigenvalues, implying that they are difficult to control and observe. In this way, we reduce the dimension of the model, while keeping the components that can be easily controlled and observed. Let

$$T = \left[\begin{array}{c|c} \Psi & T_t \end{array}\right] \tag{3.10}$$

$$S = T^{-1} = \left[\begin{array}{c} \Phi \\ \hline S_t \end{array}\right], \tag{3.11}$$

The transformed system equation is given by

$$\dot{z} = \Phi A \Psi z + \Phi B u \tag{3.12}$$

$$y = C \Psi z \tag{3.13}$$

where z is the state of the reduced model.

However, the algorithm discussed above still has some issues when put into use. First, the computation of the Gramians requires the integration of infinite time, which is hard to do numerically. Instead, the Gramians can be obtained by solving continous-time Lyapunov Equations, which is implemented in many toolboxes like Matlab and Scipy. Second, in the algorithm we first compute a square matrix T, and then discard part of it. This leads to huge waste of time, since we spend much computational power on entries that will eventually be discarded anyway. For example, if we want to reduce a model of 1000 dimensions to 10 dimensions in this way, 99% of the entries in T will be discarded and need not to be computed. Instead, we can compute balancing and reducing order transformation T directly. The modified algorithm, which is actually used in the following numerical studies, is provided in Algorithm 2.

Algorithm 2: The BT Algorithm, efficient implementation

Input: Matrices in system equation $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$ and $C \in \mathbb{R}^{p \times n}$, number of dimensions in reduced model ℓ;
Output: Transformation matrices T, T^{-1};
$P =$ Solution of Lyapunov equation $AP + PA^\top + BB^\top = 0$;
$Q =$ Solution of Lyapunov equation $QA + A^\top Q + C^\top C = 0$;
$R =$ Solve (Cholesky factorization) $P = RR^\top$;
$L =$ Solve $Q = LL^\top$;
$\ell = \min(\ell, \text{rank}(L), \text{rank}(R))$;
$U, \Sigma, V = \text{SVD}(L^\top R)$;
$T = R[:, 1:\ell] V[:, 1:\ell] (\Sigma[:, 1:\ell])^{-\frac{1}{2}}$;
$T^{-1} = (\Sigma[:, 1:\ell])^{-\frac{1}{2}} U[:, 1:\ell]^\top L[:, 1:\ell]^\top$;

3.2 LQG-Balanced Truncation

LQG-BT [3] is a variant of the BT algorithm, and it is particularly useful when the system is unstable and a stabilizing feedback controller is desired. The problems it deals with consist of systems of the same form of standard BT algorithm, as shown in (3.1), and are infinite-horizon. While the standard BT method does not specify or use a cost function, LQG-BT requires the cost function to have the following form

$$J(x,u) = \int_{t=0}^{\infty} (x^\top Q_f x + u^\top R u) dt,$$

where $x = x(t)$ is the system state at time t, $u = u(t)$ is the control input at time t, Q_f is the state cost matrix and R is the input cost matrix. By incorporating the cost function into the algorithm, LQG-BT can use the information to transform the system to compute a better control law for the closed-loop system.

The majority of steps in the LQG-BT algorithm are the same, except the computation of the matrices P and Q. Instead of computing as step 3 and 4 in Algorithm 2, where P and Q are obtained by solving Lyapunov equations, we solve the algebraic Riccati equations

$$AP + PA^\top - PC^\top CP + BB^\top = 0 \tag{3.14}$$

$$A^\top Q + QA - QBR^{-1}B^\top Q + C^\top C = 0. \tag{3.15}$$

The full LQG-BT algorithm is provided in Algorithm 3.

Algorithm 3: The LQG-BT Algorithm

Input: Matrices in system equation $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ and $C \in \mathbb{R}^{p\times n}$, number of dimensions in reduced model ℓ;
Output: Transformation matrices T, T^{-1};
$P =$ Solution of $AP + PA^\top - PC^\top CP + BB^\top = 0$;
$Q =$ Solution of $A^\top Q + QA - QBR^{-1}B^\top Q + C^\top C = 0$;
$R =$ Solution of $P = RR^\top$ (Cholesky factorization);
$L =$ Solution of $Q = LL^\top$;
$\ell = \min(\ell, \text{rank}(L), \text{rank}(R))$;
$U, \Sigma, V = \text{SVD}(L^\top R)$;
$T = R[:, 1:\ell]V[:, 1:\ell](\Sigma[:, 1:\ell])^{-\frac{1}{2}}$;
$T^{-1} = (\Sigma[:, 1:\ell])^{-\frac{1}{2}}U[:, 1:\ell]^\top L[:, 1:\ell]^\top$;

Chapter 4

Numerical Experiments

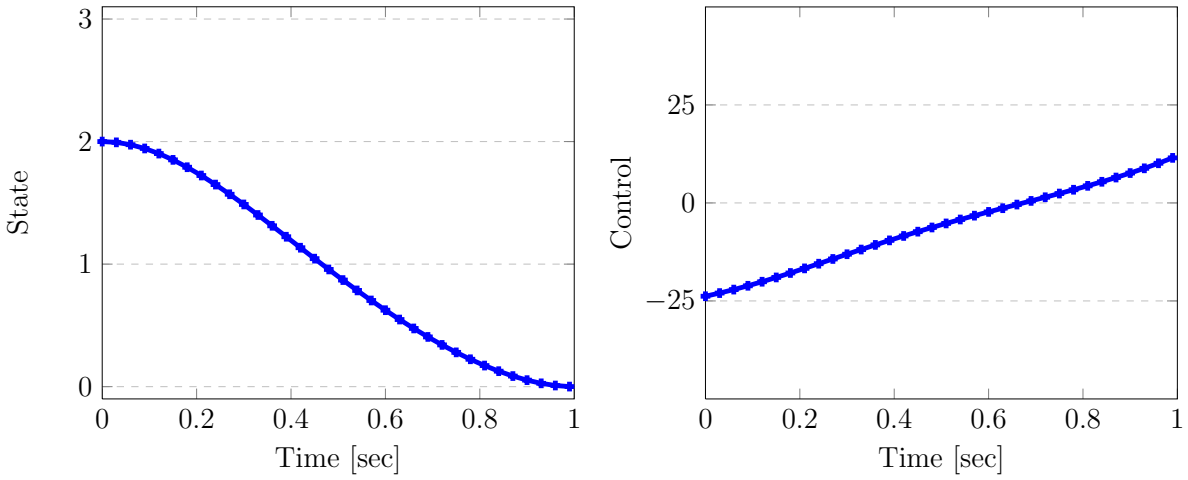
In this chapter, we provide three numerical experiments to demonstrate the capability of the ILQR algorithm and model reduction. First, in Section 4.1 and 4.2, we apply the ILQR algorithm to two nonlinear systems that are low-dimensional, where model reductions are not needed. These are also two of the models used as examples in [13]. Then, we apply a combined approach of model reduction and the ILQR algorithm to control a system of Burgers equation in Section 4.3.

4.1 Inverted Pendulum

The inverted Pendulum is one of the classical nonlinear systems studied in control theory. It consists of a simple pendulum, on which forces can be applied either clockwise or counter-clockwise, and the goal is to drive the pendulum from the initial position to the unstable state of pointing upward and stop there. The state of the system is defined by its angular position and the angular velocity. The control is defined by the one-dimensional torque applied to the pendulum. Therefore, the system can be formulated as

$$\dot{x}_1 = x_2 \tag{4.1}$$

$$\dot{x}_2 = \frac{g}{l}\sin x_1 - \frac{\mu}{ml^2}x_2 + \frac{1}{ml^2}u \tag{4.2}$$



(a) Controlled state x_1 **(b)** Control u

Figure 4.1. Controlled state x_1 and control u given by ILQR on inverted pendulum

where (x_1, x_2) is the state of the system, u is the control, m is the mass of the bob, l is the length of the rod, g is the gravity coefficient and μ is the friction coefficient. The angular position x_1 is measured by the angle between the straight upward orientation and the orientation of the rod, going clockwise, so $x^* = 0$. The cost function associated can be formulated as a first-order system of the form

$$J(x, u) = \frac{1}{2}(x_1(T)^2 + x_2(T)^2) + \frac{1}{2}\int_0^T ru^2 dt \tag{4.3}$$

where T is the final time and r is the weight of the input cost relative to final state cost. For the following experiment, we use the parameters $T = 1$, $m = 1$, $l = 1$, $g = 9.8$, $\mu = 0.01$ and $r = 10^{-5}$, which is the same parameters chosen in [13].

The result of the experiment is plotted in Figure 4.1. We can observe in the trajectory that the state is properly controlled to zero at time T. As r is very small, the control cost is not of much concern, therefore the algorithm opts for relatively large control. Notice that the control in the second half of the simulation is in the opposite direction of the control in the first half. Since the pendulum is required to stop at the final time, control must be given to decelerate it.

4.2 2-Link Arm

In this section, we study a system of human arm moving in the horizontal plane. The arm is modeled by 2 joints representing the shoulder and the elbow. The state of the system is defined by the angular positions and angular velocities of the joints, and the control is the torque applied on either of the joints. The system can be formulated as

$$\ddot{\theta} = M(\theta)^{-1}(u - C(\theta, \dot{\theta}) - B\dot{\theta}), \tag{4.4}$$

where the terms of the model are

$$M(\theta) = \begin{pmatrix} d_1 + 2d_2\cos\theta_2 & d_3 + d_2\cos\theta_2 \\ d_3 + d_2\cos\theta_2 & d_3 \end{pmatrix} \tag{4.5}$$

$$C(\theta, \dot{\theta}) = \begin{pmatrix} -\theta_2'(2\dot{\theta}_1 + \dot{\theta}_2) \\ \dot{\theta}_1^2 \end{pmatrix} d_2 \sin\theta_2 \tag{4.6}$$

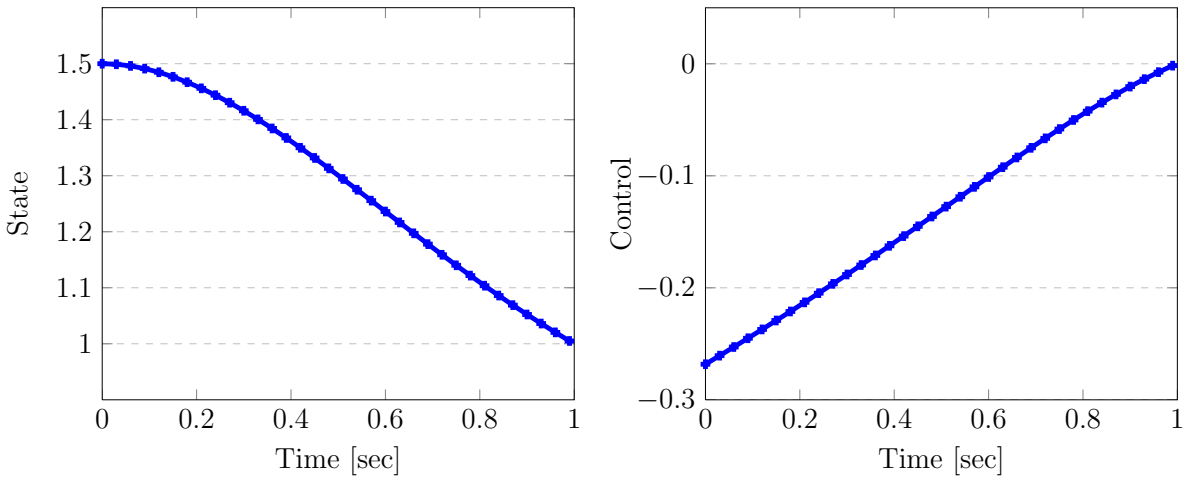
$$B = \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \tag{4.7}$$

where $x = (\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2)$ is the system state, $u = (u_1, u_2)$ is the control input, and $(d_1, d_2, d_3) = (I_1 + I_2 + m_2 l_1^2, m_2 l_1 s_2, I_2)$. The parameters are the mass of link $m_1 = 1.4, m_2 = 1.1$, the length of the link $l_1 = 0.3, l_2 = 0.33$, distance from the joint center to the link center of mass $s_1 = 0.11, s_2 = 0.16$, moment of inertia of the link $I_1 = 0.025, I_2 = 0.045$, and the joint friction matrix B defined by $b_{11} = b_{22} = 0.05, b_{12} = b_{21} = 0.025$.

The corresponding cost function is formulated as

$$J(x, u) = \frac{1}{2}((\theta_1(T) - \theta_1^*)^2 + (\theta_2(T) - \theta_2^*)^2) + \frac{1}{2}\int_0^T ru^\top u dt \tag{4.8}$$

where $T = 1$ is the final time and $r = 10^{-5}$ is the weight of the input cost relative to final



(a) Controlled x_1 **(b)** Control u_1

Figure 4.2. Controlled state x_1 and control u_1 given by ILQR on 2-link arm

state cost. These parameters are the same parameters chosen in [13]. Note that θ' is not part of the cost function, which means unlike the inverted pendulum case, the arm is not required to stop at the final time.

The result of the experiment is plotted in Figure 4.2. We can observe in the trajectory that the state is properly controlled to target state at time T. Notice that unlike the inverted pendulum simulation, the control given does not have a deceleration stage. This is the expected behavior since in the 2-link arm case the arm is not required to stop at the final time, therefore no deceleration is needed.

4.3 Burgers' Equation

As described in the introduction, Burgers' equation is a PDE that is used in modeling of fluids and traffic flows. Here we study a 1D Burgers equation with the setup as in [39]. The mathematical description of the problem is

$$\dot{z}(\xi,t) = \epsilon z_{\xi\xi}(\xi,t) - \frac{1}{2}(z^2(\xi,t))_\xi + \sum_{k=1}^{m} \chi_{[(k-1)/m,k/m]}(\xi)u_k(t) \tag{4.9}$$

for $t > 0$ and $z(\cdot, 0) = z_0(\cdot) \in H^1_{\text{per}}(0, 1)$, which means that the system has periodic boundary conditions. Here ξ is the spatial variable, $u_k(t)$ is the distributed control, and m is the dimension of control. The function $\chi_{[a,b]}(x)$ is the characteristic function over $[a, b]$. This means that the 1D space is divided into m intervals of equal length, and each control is applied on one corresponding interval.

When discretized with linear finite elements, and after an inversion with the mass matrix, the system has the form

$$\dot{x} = Ax + G(x \otimes x) + Bu \qquad x(0) = x_0 \tag{4.10}$$

$$y = Cx. \tag{4.11}$$

The operator $\otimes$ is the Kronecker product operator, and

$$x \otimes x = \begin{bmatrix} x^{(1)}x \\ \vdots \\ x^{(n)}x \end{bmatrix},$$

where $x \in \mathbb{R}^n$, and $x^{(k)}$ is the kth component of vector x. Therefore, the term $x \otimes x$ is quadratic in x. The output matrix C in equation (4.11) produces an observation of the system. We can observe the system through output y, which is related to system state x by $y = Cx$. The semi-discretized system is a nonlinear problem which is quadratic in the state, and linear in the control.

We experiment on a system with $n = 101$ states and $m = 5$ controls. The viscosity of the system, as defined in (4.9), is set to $\epsilon = 5 \times 10^{-4}$ to make the nonlinear quadratic term dominant. The system is simulated for $T = 5$ seconds, and the time interval is divided into $N = 500$ time steps. The trajectory is computed using the backward Euler method, which is implemented using the Levenberg-Marquardt algorithm. The output

matrix is $C = I_n$, the state cost matrices are $Q = Q_f = C^\top C$, and the input cost matrix is $R = 10^3 \times I_m$, where I_n is the $n \times n$ identity matrix and I_m is the $m \times m$ identity matrix.

4.3.1 Reduced-Order Models

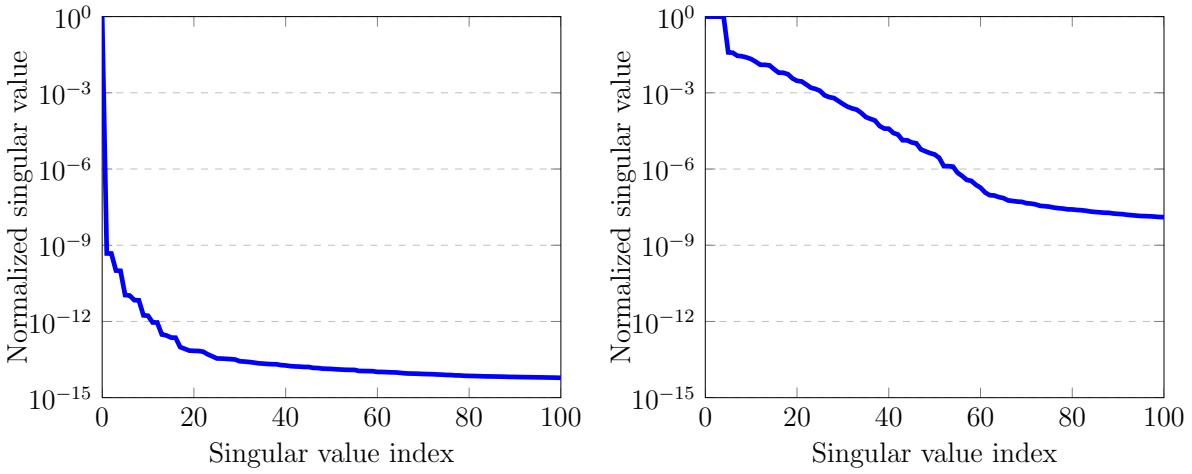
We first perform BT on the system. Since this is a nonlinear problem, we ignore the quadratic part and use matrices A, B, C to compute the balancing transformation T by Algorithm 2. Then, the reduced model of the system is

$$\dot{x}_r = A_r x_r + G_r(x_r \otimes x_r) + B_r u_r \qquad x_r(0) = T^{-1} x_0. \tag{4.12}$$

$$y_r = C_r x_r \tag{4.13}$$

where $A_r = T^{-1}AT$, $B_r = T^{-1}B$, $C_r = CT$ and $N_r = T^{-1}N(T \otimes T)$. The ROM cost is represented by $J(x_r, u_r)$, while $J(x, u_r)$ is the FOM cost with ROM controller. The Jacobians of (4.12), which are used to obtain the local linearization of the system in the ILQR algorithm, are $D_{x_r}\dot{x}_r = A_r + G_r(x_r \otimes I_{n_r} + I_{n_r} \otimes x_r)$ and $D_{u_r}\dot{x}_r = B_r$, where n_r is the number of dimensions of the state x_r in the ROM, and I_{n_r} is the $n_r \times n_r$ identity matrix. The availability of the Jacobians in explicit form significantly speeds up the computational routine, and should be used whenever available.

We first plot normalized singular values of the matrix $L^\top R$ in the balancing procedure (see step 8 in Algorithm 2 and Algorithm 3), which represents the relative importance of each dimension. For the standard BT algorithm, those are also known as the Hankel singular values, shown in Figure 4.3(a). We can observe that the first 20 Hankel singular values decay fast, and all Hankel singular values except the first one come in pairs. The normalized singular values in the LQG-BT algorithm are plotted in Figure 4.3(b). We can observe that these singular values decay much more slowly than in the model transformed by BT. Similar to the Hankel singular values, these singular values except the first one



(a) Standard BT **(b)** LQG-BT

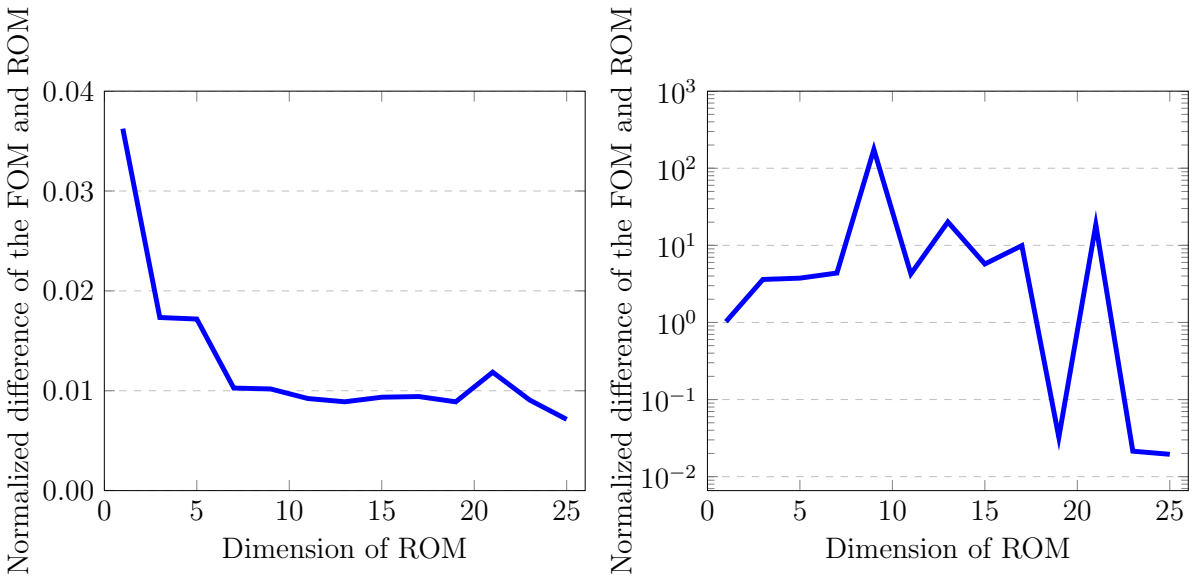
Figure 4.3. Normalized singular values of $L^\top R$

come in pairs. The fact that the singular values in the BT algorithm decay much faster indicates that it is likely that BT outperforms LQG-BT in open-loop simulation accuracy, as when we truncate the transform matrix we are discarding dimensions that have lower associated singular values.

4.3.2 Comparison of ROMs in Open-Loop Setting

We present open-loop simulations, plotted in Figure 4.4. The open-loop controls chosen for this experiment are sine functions in time, with different periods for each dimension of control: $u_k^{(i)} = 0.1\sin(it_k)$, where $u_k^{(i)}$ is the ith component of control at time step k, and t_k is the time at time step k. Recall, that we are using $m = 5$ controls. The orders of the ROMs are set to odd numbers between 1 and 25 so that each pair of singular values are either kept together or truncated together. We can observe that in the case of BT, the normalized difference of the outputs decreases as the order of ROM increases, but the gain is not very significant beyond 7. In contrast, the difference of the outputs in the case of LQG-BT is much larger, especially in ROMs with an order of less than 19. In addition, we plot the full state of the FOM and the ROM of order 5, with the ROM states projected back to FOM space, in Figure 4.5 and 4.6. These states

are visually indistinguishable. We can conclude that the BT algorithm outperforms the LQG-BT algorithm based on their accuracies in open-loop simulations. However, it is worth noting that high open-loop simulation accuracy does not necessarily lead to good controller design in closed-loop simulation, as we will see next.



(a) Standard BT **(b)** LQG-BT

Figure 4.4. Normalized difference between the outputs of FOM and ROMs $\frac{\|y-y_r\|_2}{\|y\|_2}$. Note that only odd number between 1 and 25 are used as dimensions of ROMs.

4.3.3 Comparison of ROMs When Used for Closed-Loop Systems

We use the ILQR algorithm to obtain controllers based on ROMs constructed by BT or LQG-BT, denoted via u_r^{BT} and u_r^{LQGBT}, and apply these controllers on the FOM to evaluate the performances. We set the ROM dimension in both cases to $r = 5$. The convergence tolerance of ILQR algorithm is set to 3×10^{-5}.

The cost of ROM $J(x_r, u_r^{\mathrm{BT}})$ computed by BT is 77.3, while the cost of FOM $J(x, u_r^{\mathrm{BT}})$ is 165.7. The cost of ROM $J(x_r, u_r^{\mathrm{LQGBT}})$ computed by LQG-BT is 48.5, while the cost computed of FOM $J(x, u_r^{\mathrm{LQGBT}})$ is 130.9. ILQR takes 168 iterations to converge

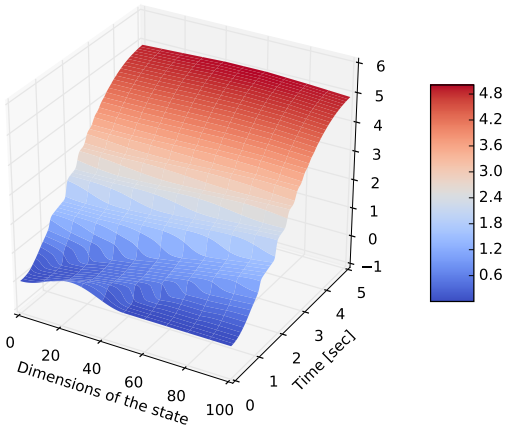
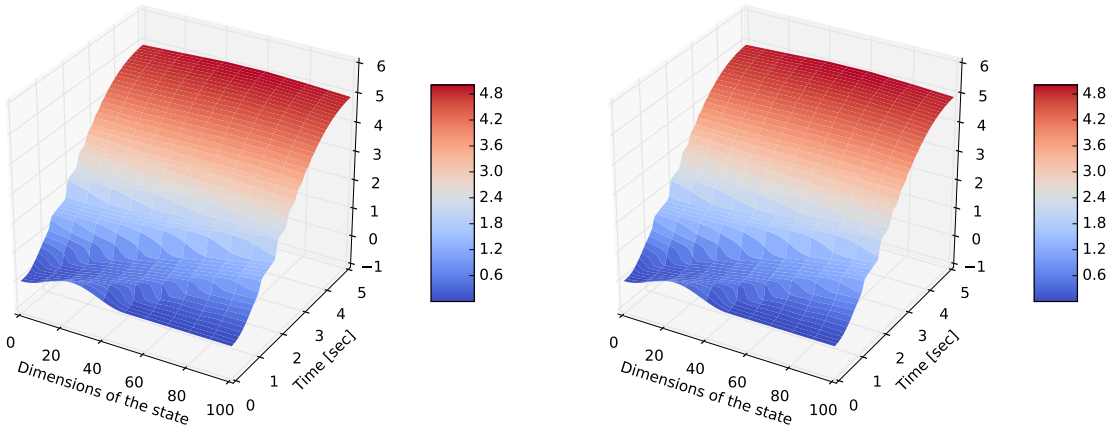


Figure 4.5. Open-loop simulation of FOM for Burgers equation with input $u_k^{(i)} = 0.1\sin(it_k)$

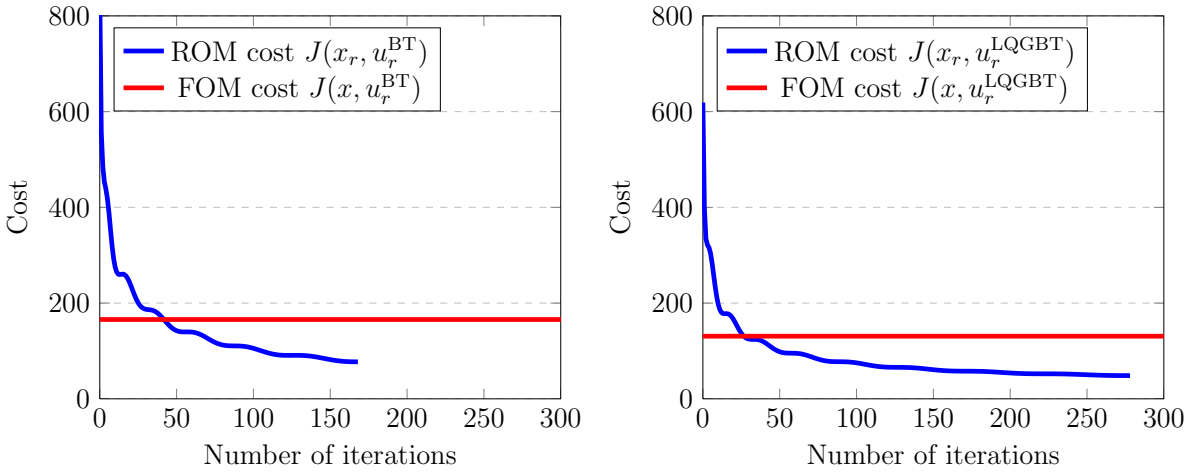
on the ROM computed by BT, and 278 iterations on the ROM computed by LQG-BT. The ROM costs of the intermediate controllers given after each iteration in ILQR are plotted in Figure 4.7, which shows that the costs consistently decrease after each iteration. Figure 4.8 shows the norm of outputs $\|y\|_2$ in the closed-loop system of FOM, with the controller given by ILQR computed using the two nonlinear ROMs. We can observe that both controllers successfully control the outputs to near zero. Despite this, we notice that in both cases, the FOM costs are between 2 to 3 times the ROM cost. These can be explained by the part of dynamics that is lost in the model reduction process. As discussed in Section 3.1, in BT and by extension, LQG-BT, we truncate the part of the dynamics that is hard to control and hard to observe, which is exactly the part of unaddressed dynamics in the ROM controller that leads to the discrepancy between the ROM cost and FOM cost. In addition, we can find out that despite BT outperforms LQG-BT in open-loop simulations, in the closed-loop system LQG-BT provides better control compared to the standard BT algorithm under the same setting, both in terms of ROM cost and FOM cost. This matches our discussion in Section 3.2 that LQG-BT is expected to provide a better controller in closed-loop simulation than BT, despite its



(a) Standard BT **(b)** LQG-BT

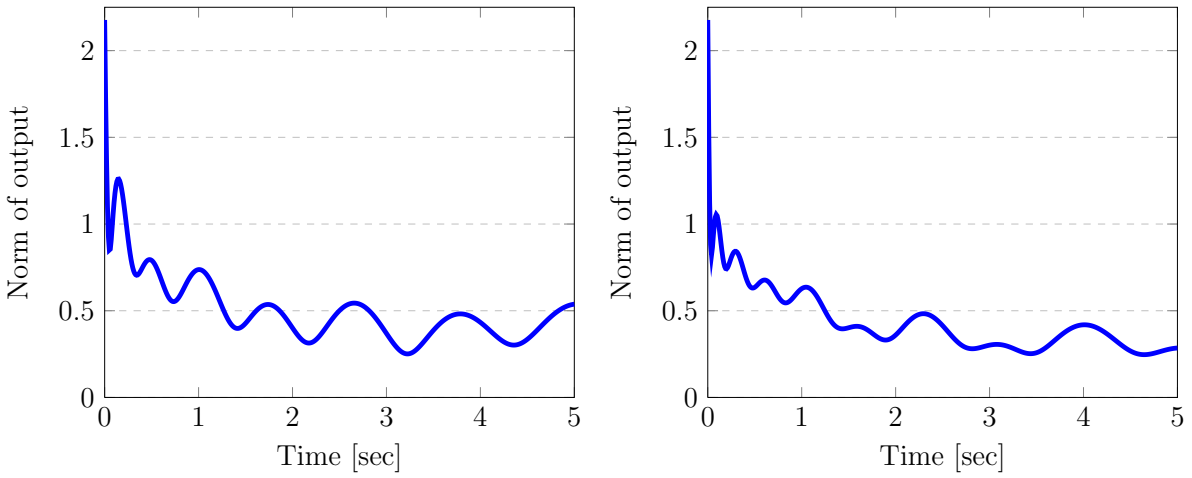
Figure 4.6. Open-loop simulation of ROM with input $u_k^{(i)} = 0.1\sin(it_k)$, with state projected back to FOM space

apparent inaccuracy in open-loop simulation.



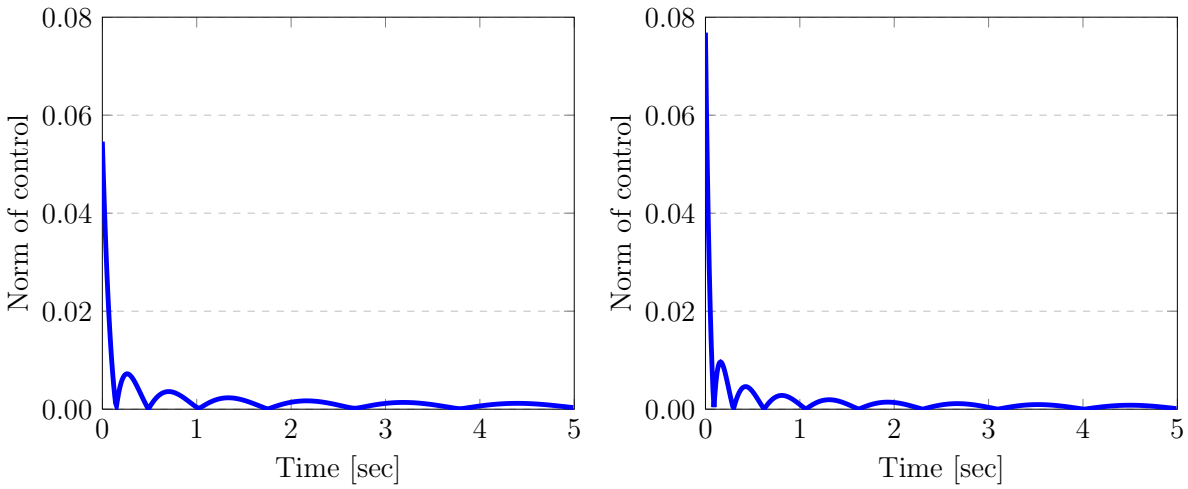
(a) standard BT **(b)** LQG-BT

Figure 4.7. ROM cost vs number of iterations in the ILQR algorithm. FOM cost on the final control included as reference



(a) standard BT **(b)** LQG-BT

Figure 4.8. Norm of outputs $\|y\|_2$ in the closed-loop system of FOM, with the controller given by ILQR computed using the two nonlinear ROMs



(a) standard BT **(b)** LQG-BT

Figure 4.9. Norm of controls $\|u\|_2$ given by ILQR computed using the two nonlinear ROMs

Chapter 5

Conclusion

High-dimensional nonlinear problems are ubiquitous in science and engineering, often resulting from semi-discretization of PDEs. They are among the most challenging problems in control theory. On one hand, using linear controllers like the LQR algorithm leads to suboptimal result on highly nonlinear systems. On the other hand, while nonlinear controllers can produce better control, on high-dimensional problems they become computationally intractable. Therefore there is the false dichotomy between choosing a controller based on linearized dynamics that is fast to compute but results in high cost and choosing a nonlinear controller that takes much computational power.

We presented a framework of combining ILQR and system-theoretic model reduction techniques to derive a controller for high-dimensional nonlinear problems. As shown in the numerical experiments in Section 4.3, both standard BT and LQG-BT result in promising controllers. Meanwhile, the order of the ROMs is only about 5% of the order of the FOM, making the algorithm computationally tractable. In addition, the framework put few constraints on the problem, as it only requires the cost function to be quadratic, making a large range of applications possible.

Further work may be done by using model reduction techniques other than BT and LQG-BT. For example, nonlinear system-theoretic model reduction, such as [35, 36, 37], can be used to obtain more accurate ROMs on nonlinear systems. In addition, trajectory-

based model reduction techniques [38] can also be incorporated to improve the accuracy of ROM using trajectory data if available.

Further work may also be done by improving the ILQR algorithm. As we have shown in Section 2.4, the ILQR algorithm is not guaranteed to converge to the optimal control, and it may take a lot of iterations to converge in extreme cases. Moreover, the scope of this framework is limited by the ILQR's assumption that the cost function is quadratic, so further works can be done to extend the ILQR algorithm to systems with more complex cost functions, in order to make the range of applications even larger.

Bibliography

[1] H. Kwakernaak and R. Sivan. *Linear Optimal Control Systems*. USA: John Wiley & Sons, Inc., 1972. ISBN: 0471511102.

[2] B. Moore. "Principal component analysis in linear systems: Controllability, observability, and model reduction". In: *IEEE Transactions on Automatic Control* 26.1 (1981), pp. 17–32.

[3] E. Jonckheere and L. Silverman. "A new set of invariants for linear systems–Application to reduced order compensator design". In: *IEEE Transactions on Automatic Control* 28.10 (1983), pp. 953–964.

[4] D. Coppersmith and S. Winograd. "Matrix multiplication via arithmetic progressions". In: *Journal of Symbolic Computation* 9.3 (1990). Computational algebraic complexity editorial, pp. 251 –280. ISSN: 0747-7171.

[5] J. A. Burns and S. Kang. "A control problem for Burgers' equation with bounded input/output". In: *Nonlinear Dynamics* 2.4 (1991), pp. 235–262.

[6] G. Berkooz, P. Holmes, and J. L. Lumley. "The proper orthogonal decomposition in the analysis of turbulent flows". In: *Annual Review of Fluid Mechanics* 25.1 (1993), pp. 539–575.

[7] H. Choi, R. Temam, P. Moin, and J. Kim. "Feedback control for unsteady flow and its application to the stochastic Burgers equation". In: *Journal of Fluid Mechanics* 253 (1993), pp. 509–543.

[8] K. Karl and V. S. "Control of the Burgers equation by a reduced-order approach using proper orthogonal decomposition". In: *Journal of Optimization Theory and Applications* 102 (1999), 345–371.

[9] J. Baker, A. Armaou, and P. D. Christofides. "Nonlinear control of incompressible fluid flow: Application to Burgers' equation and 2D channel flow". In: *Journal of Mathematical Analysis and Applications* 252.1 (2000), pp. 230–255.

[10] A. Balogh and M. Krstić. "Burgers' equation with nonlinear boundary feedback: H1 stability, well-posedness and simulation". In: *Mathematical Problems in Engineering* 6.2-3 (2000), pp. 189–200.

[11] J. A. Atwell, J. T. Borggaard, and B. B. King. "Reduced order controllers for Burgers' equation with a nonlinear observer". In: *International Journal of Applied Mathematics and Computer Science* 11 (2001), pp. 1311–1330.

[12] W.-J. Liu and M. Krstić. "Adaptive control of Burgers' equation with unknown viscosity". In: *International Journal of Adaptive Control and Signal Processing* 15.7 (2001), pp. 745–766.

[13] W. Li and E. Todorov. "Iterative linear quadratic regulator design for nonlinear biological movement systems." In: *International Conference on Informatics in Control, Automation and Robotics*. 2004, pp. 222–229.

[14] J. C. de los Reyes and K. Kunisch. "A comparison of algorithms for control constrained optimal control of the Burgers equation". In: *Calcolo* 41.4 (2004), pp. 203–225.

[15] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. SIAM, 2005. ISBN: 9780898715293.

[16] G. Kerschen, J.-c. Golinval, A. F. Vakakis, and L. A. Bergman. "The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: an overview". In: *Nonlinear Dynamics* 41.1-3 (2005), pp. 147–169.

[17] K. Kunisch and L. Xie. "POD-based feedback control of the Burgers equation by solving the evolutionary HJB equation". In: *Computers & Mathematics with Applications* 49.7-8 (2005), pp. 1113–1126.

[18] E. Todorov and W. Li. "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems". In: *Proceedings of the 2005, American Control Conference, 2005.* IEEE. 2005, pp. 300–306.

[19] B. B. King, N. Hovakimyan, K. A. Evans, and M. Buhl. "Reduced order controllers for distributed parameter systems: LQG balanced truncation and an adaptive approach". In: *Mathematical and computer modelling* 43.9-10 (2006), pp. 1136–1149.

[20] J. R. Singler and B. A. Batten. "A comparison of balanced truncation methods for closed loop systems". In: *2009 American Control Conference*. IEEE. 2009, pp. 820–825.

[21] B. A. Batten and K. A. Evans. "Reduced-order compensators via balancing and central control design for a structural control problem". In: *International Journal of Control* 83.3 (2010), pp. 563–574.

[22] M. M. Baumann. "Nonlinear model order reduction using POD/DEIM for optimal control of Burgers equation". MA thesis. Delft University of Technology, Netherlands, 2013.

[23] J. van den Berg. "Iterated LQR smoothing for locally-optimal feedback control of systems with non-linear dynamics and non-quadratic cost". In: *2014 American Control Conference*. IEEE. 2014, pp. 1912–1918.

[24] B. Kramer. "Solving algebraic Riccati equations via proper orthogonal decomposition". In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 7767–7772.

[25] A. Quarteroni and G. Rozza. *Reduced order methods for modeling and computational reduction*. Springer, 2014. ISBN: 9783319020914.

[26] T. Breiten and K. Kunisch. "Feedback stabilization of the Schlögl model by LQG-balanced truncation". In: *2015 European Control Conference (ECC)*. IEEE. 2015, pp. 1171–1176.

[27] Y. Alothman and D. Gu. "Quadrotor transporting cable-suspended load using iterative linear quadratic regulator (ILQR) optimal control". In: *2016 8th Computer Science and Electronic Engineering (CEEC)*. IEEE. 2016, pp. 168–173.

[28] B. Kramer. "Model reduction for control of a multiphysics system: Coupled Burgers' equation". In: *2016 American Control Conference (ACC)*. IEEE. 2016, pp. 6146–6151.

[29] B. Kramer and J. R. Singler. "A POD projection method for large-scale algebraic Riccati equations". In: *Numerical Algebra, Control & Optimization* 6 (2016), pp. 413–435.

[30] J. Sergey, S. Sergei, and Y. Andrey. "Comparative analysis of iterative LQR and adaptive PD controllers for a lower limb exoskeleton". In: *2016 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE. 2016, pp. 239–244.

[31] Y. Alothman, M. Guo, and D. Gu. "Using iterative LQR to control two quadrotors transporting a cable-suspended load". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 4324–4329.

[32] P. Benner, M. Ohlberger, A. Cohen, and K. Willcox. *Model Reduction and Approximation*. Society for Industrial and Applied Mathematics, 2017. ISBN: 9781611974812.

[33] J. Chen, W. Zhan, and M. Tomizuka. "Constrained iterative LQR for on-road autonomous driving motion planning". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, pp. 1–7.

[34] E. Banijamali, R. Shu, H. Bui, A. Ghodsi, et al. "Robust locally-linear controllable embedding". In: *International Conference on Artificial Intelligence and Statistics*. 2018, pp. 1751–1759.

[35] P. Benner, P. Goyal, and S. Gugercin. "$\mathcal{H}_2$-quasi-optimal model order reduction for quadratic-bilinear control systems". In: *SIAM Journal on Matrix Analysis and Applications* 39.2 (2018), pp. 983–1032.

[36] A. C. Antoulas, I. V. Gosea, and M. Heinkenschloss. "On the Loewner framework for model reduction of Burgers equation". In: *Active Flow and Combustion Control 2018*. Springer, 2019, pp. 255–270.

[37] B. Kramer and K. Willcox. "Balanced truncation model reduction for lifted nonlinear systems". In: *AIAA Journal* 57.6 (2019), pp. 2297–2307.

[38] B. Kramer and K. Willcox. "Nonlinear Model Order Reduction via Lifting Transformations and Proper Orthogonal Decomposition". In: *AIAA Journal* 57.6 (2019), pp. 2297–2307.

[39] J. Borggaard and L. Zietsman. "The quadratic-quadratic regulator problem: approximating feedback controls for quadratic-in-state nonlinear systems". In: *2020 American Control Conference (ACC)*. IEEE. 2020, pp. 818–823.