Learning-Based Pareto Optimal Control of Large-Scale Systems with Unknown Slow Dynamics

Saeed Tajik Hesarkuchak

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Almuatazbellah (Muataz) Boker, Chair
Lamine Mili
Hoda Eldardiry

April 25, 2024
Blacksburg, Virginia

Keywords: Optimal Control, Data-Driven Control, Large Scale systems.

Learning-Based Pareto Optimal Control of Large-Scale Systems with Unknown Slow Dynamics

Saeed Tajik Hesarkuchak

(ABSTRACT)

We develop a data-driven approach to Pareto optimal control of large-scale systems, where decision makers know only their local dynamics. Using reinforcement learning, we design a control strategy that optimally balances multiple objectives. The proposed method achieves near-optimal performance and scales well with the total dimension of the system. Experimental results demonstrate the effectiveness of our approach in managing multi-area power systems.

Learning-Based Pareto Optimal Control of Large-Scale Systems with Unknown Slow Dynamics

Saeed Tajik Hesarkuchak

(GENERAL AUDIENCE ABSTRACT)

We have developed a new way to manage complex systems—like power networks—where each part only knows about its own behavior. By using a type of artificial intelligence known as reinforcement learning, we've designed a method that can handle multiple goals at once, ensuring that the entire system remains stable and works efficiently, no matter how large it gets. Our tests show that this method is particularly effective in coordinating different sections of power systems to work together smoothly. This could lead to more efficient and reliable power distribution in large networks.

Dedication

This thesis is lovingly dedicated to my family—my incredible brother, my cherished sister, and my beloved parents. Their unwavering support, endless encouragement, and profound love have been the cornerstone of my journey. I am eternally grateful for every sacrifice they have made and for believing in me unconditionally. To my friends, who have been my extended family, thank you for being my constant source of joy and companionship.

Acknowledgments

I extend my deepest gratitude to my advisor, Dr. Muataz Boker, whose guidance and support were instrumental in the development of this research. His expertise and mentorship profoundly shaped this work, teaching me not only about academic rigor but also the importance of patience and persistence. I am also grateful to everyone who supported my academic journey; this thesis would not have been possible without their collective contributions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Large-scale systems, which encompass a wide range of applications from power grids to autonomous vehicles, pose unique challenges to control engineers. These systems often exhibit complex interactions, nonlinearities, and uncertainties, making them difficult to model accurately. Traditional control strategies relying on well-defined models struggle to address these challenges effectively. One of the traditional methods to tackle such challenging problems is Pareto optimality. The pursuit of Pareto optimality in control represents a significant advancement in the field, as it seeks to simultaneously optimize multiple, often conflicting, objectives. This concept is particularly relevant in large-scale systems, where control actions must balance objectives such as stability, performance, and energy efficiency.

Previous work [6] introduced adaptive control laws through which we can learn the control of the model from the input-output data. Over the years, the increase in the momentum of reinforcement learning [16] has opened up the different possibilities of learning the controller. For example [4] enhances off-policy natural actor-critic algorithms by establishing improved finite-sample convergence guarantees and [12] addresses the challenge of high variance in deep reinforcement learning algorithms focusing on enhancing the robustness of general actor-critic methods. Adaptive Dynamic Programming (ADP) is among the earliest learning techniques in reinforcement learning, employing an iterative approach with variable adjustments to identify the optimal control gain [9]. Typically, learning methods like Q-learning [17] and Actor-Critic [18, 20, 21] have shown efficacy in developing controllers

for continuous-time systems where the model dynamics are not known. Additionally, the previous works [17], [18] mostly focus on learning the dynamics of the system using offline data. This approach does not cope well if exogenous disturbances are affecting the system. To tackle this issue, [19] proposes to train the closed-loop model on the running data in an online fashion. However, in these cases, they provide an insight on how to optimize the entire system which is a challenging task for large-scale systems. Also, [11] focuses on robust control against disturbances for nonlinear systems with unknown dynamics, using off-policy reinforcement learning.

In an attempt to develop scalable optimal control algorithms, the work in [13] employs a singular perturbation method to decompose the system based on time scales. In this case, the controller is learned for the dominant slow-time-scale dynamics, assuming that the fast-time-scale dynamics are asymptotically stable and, hence, ignored. This resulted in lower dimensional and computationally efficient control systems.

1.1 Contributions

In this work, we follow the same spirit as [13] and solve a Pareto optimal control problem but with the relaxation of the assumption that the fast dynamics are asymptotically stable. In this way, we enlarge the class of applicable systems and simultaneously keep track of how the slow- and fast-time scale decomposition affects the implementation of the controller based on feedback from the system's original data. Moreover, our proposed approach is based on the multi-area modeling scheme pioneered in [8], where the decision-maker of each area utilizes a simplified model of the whole system. However, relative to [8], we employ a learning-based approach to solve the control problem when the coupled slow dynamics of all areas are unknown. Overall, The contribution of the paper can be summarized as follows: First,

we propose a systematic design procedure for the control of large-scale systems when the global coupled dynamics are not known while the local dynamics may be unstable. Second, the proposed strategy is scalable in the sense that the computational complexity and size of the local controllers are not affected by the large size of the overall system. Third, we follow a singular and regular perturbation approach to show that the closed-loop performance of the proposed strategy approaches that of a centralized model-based optimal controller as the time-scale difference among the system dynamics becomes stark and coupling among the different local dynamics becomes weak.

1.2 Problem Formulation

Consider a large-scale dynamic system with N interconnected areas, modeled in the form

$$\dot{x} = A_o x + \sum_{j=1}^{N} A_{oj} z_j + \sum_{j=1}^{N} B_{oj} u_j, \; x(0) = x_o, \tag{1.1}$$

$$\varepsilon_i \dot{z}_i = A_{io} x + A_{ii} z_i + \sum_{j \neq i} \varepsilon_{ij} A_{ij} z_j + B_{ii} u_i, \; z_i(0) = z_{io}, \tag{1.2}$$

where $x \in \mathbb{R}^{n_o}$, $z_i \in \mathbb{R}^{n_i}$ for $i = 1, \ldots, N$, are state vectors, A and B are constant matrices of appropriate dimensions and $u_i \in \mathbb{R}^{m_i}$ is the input. $0 < \varepsilon_i < 1$ is a small (unknown) parameter that represents the time constants for each subsystem, while ε_{ij} is a small (unknown) parameter that represents a weak coupling between the subsystems.

The system (1.1)-(1.2) has strongly coupled slow (x) dynamics and weakly coupled fast (z_i) dynamics, with every area has a decision maker u_i. This model represents many systems, such as power systems [22] and clustered networks [2], [3]. In this work, we assume that each decision-maker has a full knowledge of the local dynamics but does not know the global dynamics. More specifically, we assume that the matrices A_o, A_{oj}, B_{oj} and A_{ij} are unknown.

Consider the case that decision-makers cooperate to minimize a global cost function

$$J = \gamma_1 J_1 + \cdots + \gamma_k J_k + \cdots + \gamma_N J_N, \tag{1.3}$$

where $\gamma_1 + \cdots + \gamma_N = 1,\ 0 < \gamma_i < 1$, for $i = 1, \ldots, N$,

$$\begin{aligned} J_k =& \frac{1}{2}\int_0^\infty (y_k' Q_k y_k + u_k' R_k u_k)\, dt, \\ y_k =& C_{ok} x + C_{kk} z_k,\ Q_k = C_k' C_k,\ R_k > 0. \end{aligned} \tag{1.4}$$

Therefore, a controller u_k is sought for each area k such that (1.3) is minimized under the assumption that the shares γ_i of all other controllers are known a-priori and using feedback from the slow (global) state x and fast (local) state z_k.

The optimal strategy for (1.1)-(1.2) can be obtained had all the system matrices are known. It takes the form [1]

$$u_k^* = -\frac{1}{\gamma_k} R_k^{-1} B_k' P^* \hat{x}, \tag{1.5}$$

where

$$P^* A + A' P^* + Q - P^* S P^* = 0, \tag{1.6}$$

$$A = \begin{bmatrix} A_o & A_{o1} & A_{o2} & \ldots \\ \frac{A_{1o}}{\varepsilon_1} & \frac{A_{11}}{\varepsilon_1} & \frac{\varepsilon_{12} A_{12}}{\varepsilon_1} & \ldots \\ \frac{A_{2o}}{\varepsilon_2} & \frac{\varepsilon_{21} A_{21}}{\varepsilon_2} & \frac{A_{22}}{\varepsilon_2} & \ldots \\ \vdots & \vdots & \vdots & \end{bmatrix}, \qquad \hat{x} = \begin{bmatrix} x \\ z_1 \\ \vdots \\ z_N \end{bmatrix},$$

$$B_1 = \begin{bmatrix} B_{o1} \\ \frac{B_{11}}{\varepsilon_1} \\ 0 \\ \vdots \end{bmatrix}, \ldots, B_N = \begin{bmatrix} B_{oN} \\ 0 \\ \vdots \\ \frac{B_{NN}}{\varepsilon_N} \end{bmatrix}, \quad S = \frac{1}{\gamma_1} S_1 + \frac{1}{\gamma_2} S_2 + \ldots,$$

$$Q = \gamma_1 Q_1 + \gamma_2 Q_2 + \ldots, \quad S_k = B_k R_k^{-1} B_k'.$$

The optimal controller (1.5) for each area requires knowledge of the entire system. In the sequel, we will follow a singular perturbation method to relax this requirement and achieve a performance close to the optimal one.

Chapter 2

Controller Design

2.1 Singular Perturbation-Based Design

For a linear system consisting of k control areas, a simplified model of the entire system can be obtained from the perspective of the decision maker of the k^{th} subsystem. For this purpose, assume that the decision maker of the k^{th} subsystem ignores all other fast (local) dynamics and weak coupling with other subsystems. This implies setting $\varepsilon_j = 0$ for $j \neq k$ and $\varepsilon_{ij} = 0$ in (1.1)-(1.2). This, in turn, leads to the k^{th} subsystem approximate model

$$\dot{x}_k = A_k x_k + A_{ok} z_k + B_{ok} u_k + \sum_{j \neq k} B_{kj} u_j, \tag{2.1}$$

$$\varepsilon_k \dot{z}_k = A_{ko} x_k + A_{kk} z_k + B_{kk} u_k, \tag{2.2}$$

where

$$A_k = A_o - \sum_{j \neq k} A_{oj} A_{jj}^{-} 1 A_{jo}, \qquad B_{kj} = B_{oj} - Aoj A_{jj}^{-} 1 B_{jj}.$$

Assumption 1. The matrix A_{ii} is invertible.

Under Assumption 1, system (2.1)-(2.2) is in the standard singular perturbation form. This allows the use of the singular perturbation method to design a control strategy for each decision maker by solving two separate sub-problems; one in the slow time scale and one in

a fast time scale [10]. By setting $\varepsilon_k = 0$ in (2.1)-(2.2), we get the reduced order model

$$\dot{x}_s = A_s x_s + \sum_{k=1}^{N} B_{ks} u_{ks}, \quad x_s(0) = x_o, \tag{2.3}$$

with $z_{ks} = -A_{kk}^{-1}(A_{ko}x_s + B_{kk}u_{ks})$ and where $A_s = A_o - \sum_{i=1}^{N} A_{oi}A_{ii}^{-1}A_{io}$, $B_{ks} = B_{ok} - A_{ok}A_{kk}^{-1}B_{kk}$ and u_{ks} is the slow controller. The boundary layer model can be obtained by using the change of variables $z_{kf} = z_{ks} + A_{kk}^{-1}(A_{ko}x_s + B_{kk}u_{ks})$ and letting $\tau_k = t/\varepsilon_k$ to get

$$\frac{dz_{kf}}{d\tau_k} = A_{kk}z_{kf} + B_{kk}u_{kf}, \tag{2.4}$$

with u_{kf} being the fast control.

Assumption 2. The pairs (A_s, B_s), where $B_s = (B_{1s}, \ldots, B_{Ns})$, and (A_{kk}, B_{kk}) are stabilzable.

Following this approach, each decision maker can solve for the slow controller $u_{ks} =: G_{ks}x_s$ using (2.3) and fast controller $u_{kf} =: G_{kf}z_{kf}$ using (2.4), where G_{ks} and G_{kf} are controller matrices to be designed. The implementable controller can then be taken as the composite of these two controls, which takes the form

$$u_k = [(I + G_{kf}A_{kk}^{-1}B_{kk})G_{ks} + G_{kf}A_{kk}^{-1}A_{ko}]x + G_{kf}z_k. \tag{2.5}$$

Notice that all area controllers affect (2.3), whereas (2.4) indicates that the fast control problem is decoupled.

The following theorem adopted from [7] describes the behavior of the slow (2.3) and fast (2.4) subsystems compared to the actual system (1.1)-(1.2) under feedback control.

Theorem 2.1. *Let Assumptions 1 and 2 hold. If $u_{kf} = G_{kf}z_{kf}$ are designed to stabilize the*

subsystem (2.4) *and* $u_{ks} =: G_{ks}x_s$ *are designed to stabilize the subsystem* (2.3) *then there exists a positive scalar* σ *such that* $\forall\ 0 < ||\varepsilon|| \leq \sigma$ *and* $\forall\ t \geq 0$ *we have*

$$\begin{aligned} x(t) &= x_s(t) + O(\|\varepsilon\|), \\ z_1(t) &= -A_{11}^{-1}(A_{1o} + B_{11}G_{1s})x_s(t) + z_{1f}(t/\varepsilon_1) + O(\|\varepsilon\|), \\ &\vdots \\ z_N(t) &= -A_{NN}^{-1}(A_{No} + B_{NN}G_{Ns})x_s(t) + z_{Nf}(t/\varepsilon_N) + O(\|\varepsilon\|). \end{aligned} \tag{2.6}$$

where ε *is an ordered vector of all the parameters* ε_i *and* ε_{ij} *with* $i \neq j$ *and* $i = 1, \ldots, N$.

In the following, we will follow a data-driven approach to solve the Pareto optimal control of the slow subsystem and combine that with a stabilizing controller for the fast subsystem. We will show that this composite controller will achieve a performance that converges to the one obtained had we known the coupled models.

Notation Definitions: The notation $(\cdot)^*$ denotes the optimal values of the parameters. Parameters denoted by $(\cdot)$ are those derived from the singular perturbation-based design. We use $\tilde{(\cdot)}$ to represent parameters that are estimated based on learning from the x_s (assuming we have access to it), and $\bar{(\cdot)}$ to denote parameters learned using the actual x data.

2.2 Learning-Based Design

To solve the control problem, we will follow the singular perturbation method discussed in the previous section. For the slow sub-problem and given (2.3), decision-makers need to cooperate to optimize the cost function

$$J_s = \gamma_1 J_{1s} + \gamma_2 J_{2s} + \ldots + +\gamma_N J_{Ns}, \tag{2.7}$$

where

$$\begin{aligned} J_{ks} &= \frac{1}{2}\int_0^\infty (x_s' C_{ks}' C_{ks} x_s + 2u_{ks}' D_{ks}' C_{ks} x_s + u_{ks}' R_{ks} u_{ks}) dt, \\ C_{ks} &= C_{ok} - C_{kk} A_{kk}^{-1} A_{ko}, \\ D_{ks} &= -C_{kk} A_{kk}^{-1} B_{kk}, \\ R_{ks} &= R_k + D_{ks}' D_{ks}. \end{aligned} \tag{2.8}$$

The solution of this problem is given as

$$u_{ks} = -R_{ks}^{-1}(D_{ks}' C_{ks} + \frac{1}{\gamma_k} B_{ks}' P_s) x_s, \tag{2.9}$$

with P_s is the positive semidefinite stabilizing solution of the Riccati equation

$$P_s \hat{A}_s + \hat{A}_s' P_s + \sum_{i=1}^{N} [\frac{-1}{\gamma_i} P_s B_{is} R_{is}^{-1} B_{is}' P_s + \gamma_i C_{is}' (I - D_{is} R_{is}^{-1} D_{is}') C_{is}] = 0, \tag{2.10}$$

which can be written in the compact form

$$P_s \hat{A}_s + \hat{A}_s' P_s - P_s B_s R_s^{-1} B_s' P_s + Q_s = 0, \tag{2.11}$$

where

$$\begin{aligned} \hat{A}_s &= A_s - \sum_{i=1}^{k} B_{is} R_{is}^{-1} D_{is}' C_{is}, \\ B_s &= [B_{1s} \quad B_{2s} \quad ... \quad B_{ks}], \\ Q_s &= \sum_{i=1}^{k} \gamma_i C_{is}' (I - D_{is} R_{is}^{-1} D_{is}') C_{is}, \\ R_s &= \text{diag}(R_{1s}\gamma_1, R_{2s}\gamma_2, ..., R_{ks}\gamma_i). \end{aligned} \tag{2.12}$$

The reduced order model (2.3) can also be written in this form as

$$\dot{x}_s = \hat{A}_s x_s + B_s U, \qquad U = u_s + M x_s, \tag{2.13}$$

where $\hat{A}_s = A_s - B_s M$, and

$$M = \begin{bmatrix} R_{1s}^{-1} D_{1s}' C_{1s} \\ R_{2s}^{-1} D_{2s}' C_{2s} \\ \cdots \\ R_{Ns}^{-1} D_{Ns}' C_{Ns} \end{bmatrix}, u_s = \begin{bmatrix} u_{1s} \\ u_{2s} \\ \cdots \\ u_{Ns} \end{bmatrix}. \tag{2.14}$$

We now have the following assumption.

Assumption 3. The pairs (A_s, C_s), where $C_s = (C_{1s}, \ldots, C_{Ns})$, and (A_{kk}, C_{kk}) are detectable.

For the fast subproblem (2.4), we consider the cost function

$$J_{kf} = \frac{1}{2} \int_0^\infty (z_{kf}' C_{kk}' C_{kk} z_{kf} + u_{kf}' R_k u_{kf})\, d\tau_k. \tag{2.15}$$

with the minimizing controller

$$u_{kf} = -R_k^{-1} B_{kk}' P_{kf} z_{kf}, \tag{2.16}$$

where K_{kf} is the positive semidefinite stabilizing solution of the Riccati equation

$$P_{kf} A_{kk} + A_{kk}' P_{kf} + C_{kk}' C_{kk} - P_{kf} B_{kk} R_k^{-1} B_{kk}' P_{kf} = 0. \tag{2.17}$$

As explained in the previous chapter, a stabilizing controller is formed by combining the slow and fast controllers.

$$\begin{aligned} u_k = -\Big[&\left(I - R_k^{-1} B_{kk}' P_{kf} A_{kk}^{-1} B_{kk}\right) R_k^{-1} (D_{ks}' C_{ks} \\ &+ \frac{1}{\gamma_k} B_{ks}' P_s + R_k^{-1} B_{kk}' P_{kf} A_{kk}^{-1} A_{ko} \Big] x - R_k^{-1} B_{kk}' P_{kf} z_k \end{aligned} \tag{2.18}$$

Going forward, we employ Adaptive Dynamic Programming (ADP) using Kleinman's iteration [9] to learn the controller (2.9). This method focuses on finding optimal control policies by solving the Bellman equation directly and is distinct from traditional reinforcement learning methodologies that typically rely on Markov Decision Processes [14].

By defining $A^j = \hat{A}_s - B_s\tilde{K}_a^j$ for the Lyapunov equation (2.19) we can obtain the optimal value K_a so that we have $U = -K_a x_s$ to be used for (2.13). In the following algorithm the superscript $(\cdot)^j$ denotes the iteration number.

Kleinmann's Algorithm

1. Solve the Lyapunov equation below for P_s^j:

$$\tilde{P}_s^j A^j + A^{j'}\tilde{P}_s^j + \tilde{P}_s^j B_s R_s^{-1} B_s' \tilde{P}_s^j + Q_s = 0. \tag{2.19}$$

2. Update the feedback gain:

$$\tilde{K}_a^{j+1} = R_s^{-1} B_s' \tilde{P}_s^j. \tag{2.20}$$

To eliminate the dynamics of the system from (2.19), and (2.20), First, we define an arbitrary excitation signal u_o such that the system states remain bounded [13]. By substituting $u_s = u_o - \tilde{K}_a^j x_s + \tilde{K}_a^j x_s$ into the slow subsystem (2.13), we get:

$$\dot{x}_s = (A^j - B_s\tilde{K}_a^j)x_s + B_s(u_o + \tilde{K}_a^j x_s). \tag{2.21}$$

By taking the derivative of the Lyapunov function $V_{(x_s)}^j = x_s^T \tilde{P}_s^j x_s$ we get

$$\dot{V}^j = \dot{x}_s^T(\tilde{P}_s^j x_s) + x_s^T \tilde{P}_s^j(\dot{x}). \tag{2.22}$$

By substituting reduced model (2.13) into (2.22)

$$\begin{aligned}\dot{V}^j =&[x_s^j(\hat{A}_s - B_s\tilde{K}_a^j)^T + B_s^T(u_o + \tilde{K}_a^j x_s)^T](\tilde{P}_s^j x_s)\\ &+ x_s^T\tilde{P}_s^j[(\hat{A}_s - B_s\tilde{K}_a^j)x_s + (u_o + \tilde{K}_a^j x_s)B_s].\end{aligned} \tag{2.23}$$

By defining $e = u_o + \tilde{K}_a^j x_s$ and using the definition of A^j, we can write (2.23) as

$$\dot{V}^j = x_s^T(A^{j^T}\tilde{P}_s^j + \tilde{P}_s^j A^j)x_s + 2e^T B_s^T\tilde{P}_s^j x_s. \tag{2.24}$$

from (2.20) we know $B_s^T\tilde{P}_s^j = R_s\tilde{K}_a^{j+1}$, and using (2.19) we define

$$Q_j = -(A^{j^T}\tilde{P}_s^j + \tilde{P}_s^j A^j) = -Q_s - (\tilde{K}_a^{j+1})^T R_s\tilde{K}_a^{j+1}. \tag{2.25}$$

Now, by substituting (2.25) into (2.24) we get

$$\frac{d}{dt}(x_s^T\tilde{P}_s^j x_s) = -x_s^T Q_j x_s + 2e^T R_s\tilde{K}_a^{j+1}x_s. \tag{2.26}$$

Notice that the system dynamics are eliminated from (2.19) and (2.20). Moreover, (2.26) is fully independent of the dynamics of the slow sub-system. To get to the offline policy iteration we integrate both sides of (2.26) on the small interval $[\tau, \tau + \delta\tau]$

$$x_s^T(\tau+\delta\tau)\tilde{P}_s^j x_s(\tau+\delta\tau) - x_s^T(\tau)\tilde{P}_s^j x_s(\tau) = 2\int_\tau^{\tau+\delta\tau} e^T R_s\tilde{K}_a^{j+1}x_s dw \; - \int_\tau^{\tau+\delta\tau} x_s^T Q_j x_s dw. \tag{2.27}$$

We know

$$x_s^T P_s^k x_s = (x_s^T \otimes x_s^T)\mathrm{vec}(P_s^k) \tag{2.28}$$

$(\otimes)$ indicates the Kronecker product of two matrices and $\mathrm{vec}(\cdot)$ denotes the vectorization of a matrix formed by stacking the columns of the matrix into a single column vector. Using

(2.28) we can write the left side of (2.27) as

$$x_s^T(\tau+\delta\tau)P_s^k x_s(\tau+\delta\tau) - x_s^T(\tau)P_s^k x_s(\tau) = \Big[x_s^T \otimes x_s^T\Big|_{\tau}^{\tau+\delta\tau}\Big]\Big[\mathrm{vec}(P_s^k)\Big] \tag{2.29}$$

Also, we can write the first term on the right side of (2.27) as

$$\begin{aligned} &2\int_{\tau}^{\tau+\delta\tau} e^T R_s K_a^{k+1} x_s dw \\ &= \Big[2\Big(\int_{\tau}^{\tau+\delta\tau} x_s^T \otimes x_s^T dw\Big)(I_n \otimes (K_a^k)^T R_s)\Big]\Big[\mathrm{vec}(K_a^k)\Big] \\ &+ \Big[2\Big(\int_{\tau}^{\tau+\delta\tau} x_s^T \otimes u_o^T dw\Big)(I_n \otimes R_s)\Big]\Big[\mathrm{vec}(K_a^k)\Big] \end{aligned} \tag{2.30}$$

And, with knowing $x_s^T Q_k x_s = (x_s^T \otimes x_s^T)\mathrm{vec}(Q_k)$ the second term on the right side of the (2.27) can be written as

$$-\int_{\tau}^{\tau+\delta\tau} x_s^T Q_k x_s dw = -\Big[\int_{\tau}^{\tau+\delta\tau} x_s^T \otimes x_s^T dw\Big]\mathrm{vec}(Q_k) \tag{2.31}$$

By substituting (2.29), (2.30), and (2.31) back to (2.27) we get:

$$\begin{aligned} &\Big[x_s^T \otimes x_s^T\Big|_{\tau}^{\tau+\delta\tau}\Big]\Big[\mathrm{vec}(P_s^k)\Big] \\ &= \Big[2\Big(\int_{\tau}^{\tau+\delta\tau} x_s^T \otimes x_s^T dw\Big)(I_n \otimes (K_a^k)^T R_s)\Big]\Big[\mathrm{vec}(K_a^k)\Big] \\ &+ \Big[2\Big(\int_{\tau}^{\tau+\delta\tau} x_s^T \otimes u_o^T dw\Big)(I_n \otimes R_s)\Big]\Big[\mathrm{vec}(K_a^k)\Big] \\ &- \Big[\int_{\tau}^{\tau+\delta\tau} x_s^T \otimes x_s^T dw\Big]\mathrm{vec}(Q_k) \end{aligned} \tag{2.32}$$

using (2.27) we can write the offline policy iteration in the compact form below

$$\tilde{\psi}\begin{bmatrix} \text{vec}(\tilde{P}_s^j) \\ \text{vec}(\tilde{K}_a^{j+1}) \end{bmatrix} = \tilde{\Gamma}, \tag{2.33}$$

where

$$\begin{aligned}
&\tilde{\psi} = [\tilde{\delta}_{xx}, -2\tilde{I}_{xx}(I_n \otimes (\tilde{K}_a^j)^T R_s) - 2\tilde{I}_{xu_o}(I_n \otimes R_s)], \\
&\tilde{\Gamma} = \tilde{\delta}_{xx}\text{vec}(Q_j), \\
&\tilde{\delta}_{xx} = \Big[x_s^T \otimes x_s^T\Big|_{\tau_1}^{\tau_1+\delta\tau}, \quad ..., \quad x_s^T \otimes x_s^T\Big|_{\tau_j}^{\tau_j+\delta\tau}\Big]^T, \\
&\tilde{I}_{xx} = \Big[\int_{\tau_1}^{\tau_1+\delta\tau} x_s^T \otimes x_s^T dw, ..., \int_{\tau_j}^{\tau_j+\delta\tau} x_s^T \otimes x_s^T dw\Big]^T, \\
&\tilde{I}_{xu_o} = \Big[\int_{\tau_1}^{\tau_1+\delta\tau} x_s^T \otimes u_o^T dw, ..., \int_{\tau_j}^{\tau_j+\delta\tau} x_s^T \otimes u_o^T dw\Big]^T.
\end{aligned} \tag{2.34}$$

Before proceeding further, it should be emphasized that the learning in the above steps is based on the slow state system x_s, however, in practice, we only have access to the original state information x. With the help of Theorem 2.1 [8], we will show later that the learning procedure is robust to this variation as only the original state of the system is used for learning.

When collecting the data for the offline policy iteration we continue collecting x data until the $[I_{xx} \quad I_{xu_o}]$ matrix reaches full rank and $rank([I_{xx} \quad I_{xu_o}]) = \frac{n(n+1)}{2} + mn$ on the sampling interval $\delta\tau$ for the period $[\tau_i, \tau_j]$ [15]. n is the dimension of the slow subsystem and m is the number of inputs or the number of fast subsystems.

The pseudocode for the learning algorithm is given in Algorithm 1. After collecting enough data, the offline policy iteration is going to check the effectiveness of the current policy and update the policy if a pre-defined threshold is not met.

When the learning is complete, we can use the learned $\bar{K}_a$ and $\bar{P}_s$ to calculate $U = -\bar{K}_a x$.

Algorithm 1 Offline policy iteration

while rank $\left(\begin{bmatrix}\bar{I}_{xx} & \bar{I}_{xu_0}\end{bmatrix}\right) < \frac{n(n+1)}{2} + mn$ **do**
  Collect the data $x(t)$ with the excitation signal $\bar{u}_a = u_o$
  Construct the matrices $\bar{\delta}_{xx}$, $\bar{I}_{xx}$, and $\bar{I}_{xu_0}$
end while
Initialize $\bar{K}_a > 0$
while $|\bar{P}_s^j - \bar{P}_s^{j+1}| <$ Threshold **do**
  Estimate the values of $\bar{P}_s^j$ and $\bar{K}_a^{j+1}$ through (2.33)
end while
$U = -\bar{K}_a^{j+1} x$

By utilizing the calculated u_{kf} and $\bar{u}_s = U - Mx$ we write the optimal learned controller $\bar{u}_k$ as

$$\begin{aligned}\bar{u}_k =&[(I - R_k^{-1} B_{kk}' P_{kf} A_{kk}^{-1} B_{kk})\bar{u}_{ks} \\ &+ R_k^{-1} B_{kk}' P_{kf} A_{kk}^{-1} A_{ko}]x + u_{kf} z_k.\end{aligned} \tag{2.35}$$

Theorem 2.2. *Let Assumptions 1-3 hold and consider the closed-loop system (1.1)-(1.2) with (2.35) and the resulting cost functional $\bar{J}_k$ we have*

$$\lim_{\|\varepsilon\| \to 0} (\bar{J}_k - J_k^*) = 0, \quad k = 1, 2, \ldots, n.$$

where J_k^* is obtained by applying u_k^* from (1.5) on the actual system (1.1)-(1.2).

Remark 2.3. Theorem 2.2 describes how the performance of the closed-loop system (1.1)-(1.2) with (2.35) approximates the performance of the closed-loop system (1.1)-(1.2) with (1.5) as the perturbation parameters get small leading to a near-optimal performance.

Towards proving Theorem 2.2, we will present a couple of lemmas adopted from [15] that show relative convergence and closeness of the leaning-based approach.

Lemma 2.4. *At the end of the learning process, when using the original system state $x(t)$,*

the matrix $\bar{P}_s^j$ *converges as follows:*

$$\lim_{j\to\infty} \bar{P}_s^j = P_s^* + O(\|\varepsilon\|),$$

where P_s^* *is the solution to the Riccati equation* (1.6)*.*

Lemma 2.5. *During learning, when the reduced states* $x_s(t)$ *are replaced by the original states* $x(t)$ *the control system parameters undergo perturbed changes in the form*

$$\bar{P}_s^j = \tilde{P}_s^j + O(\|\varepsilon\|).$$

Equipped with these lemmas, we can now proceed to the proof for Theorem 2.2.

Proof. To avoid unboundedness as $\|\varepsilon\| \to 0$, we write the solution to the Riccati equation (1.6) P^* in this form.

$$P^* = \begin{bmatrix} P_{oo} & \varepsilon_1 P_{o1} & \varepsilon_2 P_{o2} & \dots \\ \varepsilon_1 P'_{1o} & \varepsilon_1 P_{11} & \sqrt{\varepsilon_1\varepsilon_2} P_{12} & \dots \\ \varepsilon_2 P'_{o2} & \sqrt{\varepsilon_1\varepsilon_2} P'_{12} & \varepsilon_2 P_{22} & \dots \\ \vdots & \vdots & \vdots & \end{bmatrix} \tag{2.36}$$

By defining $\alpha_{ij} = \lim \left(\frac{\epsilon_i}{\epsilon_j}\right),\ i, j = 1, 2, \dots, N$ it can be shown that

$$P_{oo}(0) = P_s, \quad P_{ii}(0) = s_i P_f, \quad P_{ij}(0) = 0, \tag{2.37}$$

$$P_{ok}(0) = P_{oo}(0)\hat{E}_k - \gamma_k \tilde{E}_k. \tag{2.38}$$

where $\hat{E}_k = \left(\tilde{S}_{ok} P_{kf} - A_{ok}\right)(A_{kk} - S_{kk} P_{kf})^{-1}$, and $\tilde{E}_k = (A'_{ko} P_{kf} + C'_{ok} C_{kk})(A_{kk} - S_{kk} P_{kf})^{-1}$. As shown in [8], the above solution does not depend on α_{ij}, and the limits of $P_{ij}(0)$ are

uniquely defined.

Using Lemma 2.4 and Lemma 2.5, we can now rewrite the bound of P_{oo} as $P_{oo}(0) = \bar{P}_s + O(\|\varepsilon\|)$. The limit of J_k^* as $\|\varepsilon\| \to 0$ is $J_k^* = \frac{1}{2}\hat{x}_o' M \hat{x}_o$, where M satisfies

$$M(A - SP) + (A - SP)'M + Q_k + \frac{1}{\gamma_k^2} P S_k P = 0. \tag{2.39}$$

Also, to evaluate the actual cost $\bar{J}_k$ we express (2.35) as

$$\bar{u}_k = -\frac{1}{\gamma_k} R_k^{-1} B_k' L \hat{x} + O(\|\varepsilon\|), \tag{2.40}$$

where

$$L = \begin{bmatrix} P_s & 0 & 0 & \dots \\ \varepsilon_1 P_{1m}' & \varepsilon_1 \gamma_1 P_f & 0 & \dots \\ \varepsilon_2 P_{2m}' & 0 & \varepsilon_2 \gamma_2 P_f & \dots \\ \vdots & \vdots & \vdots & \end{bmatrix}$$

and $P_m = \bar{P}_s \hat{E}_k - \gamma_k \tilde{E}_k + O(\|\varepsilon\|) = P_{k0}(0) + O(\|\varepsilon\|)$. When $\bar{u}_k$ is applied to (1.1), we have $\bar{J}_k = \frac{1}{2}\hat{x}_o' N \hat{x}_o$, where N satisfies the Lyapunov equation

$$N(A - SL) + (A - SL)'N + Q_k + \frac{1}{\gamma_k^2} L' S_k L + O(\|\varepsilon\|) = 0. \tag{2.41}$$

To get $\bar{J}_k - J_k^*$, we calculate $W^{(k)} = N^{(k)} - M^{(k)}$ for each area k. Next by subtracting (2.41) from (2.39) we get

$$\begin{aligned} &W^{(k)}(A - SL) + (A - SL)'W^{(k)} + \frac{1}{\gamma_k^2} L' S_k L - \frac{1}{\gamma_k^2} P S_k P \\ &\quad + M^{(k)}(S)(P - L) + (P - L)' S M^{(k)} + O(\|\varepsilon\|) = 0. \end{aligned} \tag{2.42}$$

By writing $W^{(k)}$ in the form of (2.36), it can be shown that $\lim_{\|\varepsilon\| \to 0} W_{ij}^{(k)} = 0$. This completes the proof. □

Chapter 3

Learning-Based Control of a Power System

Consider a power system consisting of two automatic load frequency control areas, which are connected with weak tie-lines and have identical characteristics.

Although an actual power system control has multiple components and controllers we assume that each automatic load frequency control consists of a generator, governor and a non-reheat steam turbine. The goal is to maintain the steady frequency and control the tie-line flows.

The power system equations for the two interconnected areas are given as [5],[22]

$$\dot{v}_k = (ACE)_k = \Delta P_{tie_k} + b_s \Delta f_k, \tag{3.1}$$

$$\Delta \dot{f}_k = \frac{1}{T}[-\Delta f_k + \frac{1}{D}(\Delta P_{G_k} - \Delta P_d - \Delta P_{tie_k})], \tag{3.2}$$

$$\Delta \dot{P}_{12} = T_{12}(\Delta f_1 - \Delta f_2), \tag{3.3}$$

$$\Delta \dot{P}_{G_k} = \frac{1}{T_t}[-\Delta P_{G_k} + \Delta a_k], \tag{3.4}$$

$$\Delta \dot{a}_k = \frac{1}{T_t}[-\Delta P_{G_k} + \Delta a_k], \tag{3.5}$$

where ($N = 2$ and $k = 1, 2$) and $\Delta P_{12} = \Delta P_{tie_1} = -\Delta P_{tie_2}$. Δf is the frequency error, ΔP_{tie} is the tie line power flow variation and v_k is the integral of the area control error $(ACE)_k$, Δa is the turbine valve position variation, ΔP_G is the turbine output variation and the ΔP_c

is variation in the speed changer.

The system parameters are given as follows: inertia time constant $T = 20$, synchronizing power flow coefficient $T_{12} = 32.7$, turbine time constant $T_t = 0.2$, and governor time constant $T_G = 0.1$. Assuming a constant load disturbance $\Delta P_d = 0$ we define speed regulation $r = 0.25$, bias factor $b_s = 4.5$ and $D = 0.5$. In addition, we define

$$\varepsilon_i = \frac{\max(T_{G_i}, T_{t_i})}{T_i} \tag{3.6}$$

leading to $\varepsilon_1 = \varepsilon_2 = \frac{0.2}{20} = 0.01$, which is a small number relative to T_G and T_t (i.e. $T_G = 10\varepsilon_i$, $T_{t_i} = 20\varepsilon_i$). Also, $\varepsilon_{12} = \varepsilon_{21} = 0$.

Accordingly, the slow and fast states are defined as

$$x = (v_1, v_2, \Delta f_1, \Delta f_2, \Delta P_{12}) \tag{3.7}$$

$$z_k = (\Delta P_{G_k}, \Delta a_k) \qquad k = 1, 2 \tag{3.8}$$

The system matrices in (1.1) are given as:

$$A_o = \begin{bmatrix} 0 & 0 & b_s & 0 & 1 \\ 0 & 0 & 0 & b_s & -1 \\ 0 & 0 & -\frac{1}{T} & 0 & -\frac{1}{TD} \\ 0 & 0 & 0 & -\frac{1}{T} & \frac{1}{TD} \\ 0 & 0 & T_{12} & -T_{12} & 0 \end{bmatrix}, A_{1o} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{rT_G} & 0 & 0 \end{bmatrix}, A_{2o} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{rT_G} & 0 \end{bmatrix},$$

$$A_{o1} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{TD} & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, A_{o2} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{TD} & 0 \\ 0 & 0 \end{bmatrix}, B_{o1} = B_{o2} = \begin{bmatrix} 0 \end{bmatrix}, A_{11} = A_{22} = \begin{bmatrix} -\frac{1}{T_t} & \frac{1}{Tt} \\ 0 & -\frac{1}{T_G} \end{bmatrix}, B_{11} = B_{22} = \begin{bmatrix} 0 \\ \frac{1}{T_G} \end{bmatrix}.$$

Suppose $\gamma_1 = \gamma_2 = 0.5$, $R_1 = R_2 = 20$, and

$$C_{o1} = C_{o2} = \begin{bmatrix} I_{(5\times5)} \\ 0_{(4\times5)} \end{bmatrix}, C_{11} = \begin{bmatrix} 0_{(5\times2)} \\ I_{(2\times2)} \\ 0_{(2\times2)} \end{bmatrix}, C_{22} = \begin{bmatrix} 0_{(5\times2)} \\ 0_{(2\times2)} \\ I_{(2\times2)} \end{bmatrix},$$

By writing (2.8) and (2.15) and solving the Riccati equations (2.10) and (2.17) we get the solution for the slow and fast subsystem respectively as

$$u_{1s} = \begin{bmatrix} -0.3015 & 0 & -4.079 & 1.8219 & -0.0478 \end{bmatrix} x_s$$
$$u_{2s} = \begin{bmatrix} 0 & -0.3015 & 1.8219 & -4.079 & 0.0478 \end{bmatrix} x_s$$
$$u_{kf} = \begin{bmatrix} -0.0162 & -0.0326 \end{bmatrix} z_{kf} \tag{3.9}$$

Then, according to (2.18) we calculate u_k as

$$u_1 = [-0.3162 \quad 0 \quad -4.4733 \quad 1.9108 \quad -0.0501]x + [-0.0162 \quad -0.0326]z_1$$
$$u_2 = [0 \quad -0.3162 \quad 1.9108 \quad -4.4733 \quad 0.0501]x + [-0.0162 \quad -0.0326]z_2 \tag{3.10}$$

By writing the original system (1.1), The optimal value for the controller (1.5) for the actual

system (1.1) would be calculated as

$$\begin{aligned} u_1^* &= [-0.3162 \quad 0 \quad -4.5806 \quad 1.8694 \quad 0.0530]x \\ &\quad + [-0.1079 \quad -0.0759]z_1 + [0.0378 \quad 0.0176]z_2 \\ u_2^* &= [0 \quad -0.3162 \quad 1.8694 \quad -4.5806 \quad -0.0530]x \\ &\quad + [0.0378 \quad 0.0176]z_1 + [-0.1079 \quad 0.0759]z_2 \end{aligned} \tag{3.11}$$

Following Section 2.2, we implemented Algorithm 1 considering a sampling interval $\delta\tau = 0.1s$, with the initial states being 0.1. Using Q_s and R_s from (2.11) we were able to learn the controller $\bar{u}_s$ using different ε_k values. By having the solution for the fast subsystem we use (2.35) to calculate the controller $\bar{u}_k$. (3.12) shows the learned controller value $\bar{u}_s$ and $\bar{u}_k$ based on (2.35) for $\varepsilon_k = 10^{-7}$.

$$\begin{aligned} u_{1s} &= \begin{bmatrix} -0.3001 & -0.0020 & -3.9105 & 1.7897 & -0.0481 \end{bmatrix} x_s \\ u_{2s} &= \begin{bmatrix} -0.0014 & -0.2985 & 1.7896 & -3.9059 & 0.0482 \end{bmatrix} x_s \end{aligned}$$

$$\begin{aligned} \bar{u}_1 &= [-0.3147 \quad -0.0021 \quad -4.2966 \quad 1.8771 \quad -0.0504]x + [-0.0162 \quad -0.0326]z_1 \\ \bar{u}_2 &= [-0.0015 \quad -0.3131 \quad 1.8769 \quad -4.2918 \quad 0.0506]x + [-0.0162 \quad -0.0326]z_2 \end{aligned} \tag{3.12}$$

Figure 3.1 shows the x_1 and x_3 states trajectory when applying the optimal controller (1.5), using the MATLAB command *lqr* for the system (1.1)-(1.2), and controller based on learning (2.35) with different ε_i values over 30 seconds. It can be seen that the state trajectories using the learned controller get closer to the optimal controller as $\|\varepsilon\|$ decreases. It is worth stressing that the optimal controller is designed based on a system of dimension 9. However,

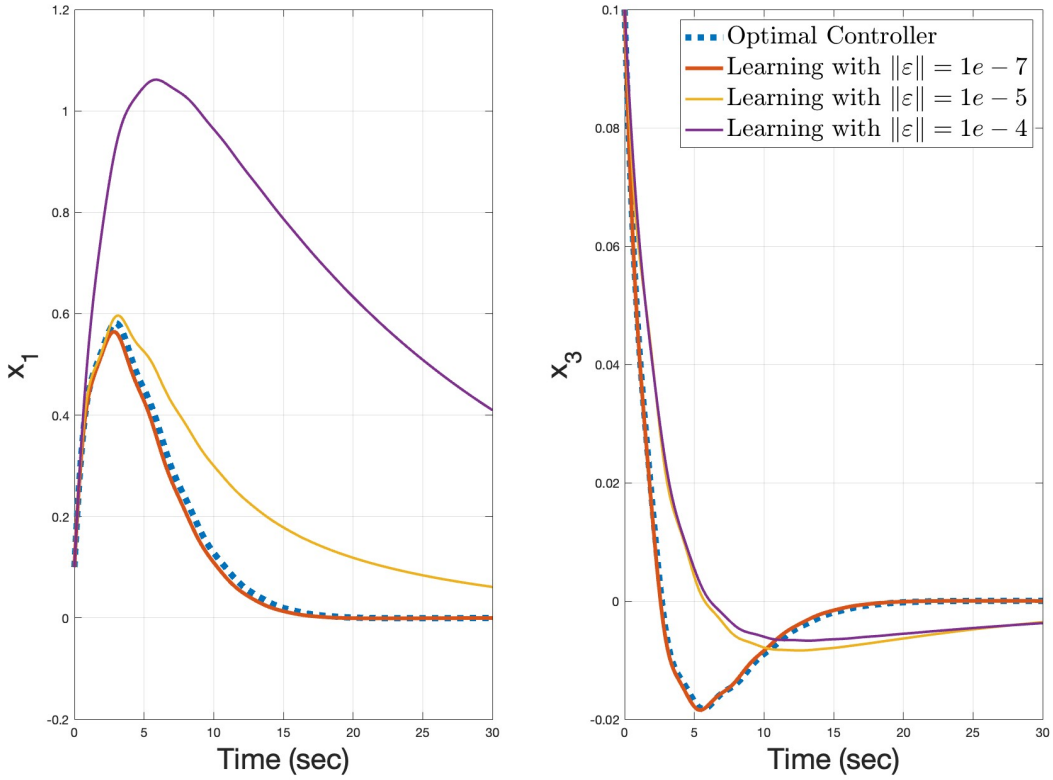

Figure 3.1: x_1 and x_3 states trajectory using for different controllers

the learning controller has dimension 5 as it makes use of state vector (3.7). In addition, the fast controller is designed based on a subsystem of dimension 2. So it can be seen that the proposed method achieves comparable results while significantly reducing the complexity of the problem and, hence, computational costs.

Figure 3.2 illustrates the positioning of the actual system's closed-loop slow poles when various control strategies are applied. It is evident that as the closed loop poles of the system progressively converge towards the positions achieved by the optimal controller ε_k decreases. Notably, with the value of ε_k equal to 1×10^{-7} the resulting closed-loop slow poles are remarkably close to those determined by the optimal controller. This suggests that with a sufficiently small epsilon, the learned controller is highly effective in emulating the optimal

controller's behavior, thereby enhancing the system's performance and stability.

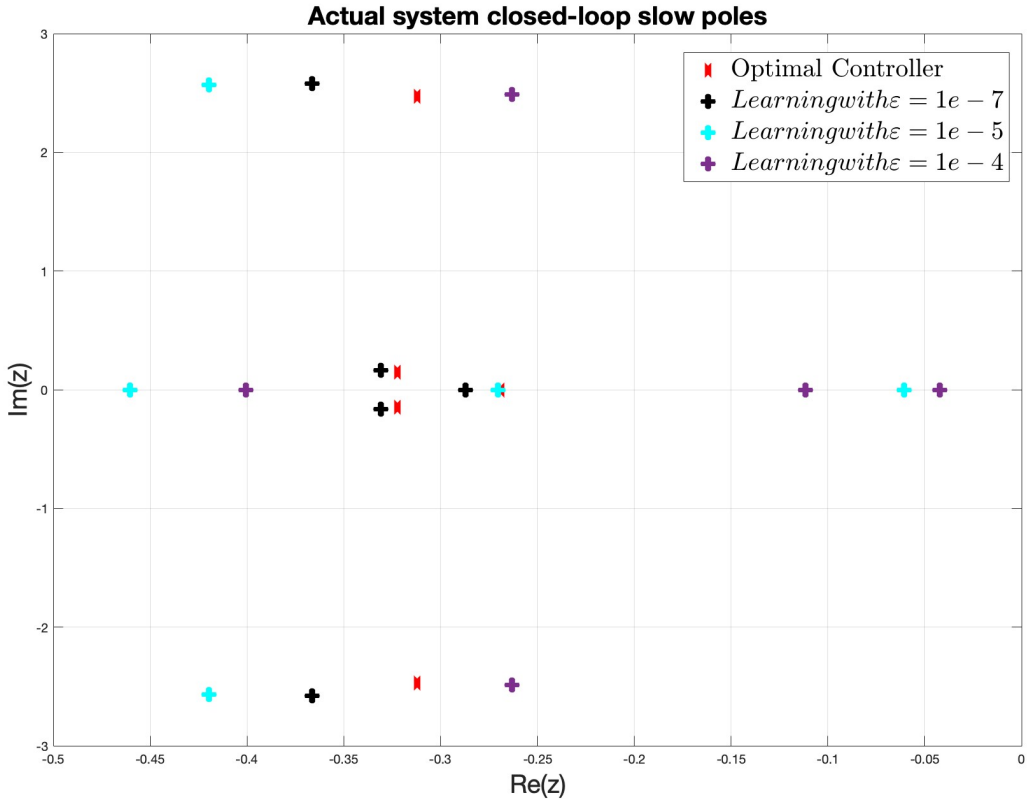

Figure 3.2: Actual system closed loop slow pole-location with different controllers

	Control Area1 $(J_1^*, \bar{J}_1)$	Control Area2 $(J_2^*, \bar{J}_2)$	Overall Cost $(J^*, \bar{J})$
Optimal Value	0.3445	0.3472	0.3459
Learning Based Value	0.3514	0.3532	0.3523
Error Percentage	2.0077%	1.7160%	1.8613%

Table 3.1: Average cost function values over 15 seconds

Table 3.1 validates the result presented in Theorem 2.2, where it shows that the learning-based values $\bar{J}_k$ get closer to the optimal cost J_k^* as $\|\varepsilon\|$ decreases. With $\varepsilon_1 = \varepsilon_2 = 10^{-7}$, the overall cost value of the learning-based controller is within 1.8613% of the optimal cost value. Table 3.1 values have been calculated by averaging the cost function values until the

system states became steady at zero.

Chapter 4

Conclusion and Future work

In this work, we presented a model-free reinforcement learning method to solve a Pareto optimal control in large-scale systems with unknown slow dynamics. This approach scales well as the size of the system gets large as the learning is performed based on a reduced-order model and is done on the level of each area. In addition, the proposed method considers all the dynamics of the systems and leads to a near-optimal performance.

In future work, it would be interesting to investigate the effectiveness of the proposed approach when considering nonlinear and stochastic nature of systems. Furthermore, it is worth pointing out that this paper presents a general framework that facilitates learning by exploiting time scale separation of the system dynamics. This peaks interest in comparing the proposed learning method with others in terms of computational costs.

Bibliography

[1] Michael Athans and Peter L Falb. *Optimal control: an introduction to the theory and its applications*. Courier Corporation, 2013.

[2] Almuatazbellah M. Boker, Thomas R. Nudell, and Aranya Chakrabortty. On aggregate control of clustered consensus networks. In *2015 American Control Conference (ACC)*, pages 5527–5532, 2015. doi: 10.1109/ACC.2015.7172204.

[3] Almuatazbellah M Boker, Chengzhi Yuan, Fen Wu, and Aranya Chakrabortty. Aggregate control of clustered networks with inter-cluster time delays. In *2016 American Control Conference (ACC)*, pages 5340–5345. IEEE, 2016.

[4] Zaiwei Chen, Sajad Khodadadian, and Siva Theja Maguluri. Finite-sample analysis of off-policy natural actor-critic with linear function approximation. In *IEEE Control Systems Letters*, 2022.

[5] Olle I. Elgerd and H. H. Happ. Electric energy systems theory: An introduction. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-2(2):296–297, 1972. doi: 10.1109/TSMC.1972.4309116.

[6] Petros Ioannou and Bariş Fidan. *Adaptive control tutorial*. SIAM, 2006.

[7] H. Khalil and P. Kokotovic. Control strategies for decision makers using different models of the same system. *IEEE Transactions on Automatic Control*, 23(2):289–298, 1978. doi: 10.1109/TAC.1978.1101712.

[8] H. Khalil and P. Kokotovic. Control strategies for decision makers using different models

of the same system. *IEEE Transactions on Automatic Control*, 23(2):289–298, 1978. doi: 10.1109/TAC.1978.1101712.

[9] David Kleinman. On an iterative technique for Riccati equation computations. *IEEE Transactions on Automatic Control*, 13(1):114–115, 1968.

[10] Petar V. Kokotovic, Robert E. O'Malley, and Peddapullaiah Sannuti. Singular perturbations and order reduction in control theory - an overview. *Autom.*, 12, 1975.

[11] Biao Luo, Huai-Ning Wu, and Tingwen Huang. Off-policy reinforcement learning for h_infty control design. *IEEE transactions on cybernetics*, 45:65–76, 01 2015. doi: 10.1109/TCYB.2014.2319577.

[12] Lakshmi Mandal, Raghuram Bharadwaj Diddigi, and Shalabh Bhatnagar. Variance-reduced deep actor-critic with an optimally sub-sampled actor recursion. *IEEE Transactions on Artificial Intelligence*, March 2024.

[13] Sayak Mukherjee, He Bai, and Aranya Chakrabortty. Reduced-dimensional reinforcement learning control using singular perturbation approximations. *Automatica*, 126: 109451, 2021. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2020.109451.

[14] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[15] Vasanth Reddy, Almuatazbellah Boker, and Hoda Eldardiry. Learning-based optimal control of linear time-varying systems over large time intervals. *Systems Control Letters*, 185:105750, 2024. ISSN 0167-6911. doi: https://doi.org/10.1016/j.sysconle.2024.105750.

[16] Richard S Sutton and Andrew G Barto. Reinforcement learning: an introduction mit press. *Cambridge, MA*, 22447, 1998.

[17] Kyriakos G Vamvoudakis. Q-learning for continuous-time linear systems: A model-free infinite horizon optimal control approach. *Systems & Control Letters*, 100:14–20, 2017.

[18] Kyriakos G Vamvoudakis and Frank L Lewis. Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem. *Automatica*, 46(5):878–888, 2010.

[19] Kyriakos G Vamvoudakis, Draguna Vrabie, and Frank L Lewis. Online adaptive learning of optimal control solutions using integral reinforcement learning. In *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 250–257.

[20] Haoyan Zhang, Huanqing Wang, Ben Niu, Liang Zhang, and Adil M Ahmad. Sliding-mode surface-based adaptive actor-critic optimal control for switched nonlinear systems with average dwell time. *Information Sciences*, 580:756–774, 2021.

[21] Haoyan Zhang, Xudong Zhao, Huanqing Wang, Guangdeng Zong, and Ning Xu. Hierarchical sliding-mode surface-based adaptive actor–critic optimal control for switched nonlinear systems with unknown perturbation. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[22] Milan S. Ćalović. Power system load and frequency control using an optimum linear regulator with integral feedback. *IFAC Proceedings Volumes*, 5(1, Part 1):400–408, 1972.