공학박사 학위논문

Model-Based Reinforcement Learning for Process Control and Optimization

모델 기반 강화학습을 이용한 공정 제어 및 최적화

2020년 2월

서울대학교 대학원
화학생물공학부
김 종 우

Model-Based Reinforcement Learning for Process Control and Optimization

지도교수 이 종 민

이 논문을 공학박사 학위논문으로 제출함
2019년 12월

서울대학교 대학원
화학생물공학부
김 종 우

김 종 우의 공학박사 학위논문을 인준함
2019년 12월

위 원 장 ______________________ (인)

부위원장 ______________________ (인)

위 원 ______________________ (인)

위 원 ______________________ (인)

위 원 ______________________ (인)

Abstract

Model-Based Reinforcement Learning for Process Control and Optimization

Jong Woo Kim
School of Chemical and Biological Engineering
The Graduate School
Seoul National University

Sequential decision making problem is a crucial technology for plant-wide process optimization. While the dominant numerical method is the forward-in-time direct optimization, it is limited to the open-loop solution and has difficulty in considering the uncertainty. Dynamic programming method complements the limitations, nonetheless associated functional optimization suffers from the curse-of-dimensionality. The sample-based approach for approximating the dynamic programming, referred to as reinforcement learning (RL) can resolve the issue and investigated throughout this thesis. The method that accounts for the system model explicitly is in particular interest. The model-based RL is exploited to solve the three representative sequential decision making problems; scheduling, supervisory optimization, and regulatory control. The problems are formulated with partially observable Markov decision process, control-affine state space model, and general state space model, and associated model-based RL algorithms are point based value iteration (PBVI), globalized dual heuristic programming (GDHP), and differential dynamic programming (DDP), respectively.

The contribution for each problem can be written as follows: First, for the scheduling problem, we developed the closed-loop feedback scheme which highlights the strength compared to the direct optimization method. In the second case, the regulatory control problem is tackled by the function approximation method which relaxes the functional optimization to the finite dimensional vector space optimization. Deep neural networks (DNNs) is utilized as the approximator, and the advantages as well as the convergence analysis is performed in the thesis. Finally, for the supervisory optimization problem, we developed the novel constraint RL framework that uses the primal-dual DDP method. Various illustrative examples are demonstrated to validate the developed model-based RL algorithms and to support the thesis statement on which the dynamic programming method can be considered as a complementary method for direct optimization method.

Keywords: Reinforcement learning, optimal control, dynamic optimization, scheduling, deep learning, partially observable Markov decision process, differential dynamic programming
Student Number: 2014-21532

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and previous work

Plant-wide process operation is generally described as a decision-making hierarchy consisted with several time-scales and spatial-scales [1, 2, 3, 4, 5]. While details differ from each other, our three-level description includes the scheduling for plant-wide level, the supervisory optimization for process unit level, and the regulatory control problem for the actuator level (see, Fig. 1.1). This hierarchy separation is considered to be necessary, for the formulation becomes easily intractable when multiple scales are considered within a single problem [5].

Considering the process dynamics, all problems can be translated as a single framework of 'sequential' decision making problem. Finding efficient algorithm is the way to enable the process automation, and thus regarded as the core topics in modern process systems engineering (PSE) field [6, 7, 8]. The ultimate goal for the plant-wide optimization is to develop the methodology that is capable for attaining the optimality in the large-scale hybrid model with constraints satisfaction, fast online calculation and adaptation, a closed-loop solution, and to possess a robustness against the online disturbance.

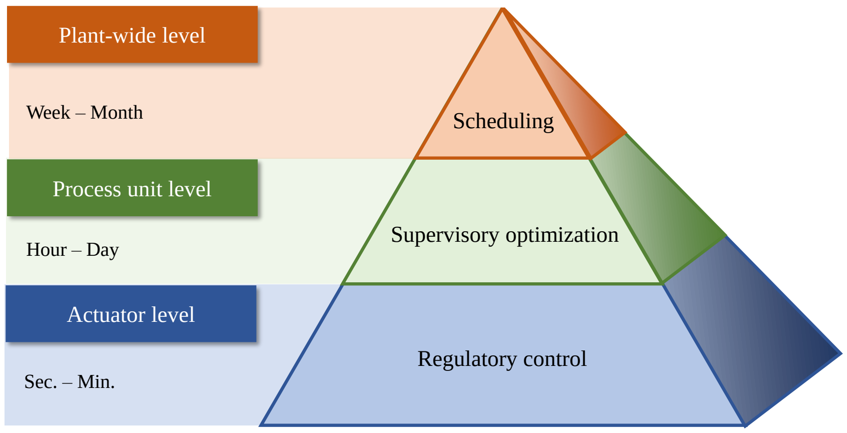


Figure 1.1: The canonical hierarchy of plant-wide process operation.

Solution approaches for sequential decision making problem are generally divided into three major classes by their transcription method, which are direct method, indirect method, and dynamic programming method [9]. In direct method, entire state and/or control variables throughout the horizon are concatenated and considered as a single vector of decision variable. The system dynamics usually given as differential algebraic equation (DAE) is discretized temporally and then considered as equality constraints. The transcription result is a static nonlinear optimization problem (NLP), to which well known optimization solvers can be utilized. The indirect approach uses Pontryagin's maximum principle which is developed by the calculus of variations. The additional variable called costate is introduced to yield the adjoint differential equation defined backward-in-time. The multiple-point boundary value problem (BVP) which is served as the necessary condition of optimality is formulated. Note that the discretization happens after which the differential equations are obtained. The first two methods are having the opposite properties in terms of transcription results, discretization, and the temporal direction. Direct method is characterized as forward-in-time and discretize-and-optimize, whereas indirect method can be viewed as backward-in-time and optimize-and-discretize.

The third method, dynamic programming method provides the sufficient condition of the optimality by the verification theorem [10], hence poses a stronger condition then the other two methods. In dynamic programming method, the value function which states the potential function or 'cost-to-go' of the optimization objective is introduced. Hamilton-Jacobi-Bellman (HJB) which has a partial differential equation (PDE) form with respect to the value function is derived by using the dynamic programming principle. Dynamic programming solves the infinite dimensional functional optimization problem on the contrary to the finite vector space optimization problem in the direct method. The numerical method for PDE involves the discretization in time domain as well as the spatial domain. One can observe the similarities between the indirect method; backward-in-time, optimize-and-discretize, and using Hamiltonian terms.

Among the three methods, the dominant in PSE field has been the direct method approach [8]. This is because the two optimize-and-discretize methods, i.e., the indirect method and dynamic programming method have disadvantages that prevent them from being adopted in practice. The former is too sensitive to the initial guess [11] and inefficient in the presence of the inequality constraint [12], while the latter has the curse-of-dimensionality that comes from the spatial discretization [13]. On the other hand, discretize-and-optimize paradigm for the direct method makes itself applicable to the various problem settings such as DAE, mixed integer hybrid dynamics [14], and associated inequality constraints can be easily formulated. The combination with the powerful large-scale optimizer solvers such as IPOPT [15] enables the active applications of direct optimization method to the whole stages of plant-wide optimization hierarchy such as scheduling, supervisory optimization, and regulatory control [16].

Despite having the success history, there exist several limitations on the direct optimization method. First problem is that the number of decision variables is proportional to the horizon. The problem becomes insignificant in case of deterministic system due to the development of large-scale optimization solver which can deal with millions of variables [16]. However, in case of stochastic optimization, decision variable grows exponentially unlike the deterministic case, which motivates various relaxation methods [17, 18]. Second problem is that only the open-loop solution is available in direct optimization method [19]. Note that the solution of the dynamic programming method is the policy which essentially makes the closed-loop by mapping the state to the optimal control. Noticing these limitations, the recent researches on the direct method focus on avoiding the online reoptimization in the presence of uncertainty or model-plant mismatch [5]. The list of methodologies utilize the multiparametric programming [20, 21], modeling the uncertainty [22, 23, 24], receding horizon methods [25, 26], and the NLP sensitivity approaches [27, 28, 16].

Based on these observations, the dynamic programming method has been considered as a complementary method under which the closed-loop policy can be obtained without being intractable in the stochastic setting [29, 30, 31, 32]. The principal task for the dynamic programming is to perform the functional optimization and to overcome the curse-of-dimensionality. However, the complete solution is only available in the case where spatial discretization is possible such as finite set or low-dimensional domain or in the case where the analytic function form can be obtained such as linear quadratic regulation problem [13]. Since the HJB equation seldom can be solved exactly, various approaches to build or implement approximate solutions have been tried. The approaches try to construct approximate solutions to the HJB equation via simulation-based learning, as in neurodynamic programming [33], heuristic dynamic programming [34], and approximate dynamic programming (ADP) [35]). The fact that optimal control policies are learned iteratively through data connects the approach to the machine-learning framework of reinforcement learning (RL), where an agent also tries to learn optimal decision policies iteratively by interacting with its environment [36].

The common procedure of RL consisted with three steps, i.e., data generation, performance evaluation, and policy improvement. Hence the algorithms can be classified by how they take each step differently, and the success of RL relies on properly confining the search space to the manifold in the neighborhood of the data appearance [35]. The recent advances in machine learning community enables the feature analysis of the raw sensory-level data by using deep neural networks (DNNs) and the implementation of various information-theoretic techniques. The resulting deep RL (DRL) shows remarkable performances in certain applications such as robotics, autonomous driving, games, etc [37, 38, 39]. The potential advantage of the learning based approach over the direct optimization approach, apart from which the ability for the closed-loop solution and the online computation elimination, is that it is flexible enough to work with varying levels of system knowledge ranging from a completely known system to no system knowledge at all, i.e., having on-line data only (i.e., model-free) [34].

Model-free is a characteristic that distinguishes RL from other optimal control methods. All steps of RL algorithm can be performed just with the reinforcement signal (i.e., reward or cost) and without the model knowledge. The only requirement for the system models in RL framework is a Markov property [40]. Since the model identification and parameter estimation are also large fields in PSE, bypassing these provides a huge advantage for designing the controller [41].

The model-free RL contains a value-based method and a policy based-method. The value-based methods rely on the property of contraction Bellman operator and solves the fixed point problem for obtaining the optimal value function [13]. The particular methods range from the simple tabular-based temporal difference learning such as Q-learning and SARSA (see [36]), to the methods with the linear function-approximated value function such as fitted Q-iteration and least square methods (e.g., LSTD, LSPE) introduced in [42], and to the DRL methods represented as deep Q-network (DQN) and its variants [43, 44, 45, 46]. In contrast to value-based method, the policy-based method focuses on the parameter optimization of the parametrized policy function [47]. The optimization is performed by policy or natural gradient [48, 49], with the expectation-maximization algorithm [50], or using the information-theoretic approaches [51]. In actor-critic (AC) method, both value and policy functions are parametrized. Various benchmark DRL algorithms have been developed within this category, e,g, DDPG, A3C, ACER, TRPO, SAC, etc [52, 53, 54, 55, 56].

Model-based method, on the other hand, utilizes the model explicitly. Although the process modeling requires extensive domain knowledge and the results are inaccurate in most cases, it provides a theoretical background of optimality under the specific conditions [57]. Moreover, the amount of data for model-free RL is usually intractable for the direct process application [58, 59, 5, 60]. Hence, available knowledge on the system dynamics and cost should be fully used. Fortunately, several system and cost structures allows for the transformation of HJB equation to more tractable forms. For example, when the system dynamics is control-affine and the objective function is quadratic, the derivatives of model structure and the value function are incorporated to yield the explicit form of the control policy. Several model-based RL algorithms have been developed with the aim to solve the HJB adaptively, hence referred to as adaptive DP (ADP) [61, 62, 34, 63]. The model-based local optimal policy can be parametrized in order to design the end-to-end controller [37]. Another extension is to study the connection between the stochastic optimal control framework [64, 65, 38].

The pioneer works on the PSE applications of RL was first appeared in [66, 67, 68, 69, 70]. The model-free RL was adopted for solving the optimal control, dual adaptive control, and the scheduling problems. It was discussed that the value function approximation can achieve the robust decision with respect to the process noise and the system drift. The result is extended to the dynamic optimization and robust optimal control problem [71, 72, 73]. The recent applications of RL based process control are yet mainly based on the linear approximator in either model-based and model-free methods [74, 75, 76, 77, 78]. Meanwhile, a few DRL applications on PSE are studied in [79, 41].

From the perspectives of PSE and machine learning fields, RL, the sample-based approximate solution of dynamic programming, can be considered as a candidate of the complementary solution for the direct optimization method in PSE field. The model-based RL method is the scope of the thesis due to the intractable data amount requirement for the model-free counterpart. We developed the RL methods for three sequential decision making problems in Fig. 1.1 in order to validate the thesis statement.

1.2 Statement of contributions

The main objective of this thesis is to develop the backward-in-time value based methods for the three sequential decision-making problems in process systems engineering; scheduling, supervisory optimization, and regulatory control. The feedback property which is obtained as a natural consequence of using value based methods is investigated. Moreover, the value and the policy function are parametrized by deep neural networks (DNNs) so that the methods can be extended to large-scale state space problems. Advantages and convergence analysis of using DNNs parametrization have been performed. Proposed methods for each problem is validated with illustrative examples. The summary of the four chapters are below:

- A POMDP framework for integrated scheduling of infrastructure maintenance and inspection.
- A model-based deep reinforcement learning method applied to finite-horizon optimal control of nonlinear control-affine system.
- Convergence analysis of the model-based deep reinforcement learning for optimal control of nonlinear control-affine system.
- Primal-dual differential dynamic programming for constrained dynamic optimization of continuous system.

The first work is the scheduling problem formulated as a partially observable Markov decision process (POMDP) with finite state space. In POMDP, the Bellman equation is expressed with respect to the probability distribution of state, belief. As a consequence, even for the finite state problem the domain becomes a simplex set, which is an uncountable set. We exploited the point based value iteration (PBVI) for the RL algorithm in POMDP case. The receding horizon POMDP is proposed in order to construct the feedback loop with respect to the newly obtained observations. The water main system infrastructure maintenance and inspection problem is illustrated as an example. Sensor-installation is considered as a part of the integrated scheduling, as an improvement to previous maintenance and inspection-only scheduling.

The second part is the finite-horizon regulatory tracking control (FHOC) problem for the nonlinear control-affine system. Proposed RL algorithm for control-affine system is the globalized dual heuristic programming (GDHP) with the DNNs parametrization. A suitable DNNs structure for the GDHP method, which handles certain features of the FHOC problem, e.g., time-varying value and policy functions, presence of boundary conditions, is suggested. The governing equation is modified to a suitable form for the delta-input formulation used to address the time-varying reference tracking requirement. In addition, several ways to alleviate difficulties arising in training DNNs are introduced and adopted. Finally, the overall method is applied to examples of a nonlinear batch reactor and 1-dimensional diffusion-convection-reaction process, which have high dimensional state space. It is shown that use of DNNs is critical in obtaining a converging policy with satisfactory performance in the presence of uncertainties.

The third topic is the convergence analysis of the infinite-horizon regulation control under the GDHP algorithm with DNNs parametrization. A similar formulation and methodology is taken with that in the FHOC problem with slight modifications on time-invariant function and discount factor. We provided the thorough convergence analysis of the proposed methods; First, a new convergence result for the costate iteration in GDHP is provided. Second, the convergence analysis for DNNs parameters and the closed-loop stability are performed. Finally, the proposed method is applied to the high-dimensional state-space system governed by PDE heat equation to highlight the necessity of DNNs as the function approximator.

The final part is on the supervisory optimization problem, whose formulation is continuous-time control-nonaffine system. In contrast to the regulatory control problems, the policy function cannot be expressed as an explicit function form. Instead, only the optimal variation with respect to the previous iteration is available, which is essentially similar to the Newton's method for HJB problem. Differential dynamic programming (DDP) is the particular method that follows this approach. We considered the constrained optimization problem with the novel formulation called primal-dual DDP. The min-max operation computing the optimal primal (control) and dual (constraint Lagrangian) is cast to the HJB equation. Moreover, the primal-dual augmented Lagrangian method is implemented for enabling the Newton update for dual variable as well, which results in the improved convergence property.

1.3 Outline of the thesis

The remainder of the thesis is organized as follows. In Chapter 2, the background on the formulation of principle of optimality to the Markov model and the state space model is introduced, and the overview of the RL algorithms for each sequential decision making problems are provided. In Chapter 3, the POMDP scheduling problem for integrated infrastructure maintenance and inspection is discussed. Chapter 4 proposes algorithm of GDHP with DNNs approximation for discrete-time FHOC problem. The convergence analysis for the algorithm developed in Chapter 4 is provided in Chapter 5. The constrained supervisory optimization problem is solved with primal-dual DDP and the details is given in Chapter 6. Finally, general concluding remarks and possible directions for further study are given in Chapter 7.

Chapter 2

Background and preliminaries

2.1 Optimization problem formulation and the principle of optimality

In this section, the optimization problems for scheduling, supervisory optimization, and regulatory control are formulated. Scheduling problem is modelled with the partially observable Markov decision process (POMDP), and supervisory and regulatory problems are described with the state space model. Bellman equations which state the principle of optimality are developed for each problem.

Note that in spite of the POMDP model and the state space model are separately described, the two models are fundamentally analogous and only differs in their nomenclature (see Fig. 2.1). Henceforth, we distinguish two methods that POMDP for the cases when the finite state set with unstructured model, while state space model for the cases when the uncountable state space with the structured first-principle model is used.

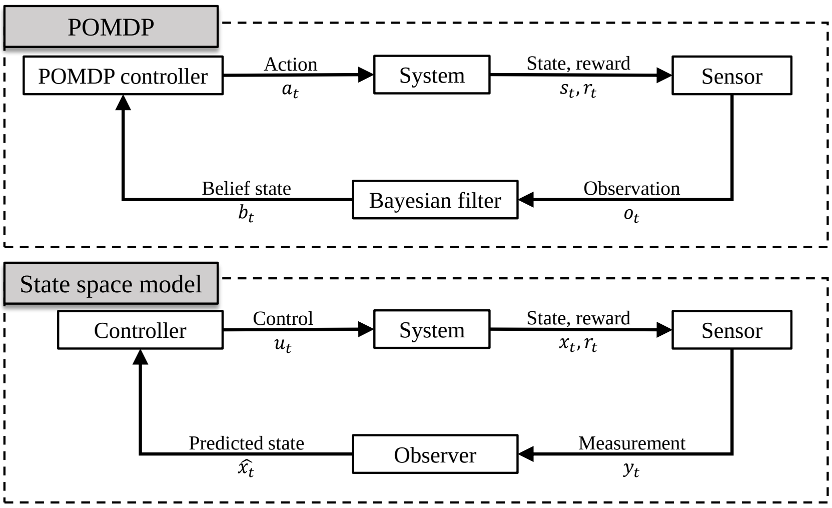


Figure 2.1: Correspondence between POMDP and state space model.

2.1.1 Markov decision process

Markov process is a stochastic process that the conditional probability distribution of the future state depends only on the present state, not the past history of the process. It is the standard requirement for the process model to have a Markov property for developing various decision-related methods [13]. A sequential decision making problem that satisfies the Markov property is called Markov decision process (MDP). The objective of MDP is to find the optimal feedback policy that maps the state into the action which optimizes the objective function defined in either finite or infinite horizon.

The scheduling problem of Chapter 3 particularly concerns the finite MDP, where the state and action variables is subject to the finite set. Formally speaking, MDP is described by a tuple $< S, A, T, R, \gamma >$. The nomenclature is given as follows: $S = \{1, \ldots, |S|\}$ is a set of discrete internal states. $A = \{1, \ldots, |A|\}$ is a set of actions that agent can take. $T : S \times A \times S \to [0,\ 1]$ is a stochastic state transition function, and $T(s, a, s') = p(s'|s, a)$ means the probability of the successor state being in s' when the current state is s and the action a is taken. $R : S \times A \times S \to \mathbb{R}$ is a single stage reward function, when the current state is s and the successor state is s' with action a taken. $\gamma \in [0, 1)$ is the discount factor to decrease the utility of later rewards.

POMDP is an extension of MDP to situations where observation uncertainty exists. The tuple structure for POMDP is extended to $< S, A, O, T, R, \Omega, \gamma >$. Additional features are $\Omega = \{1, \ldots, |\Omega|\}$, which is a set of discrete observations revealed to the agent, and an observation probability function $O : S \times A \times \Omega \to [0,\ 1]$. $O(a, s', o) =$

$p(o|s', a)$ means the probability of observing o when successor state is s' and action a is taken.

In the finite MDP framework, the well-known solution algorithm is the value iteration [36]. Denote the optimal policy as $\pi : S \rightarrow A$, which is a mapping of each state to the corresponding optimal action. The objective function or the value function is defined as the expected summation of the reward function R where any control policy π can achieve starting from state s_0.

$$V_\pi(s_0) = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \middle| \pi, s_0\right] \tag{2.1}$$

Eq. (2.2) is referred to as the Bellman equation for the infinite horizon discounted MDP, where $R(s, a)$ is a single stage expected reward.

$$\begin{aligned} V^*(s) &= \max_{a \in A}[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V^*(s')] \\ R(s,a) &= \sum_{s' \in S} T(s,a,s')R(s,a,s') \end{aligned} \tag{2.2}$$

The optimal value function $V^*(s)$ is the maximal expected total discounted reward. We note that in the infinite horizon formulation, V^* is time-invariant and its uniqueness and existence are proven when $0 \leq \gamma < 1$, and $R(s, a)$ is bounded [40]. Once V^* is available, an optimal policy π^* can be obtained by solving for a given s using Eq. (2.2).

In the POMDP framework, the internal state S cannot be observed deterministically. Instead, a belief space B is defined as a probability distribution over S. The complete system history tuple from initial time $\eta =< S, \Omega, A >$ should be known to determine b_t.

Thus a belief is expressed as follows:

$$b = p(s|h), h \in \eta \tag{2.3}$$

Similar to MDP, the solution of the infinite horizon POMDP is expressed as an optimal policy which maps beliefs b to optimal actions: $\pi : B \to A$. The objective function is the expected summation of the reward function R where the policy π starts at belief b_0:

$$V_\pi(b_0) = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R(b_t, \pi(b_t))|\pi, b_0\right] \tag{2.4}$$

When action a and observation o are received, the belief can be updated to $b_{a,o}$ by Bayes' rule,

$$b_{a,o}(s') = \frac{O(a, s', o)}{p(o|b, a)} \sum_{s \in S} T(s, a, s')b(s) \tag{2.5}$$

where the normalization constant $p(o|b, a)$ is defined as

$$p(o|b, a) = \sum_{s' \in S} O(a, s', o) \sum_{s \in S} T(s, a, s')b(s) \tag{2.6}$$

Since beliefs actually provide sufficient statistics for history h [80], the value function for POMDP can be stated as a function of b only. Moreover, since the belief $b_{a,o}$ only depends on the previous belief b using Eq. (2.5), POMDP can be considered as an augmented MDP with continuous state. The Bellman equation for POMDP can be derived by taking the additional expectation of Eq. (2.2) with respect to the belief $b(s)$. We omitted s' index of b in Eq. (2.7) since the state dependency of b is trivial.

$$\begin{aligned}
&V^*(b) \\
&= \max_{a \in A} \left[\sum_{s \in S} b(s) \Bigg(R(s,a) \right. \\
&\qquad \left. + \gamma \sum_{o \in \Omega} \sum_{s' \in S} O(a,s',o) T(s,a,s') V^*(b_{a,o}(s')) \Bigg) \right] \\
&= \max_{a \in A} \left[\sum_{s \in S} b(s) R(s,a) \right. \\
&\qquad \left. + \gamma \sum_{o \in \Omega} \sum_{s' \in S} O(a,s',o) \sum_{s \in S} b(s) T(s,a,s') V^*(b_{a,o}(s')) \right] \\
&= \max_{a \in A} \left[R(b,a) + \gamma \sum_{o \in \Omega} p(o|b,a) V^*(b_{a,o}) \right] \qquad (2.7)
\end{aligned}$$

$$R(b,a) = \sum_{s \in S} b(s) R(s,a) \qquad (2.8)$$

If the functional operator H is defined as,

$$(HV)(b) = \max_{a \in A} \left[R(b,a) + \gamma \sum_{o \in \Omega} p(o|b,a) V(b_{a,o}) \right] \qquad (2.9)$$

then the Bellman equation can be stated equivalently with a mapping through H. Then Eq. (2.7) can be written as $V^* = HV^*$, and it is proven that the Bellman operator H for POMDP is a contraction mapping [81].

2.1.2 State space model

State space model is used for the development of control-theoretic RL algorithms, where the first-principle based structured model is available. We consider three optimization problems whose settings are differ with respect to the finiteness of the horizon, continuity in the time domain, and control-affine and nonaffine state space model. The three problems are the subjects for Chapters 4, 5, and 6, respectively.

2.1.2.1 Finite horizon tracking control of discrete-time control-affine system

Consider a stochastic nonlinear control-affine system of

$$\begin{aligned} d\overline{x} &= \big(\tilde{f}(\overline{x}) + \tilde{g}(\overline{x})u\big)dt + dw \\ y &= h(\overline{x}), \quad \overline{x}(0) \sim p(\overline{x}_0), \quad \overline{x} \in \Omega \end{aligned} \tag{2.10}$$

where $\overline{x} \in \mathbb{R}^S$ is the state, $u \in \mathbb{R}^A$ is the control, $y \in \mathbb{R}^O$ is the output vector, and $p(x_0)$ is the probability distribution of initial state. $\Omega \subset \mathbb{R}^S$ denotes the compact state space. $\tilde{f}(x) \in \mathbb{R}^S$, $\tilde{g}(x) \in \mathbb{R}^{S\times A}$, and $h(x) \in \mathbb{R}^O$ are the smooth nonlinear functions that represent the system dynamics. The state dynamics is subject to the state transition uncertainty $dw \in \mathbb{R}^S$ which is independent of both state and control.

For the development and discussion of sampled-data control, it is convenient to express Eq. (2.10) in the discrete-time form of

$$\begin{aligned}\overline{x}_{k+1} &= \overline{f}(\overline{x}_k) + \overline{g}(\overline{x}_k)u_k + w_k \\ y_k &= h(\overline{x}_k), \quad \overline{x}_0 \sim p(\overline{x}_0), \\ & \overline{x} \in \Omega, \quad k \in \{0, \dots T-1\} \end{aligned} \tag{2.11}$$

where the subscript k denotes the sample time index and T is the time horizon. Note that operators $\overline{f}$, $\overline{g}$, and w denote the explicit integration of the ODEs for time interval $[t_k, t_k + \Delta t_k]$, either analytically or through their numerical approximations. For example, $\overline{f}$ and $\overline{g}$ may be defined through the Runge-Kutta (RK) integration equations, which are explicit but nevertheless include some intermediate variables for the calculation. In the case of w, the Euler-Maruyama method which is a stochastic version of Euler discretization can be used [82]. In constant reference tracking problems, it is common that delta-input $\Delta u_k = u_k - u_{k-1}$ and the augmented state space formulation. Additionally, to express the time dependency in FHOC problem, the time variable is augmented as $x_k = [\overline{x}_k, u_{k-1}, t_k]$ to define the new state space dynamics of

$$\begin{aligned}\overbrace{\begin{bmatrix}\overline{x}_{k+1} \\ u_k \\ t_{k+1}\end{bmatrix}}^{x_{k+1}} &= \overbrace{\begin{bmatrix}\overline{f}(\overline{x}_k) + \overline{g}(\overline{x}_k)u_{k-1} \\ u_{k-1} \\ t_k + \Delta t_k\end{bmatrix}}^{f(x_k)} + \overbrace{\begin{bmatrix}\overline{g}(\overline{x}_k) \\ I \\ 0\end{bmatrix}}^{g(x_k)} \Delta u_k + \overbrace{\begin{bmatrix} I \\ 0 \\ 0\end{bmatrix}}^{B} w_k \\ y_k &= h(\overline{x}_k, u_{k-1}, t_k), \quad \overline{x}(0) \sim p(\overline{x}_0), \quad \overline{x} \in \Omega \end{aligned} \tag{2.12}$$

to ensure integral action.

Representing the distribution function for the discrete-time state transition as $p(x_{k+1}|x_k, \Delta u_k, w_k)$, the likelihood of the state trajectory $x_{0:T} := \{x_0, x_1, \ldots, x_T\}$ is obtained by the successive multiplication of the probability distributions of the initial state and the state transitions as

$$p(x_{0:T}) = p(x_0) \prod_{k=0}^{T-1} p(x_{k+1}|x_k, \Delta u_k, w_k) \tag{2.13}$$

An objective function of the FHOC problem is defined as

$$J(x_0; \Delta u_{0:T-1}) = \mathbb{E}_{p(x_{0:T})}\left[\phi(x_T) + \sum_{i=0}^{T-1} r(x_i, \Delta u_i)\right] \tag{2.14}$$

where T is the terminal time index, $\phi(x_T)$ is the terminal cost and $r(x_k, \Delta u_k)$ is the stage-wise cost at time t_k. The expectation operator $\mathbb{E}[\cdot]$ is defined with respect to r the state trajectory likelihood $p(x_{0:T})$. The objective function measures the expectation value of the performance of a trajectory. The stage-wise cost and the terminal cost functions are positive definite, and quadratic as below:

$$r(x_k, \Delta u_k) := \|y_k - \rho_k\|_Q^2 + \|\Delta u_k\|_R^2, \quad k = 0, \cdots, T-1 \tag{2.15}$$

$$\phi(x_T) := \|y_T - \rho_T\|_H^2 \tag{2.16}$$

where $\rho_k = \rho(t_k)$ denotes the reference function for state $\bar{x}_k$, $\| \cdot \|$ is the weighted Euclidean norm, and Q, R, and H are the positive-definite weighting matrices.

The time varying 'cost-to-go' function V, also referred to as the value function is defined as

$$\begin{aligned} V(x_k; \Delta u_{k:T-1}) &:= J(x_k; \Delta u_{k:T-1}) \\ &= \mathbb{E}_{p(x_{k:T})}\left[\phi(x_T) + \sum_{i=k}^{T-1} r(x_i, \Delta u_i)\right] \end{aligned} \tag{2.17}$$

which measures the expected value of the objective function J starting from x_k at time k. By Bellman's principle of optimality, the optimal cost-to-go function satisfies the following recursive equation

$$\begin{aligned} V^*(x_k) = \min_{\Delta u_k} \mathbb{E}_{p(x_{k+1}|x_k, \Delta u_k, w_k)}\left[r(x_k, \Delta u_k) + V^*(x_{k+1})\right], \\ k = 0, \cdots, T-1 \end{aligned} \tag{2.18}$$

where $V^*(\cdot)$ denotes the optimal value function. Costate function is defined as the partial derivative of the value function with respect to the augmented state x_k.

$$\lambda^*(x_k) := \frac{\partial V^*(x_k)}{\partial x_k} \tag{2.19}$$

The optimality condition expressed in terms of the costate can be obtained by differentiating both sides of the Bellman equation with respect to the state x_k as in [61]

$$\lambda^*(x_k) = \mathbb{E}_{p(x_{k+1}|x_k, \Delta u_k, w_k)}\left[\frac{\partial r(x_k, \Delta u_k)}{\partial x_k} + \left(\frac{\partial x_{k+1}}{\partial x_k}\right)^T \lambda^*(x_{k+1})\right] \tag{2.20}$$

With the state independent assumption of the state transition uncertainty w_k,

$$\frac{\partial r(x_k, \Delta u_k)}{\partial x_k} = 2 \left(\frac{\partial h(x_k)}{\partial x_k} \right)^T Q(y_k - \rho_k) \tag{2.21}$$

Evaluation of Eqs. (2.20) and (2.21) requires the first-order differentiation of system dynamics functions $f(x_k)$, $g(x_k)$ and $h(x_k)$, which can be done either analytically or numerically.

The optimal control policy function $\pi^*(x_k)$ is a state feedback control law that minimizes the objective function. For the control-affine system with the quadratic cost function, the explicit form of the optimal control policy function can be obtained by the first-order optimality condition of Eq. (2.18) as

$$\begin{aligned} \pi^*(x_k) &= \arg\min_{\Delta u_k} \mathbb{E}_{p(x_{k+1}|x_k, \Delta u_k, w_k)} \Big[r(x_k, \Delta u_k) + V^*(x_{k+1}) \Big] \\ &= -\frac{1}{2} R^{-1} g^T(x_k) \mathbb{E}_{p(x_{k+1}|x_k, \Delta u_k, w_k)} \big[\lambda^*(x_{k+1}) \big] \end{aligned} \tag{2.22}$$

Substituting Eq. (2.22) into Eq. (2.18) yields the following discrete-time version of the Hamilton-Jacobi-Bellman (HJB) equation:

$$\begin{aligned} V^*(x_k, k) = \mathbb{E}_{p(x_{k+1}|x_k, \Delta u_k, w_k)} \Big[&\|y_k - \rho_k\|_Q^2 + \\ &\frac{1}{4} \lambda^*(x_{k+1}) g(x_k) R^{-1} g(x_k)^T \lambda^*(x_{k+1}) + V^*(x_{k+1}) \Big] \end{aligned} \tag{2.23}$$

$$V^*(x_T) = \phi(x_T), \qquad \lambda^*(x_T) = \partial \phi(x_T) / \partial x_T \tag{2.24}$$

whose solution is the optimal value function V^*.

2.1.2.2 Infinite horizon regulation control of discrete-time control-affine system

The second case studies the infinite horizon optimal control (IHOC) for nonlinear control-affine system defined in the discrete time of

$$x_{k+1} = f(x_k) + g(x_k)u_k, \quad x(0) = x_0, \quad x \in \Omega \tag{2.25}$$

where $x_k \in \mathbb{R}^S$ is the state and $u_k \in \mathbb{R}^A$ is the control vector. $\Omega \subset \mathbb{R}^S$ denotes the compact set of the state space. $f(x_k) \in \mathbb{R}^S$ and $g(x_k) \in \mathbb{R}^{S\times A}$ represent the system dynamics and assumed to be C^1 functions in Ω.

An objective function V is defined as the infinite-sum of the cost function as $V(x_0; u_{0:\infty}) = \sum_{k=0}^{\infty} \gamma^k r(x_k, u_k)$, where $r(x_k, u_k)$ is the cost function that occurs in time step k and $\gamma < 1$ is the discount factor. The cost function has the following quadratic form: $r(x_k, u_k) = Q\big(x_k\big) + u_k^T R u_k$, where $Q \in \mathbb{R}$ is a positive semi-definite (PSD) function and $R \in \mathbb{R}^{A\times A}$ is a positive definite matrix. The 'cost-to-go' function, also referred to as the value function is defined as

$$V(x_k; u_{k:\infty}) = \sum_{i=k}^{\infty} \gamma^{i-k} r(x_i, u_i) \tag{2.26}$$

which measures the objective value starting from time step k. By Bellman's principle of optimality, the optimal value function V^* satisfies the following relationship:

$$V^*(x_k) = \min_{u_k}\big(r(x_k, u_k) + \gamma V^*(x_{k+1})\big) \tag{2.27}$$

Costate function is defined as a derivative of the value function with respect to the state, i.e., $\lambda(x_k) = \dfrac{\partial V(x_k)}{\partial x_k}$. The optimality condition in terms of the costate equation can be obtained by differentiating both sides of the Bellman equation with respect to the state x_k as

$$\lambda^*(x_k) = \frac{\partial r(x_k, u_k)}{\partial x_k} + \left(\gamma \frac{\partial x_{k+1}}{\partial x_k}\right)^T \lambda^*(x_{k+1}) \tag{2.28}$$

The optimal control policy function $\pi^*(x_k)$ is a state feedback control which minimizes the objective function. In the control-affine system with the quadratic cost function, the explicit form of the optimal control policy function is obtained by the first order optimality condition of Eq. (2.27) as

$$\begin{aligned} \pi^*(x_k) &= \arg\min_{u_k}\big(r(x_k, u_k) + \gamma V^*(x_{k+1})\big) \\ &= -\frac{\gamma}{2} R^{-1} g^T(x_k) \lambda^*(x_{k+1}) \end{aligned} \tag{2.29}$$

Substituting Eq. (2.29) into Eq. (2.27) yields the following Hamilton-Jacobi-Bellman (HJB) equation of the discrete time version.

$$\begin{aligned} &V^*(x_k) = Q\big(x_k\big) + \\ &\frac{\gamma^2}{4} \lambda^*(x_{k+1})^T g(x_k) R^{-1} g(x_k)^T \lambda^*(x_{k+1}) + \gamma V^*(x_{k+1}) \end{aligned} \tag{2.30}$$

2.1.2.3 Constrained dynamic optimization of continuous-time system

In the final case, we consider the supervisory optimization defined in the continuous-time domain and the state space model and the cost function not being restricted to the control-affine system and quadratic function, respectively. Consider the dynamic optimization problem with minimizing the following objective function

$$J(u) = \phi(x(t_f), t_f) + \int_{t_0}^{t_f} L(x(t), u(t), t)dt \tag{2.31}$$

where $x \in \mathbb{R}^S$ is the state, $u \in \mathbb{R}^A$ denotes the control, t_f is terminal time, $L : \mathbb{R}^S \times \mathbb{R}^A \times \mathbb{R} \to \mathbb{R}$ denotes the path cost function and $\phi : \mathbb{R}^S \times \mathbb{R} \to \mathbb{R}$ denotes the terminal cost function. The optimization problem is subject to the following state space system dynamics with the initial condition,

$$\dot{x}(t) = f(x(t), u(t), t), \quad x_0 = x(t_0) \tag{2.32}$$

under the path constraints

$$g_k(x(t), u(t), t) \leq 0, \quad k = 1, \ldots, G \tag{2.33}$$

and the terminal constraints

$$h_k(x(t_f)) \leq 0, \quad k = 1, \ldots, H \tag{2.34}$$

$g_k : \mathbb{R}^S \times \mathbb{R}^A \times \mathbb{R} \to \mathbb{R}$ and $h_k : \mathbb{R}^S \times \mathbb{R} \to \mathbb{R}$ denote the path and terminal constraint functions and G and H denote the number of path and terminal constraints, respectively. Note that without loss of generality, only the inequality constraint is considered.

Constrained optimization is solved by the augmentation of the constraints to the cost function in various ways (e.g., Lagrangian method, penalty method, and barrier method). We denote the augmented path and terminal cost functions $\tilde{L}(x(t), u(t), \lambda(t), t) = L + \psi(\lambda, g)$ and $\tilde{\phi}(x(t_f), \mu, t_f) = \phi + \psi(\mu, h)$, respectively, where $\lambda(t) \in \mathbb{R}_+^G$ and $\mu \in \mathbb{R}_+^H$ denotes the Lagrangian dual variables for path and terminal constraints, and ψ is an arbitrary augmentation function. The augmented 'cost-to-go' objectives for the unconstrained problem starting from state $x(t)$ at time t is defined with the augmented cost functions of

$$\tilde{J}_{x,t}(u, \lambda, \mu) = \tilde{\phi}(x(t_f), \mu, t_f) + \int_{t_0}^{t_f} \tilde{L}(x(t), u(t), \lambda(t), t) dt \quad (2.35)$$

This yields the optimization problem $V(x(t), t) = \min_u \max_{\lambda,\mu} \tilde{J}_{x,t}(u, \lambda, \mu)$ for which the minimum and maximum operators are defined with respect to the primal (control) and the dual (Lagrangian) variables, respectively.

According to the dynamic programming principle, we have a equation for the value function V, which is referred to as Hamilton-Jacobi-Bellman (HJB) equation

$$-\frac{\partial V(x,t)}{\partial t} = \min_u \max_\lambda \mathcal{H}(x, u, \lambda, t) \quad (2.36)$$

with the boundary condition

$$V(x, t_f) = \max_\mu \tilde{\phi}(x(t_f), \mu, t_f) \quad (2.37)$$

where the Hamiltonian is given as

$$\mathcal{H}(x, u, \lambda, t) = \tilde{L}(x, u, \lambda, t) + \frac{\partial V(x,t)}{\partial x} \cdot f(x, u, t) \quad (2.38)$$

2.2 Overview of the developed RL algorithms

The optimization problems presented in Section 2.1 are the POMDP in finite countable state set, discrete-time FHOC and IHOC for control-affine system, and continuous-time supervisory optimization problems. All problems have different forms of Bellman's principle of optimality, given as Eqs. (2.7), (2.23), (2.30), and (2.36), respectively. Unfortunately, none of which has a tractable method to obtain the solution that satisfies the Bellman equation since the belief domain B or the state domain Ω are the uncountable set. Note that the exception cases that allows for the tractable method or the analytic solution of the Bellman equation are only in the finite state set or in the linear system. Based on the observation, the approximate solution methods only search for the subset that the state trajectory lies. The overview of the sample based methods, also referred to as RL is described in this section.

2.2.1 Point based value iteration

The RL method for POMDP focuses on the special structure of POMDP with respect to the belief b. Eq. (2.7) shows that the optimal value function is a function of the belief b. The belief domain B is the simplex domain, which makes the exact value iteration intractable. As an alternative form, the value function can be represented as a piecewise linear and convex (PWLC) function [80]:

$$\alpha(s) = R(s,a) + \gamma \sum_{o\in\Omega} \sum_{s'\in S} O(a,s',o)T(s,a,s')V(b_{a,o}(s')) \tag{2.39}$$

$$V^*(b) = \max_{\alpha\in\Gamma} \sum_{s\in S} b(s)\alpha(s) \tag{2.40}$$

$\alpha(s)$ is the gradient of the value function at any belief point and called 'alphavector', and Γ is the set of alphavectors. Since the belief space is infinite, Γ is an infinite set. With this representation, the solution algorithm for POMDP focuses on the efficient computation of Γ.

Although the alphavector (α) representation in Eq. (2.40) shows the PWLC structure of the value function for POMDP, the computation of the alphavector set throughout the whole belief space still remains intractable. An important contribution to state-of-the-art POMDP research follows by pruning down the infinite alphavector set into a finite parsimonious subset, maintaining the structure of the value function. The so-called Point Based Value Iteration (PBVI) algorithm performs the value iteration only in the finite sets of sampled belief and alphavector [83]. Detailed implementation of PBVI algorithm is presented in Section 3.2.1.

2.2.2 Globalized dual heuristic programming

The nonlinear PDE of HJB (Eq. (2.23)) does not yield an analytical solution in general, except for the linear quadratic optimal control problem where $V^*(x_k) = \frac{1}{2}x_k^T P_k x_k$. The PDE can be transformed into ODE with respect to P_k, which is the well-known Riccati difference equation. However, in the nonlinear system, the PDE cannot be transformed to an ODE because of the function $g(x_k)$, thus motivating the development of an approximate numerical solution method.

The developed numerical algorithm for solving the HJB equation involves an successive, iterative evaluation of the value, costate, and policy functions. The algorithm is not new and is referred to as the GDHP algorithm in the literature [61]. Instead of solving Eq. (2.23) directly, the algorithm sequentially updates the individual optimality conditions with respect to the value, costate, and policy functions computed by the measurement data $(x_k, \Delta u_k, r_k, x_{k+1})$ obtained at each time step k.

Assume that there exists an admissible control policy μ which satisfies $V(x_0; \mu) < \infty$ [61]. The iterative scheme of GDHP is written as follows: The functions are initialized as

$$V^{(0)} = 0, \quad \lambda^{(0)} = \mathbf{0}_S, \quad \pi^{(0)} = \mu(x_k) \tag{2.41}$$

where $\mathbf{0}_n$ denotes a zero vector with n entries. At time step k and the iteration step i, state and output of the next time step are computed with the system dynamics under the policy $\pi^{(i)}(x_k)$ and the additive state noise w_k.

$$\begin{aligned} x_{k+1} &= f(x_k) + g(x_k)\pi^{(i)}(x_k) + Bw_k &\quad (2.42)\\ y_k &= h(x_k) &\quad (2.43)\\ r_k &= \left\|y_k - \rho_k\right\|_Q^2 + \left\|\pi^{(i)}(x_k)\right\|_R^2 &\quad (2.44) \end{aligned}$$

A measurement data tuple $(x_k, \Delta u_k, r_k, x_{k+1})$ is stored to evaluate the value, costate, and policy functions in the next iteration step.

At the i^{th} iteration step, the value and costate functions are updated by using the optimality equations (Eqs. (2.18) and (2.20)), in which the minimum values of the right-hand-sides are evaluated by the optimal policy used in the i^{th} iteration, $\pi_k^{(i)} = \pi^{(i)}(x_k)$ [61].

$$V^{(i+1)}(x_k) = r(x_k, \pi_k^{(i)}) + V^{(i)}(x_{k+1}) \tag{2.45}$$

$$\lambda^{(i+1)}(x_k) = \frac{\partial r(x_k, \pi_k^{(i)})}{\partial x_k} + \left(\frac{\partial x_{k+1}}{\partial x_k}\right)^T \lambda^{(i)}(x_{k+1}) \tag{2.46}$$

Finally, the policy function of the next iteration step is obtained with the updated costate function.

$$\pi^{(i+1)}(x_k) = -\frac{1}{2}R^{-1}g^T(x_k)\lambda^{(i+1)}(x_{k+1}) \tag{2.47}$$

Note that the expectation with respect to $p(x_{k+1}|x_k, \Delta u_k, w_k)$ in the optimality equations of the value, costate, and policy functions are estimated with a single sampled data in the GDHP algorithm, as in the Monte-Carlo fashion.

In the case of IHOC problem with discount factor γ, the sequential updates for value, costate, and policy functions are slightly modified as:

$$V^{(i+1)}(x_k) = r(x_k, \pi_k^{(i)}) + \gamma V^{(i)}(x_{k+1}) \tag{2.48}$$

$$\lambda^{(i+1)}(x_k) = \frac{\partial r(x_k, \pi_k^{(i)})}{\partial x_k} + \left(\gamma \frac{\partial x_{k+1}}{\partial x_k}\right)^T \lambda^{(i)}(x_{k+1}) \tag{2.49}$$

$$\pi^{(i+1)}(x_k) = -\frac{\gamma}{2}R^{-1}g^T(x_k)\lambda^{(i+1)}(x_{k+1}) \tag{2.50}$$

The above-mentioned iteration scheme is performed until the value, costate, and policy functions converge. Since expectations are computed by the Monte-Carlo method with single sampled data per iteration, the procedure typically requires large amounts of data for convergence [36]. In view of this, any model knowledge incorporated into the iteration procedure (e.g. $\frac{\partial x_{k+1}}{\partial x_k}$ and $g(x_k)$) can be helpful. The iteration period can be set arbitrarily with respect to the time interval, such that the algorithm can be implemented either on a run-to-run basis or in batch mode where iteration is performed after data from several runs are collected. In this study, we perform iteration at every sample time.

GDHP is in fact a generalized version combining the ideas of heuristic dynamic programming (HDP) and dual heuristic programming (DHP) [61]. In HDP, only the equations for the value and policy functions, (Eqs. (2.48) and (2.50)), are used whereas only those for the costate and policy functions, (Eqs. (2.49) and (2.50)), are employed in DHP. Since the costate function can be differentiated directly from the value function in HDP, the system dynamics function $f(x_k)$ is not required. However, errors in the value function can be amplified through differentiation, which can cause unstable learning behavior throughout the HDP algorithm.

2.2.3 Differential dynamic programming

Differential dynamic programming (DDP) algorithm is the sequential quadratic approximation method for solving the HJB equation Eq. (2.36). The modifications of the fundamental partial differential equation (PDE) of HJB are threefold: First, the decision variables are changed from u and λ to the deviations from their nominal (or previous) values, i.e., δu and $\delta\lambda$, second, the objective for min-max operator becomes the second order Taylor expansion of the Hamiltonian $\mathcal{H}$, and finally, the PDE form is transformed to series of ODEs with respect to value function and its first and second order derivatives. Under which the DDP modifications are taken, the backward and the forward sweeps are iteratively performed until convergence. In the backward sweep, given the boundary condition Eq. (2.37), ODEs are solved backward-in-time, whereas in the forward sweep, the optimal deviations δx, δu, and $\delta\lambda$ are obtained forward-in-time. The schematic description of the iteration is given in Fig. 2.2.

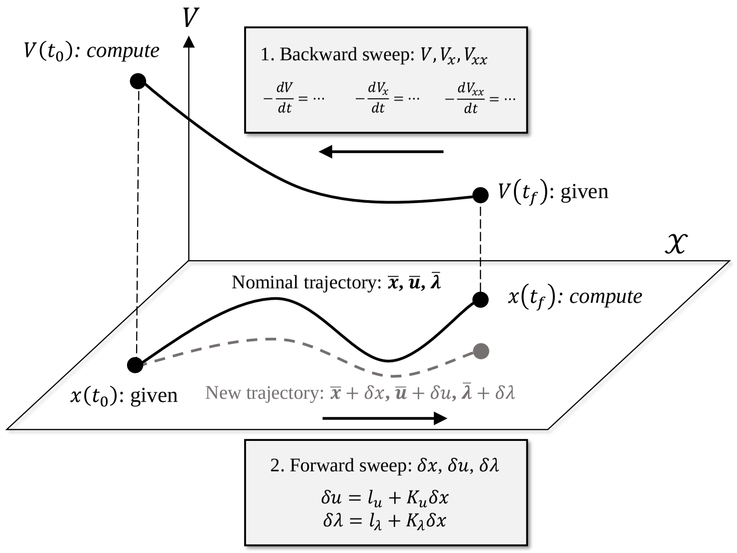

V
$V(t_0)$: *compute*
1. Backward sweep: V, V_x, V_{xx}
$-\frac{dV}{dt} = \cdots$ $-\frac{dV_x}{dt} = \cdots$ $-\frac{dV_{xx}}{dt} = \cdots$
$V(t_f)$: given
$\mathcal{X}$
Nominal trajectory: $\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{\lambda}}$
$x(t_f)$: *compute*
$x(t_0)$: given
New trajectory: $\bar{\boldsymbol{x}} + \delta x, \bar{\boldsymbol{u}} + \delta u, \bar{\boldsymbol{\lambda}} + \delta\lambda$
2. Forward sweep: δx, δu, $\delta\lambda$
$\delta u = l_u + K_u \delta x$
$\delta\lambda = l_\lambda + K_\lambda \delta x$

Figure 2.2: Backward and forward sweep in DDP algorithm.

Chapter 3

A POMDP framework for integrated scheduling of infrastructure maintenance and inspection [1]

3.1 Introduction

According to the report of the America's infrastructure grades in 2013 [85], the overall grade of the infrastructure was diagnosed as D+ (poor) and the required investment in infrastructure upgrades and maintenance by 2020 was estimated to be \$3.6 trillion. Infrastructure scheduling plays a critical role in ensuring safe operation and economic maintenance for chemical processes and process system peripherals. Infrastructure scheduling includes three major tasks: maintenance, inspection, and sensor-installation. Maintenance is done to improve the overall functionality of the system such as grade and failure rate. Inspection is carried out to assess the current condition and gather the information of the system. Sensor-installation allows inspection using a device rather than human senses. Proper and timely installation of sensors can enhance the quality of service (QoS) of a

[1]This chapter is an adapted version of J. W. Kim, G. B. Choi, J. C. Suh, and J. M. Lee, “Dynamic optimization of maintenance and improvement planning for water main system: Periodic replacement approach," *Korean Journal of Chemical Engineering*, vol. 33, no. 1, pp. 25–32, 2016. [84] and J. W. Kim, G. B. Choi, and J. M. Lee, “A POMDP framework for integrated scheduling of infrastructure maintenance and inspection," *Computers & Chemical Engineering*, vol. 112, pp. 239–252, 2018. [78]

system by structural health monitoring (SHM) [86, 87].

Infrastructure has two distinguishing characteristics which hinder rational decision-making. First, it deteriorates with a low failure rate whereas serious damage to a wide area is inevitable when it fails. Second, it is nearly impossible to measure or estimate the system's state in real time due to the size and complexity of the system, the high cost of scheduling actions, and the intrinsic uncertainty of non-destructive inspection methods.

A scheduling approach based on deterioration model and optimization can be a promising tool to address these two issues, and thus can lead to economical and sustainable operation. In the case of the infrastructure network, a prioritization method can be applied to narrow the scope of the scheduling problem into the single infrastructure system [88, 89]. The majority of deterioration models for infrastructure system consider uncertainties due to the lack of knowledge about fundamental principles and the limited amount of available data. Discrete-time states and actions are usually appropriate for stochastic optimization models. The discrete system state is often designated as a grade or condition of the system, and its validity is discussed in [90]. For this reason, the infrastructure scheduling problem can be effectively regarded as a discrete stochastic sequential decision process. An appropriate framework for this problem is Markov Decision Process (MDP), whose objective is to calculate an optimal maintenance schedule within a control horizon [40].

State transition randomness has been successfully modeled in the MDP framework via probability matrix [91, 92]. However, observation uncertainty leads to suboptimal solutions to MDP. Moreover, inspection and sensor-installation, which are other essential parts of the infrastructure scheduling, cannot be considered in the MDP framework which inherently assumes the 'fully observable state'. However, Partially Observable MDP (POMDP) formalism, which is a natural extension of MDP and involves probability of state observation, can tackle the problem of the uncertainty in observation and allow for integrating inspection scheduling and sensor-installation.

Several studies of the infrastructure scheduling using POMDP model have been conducted. Small-scale problems with the number of states less than 10 are solved with POMDP formulation [93, 94, 95]. Jiang [95] suggested failure criteria by comparing minimum resistance and maximum loading effect on the system. Byon [96] solved a finite horizon POMDP problem with season-dependent parameters.

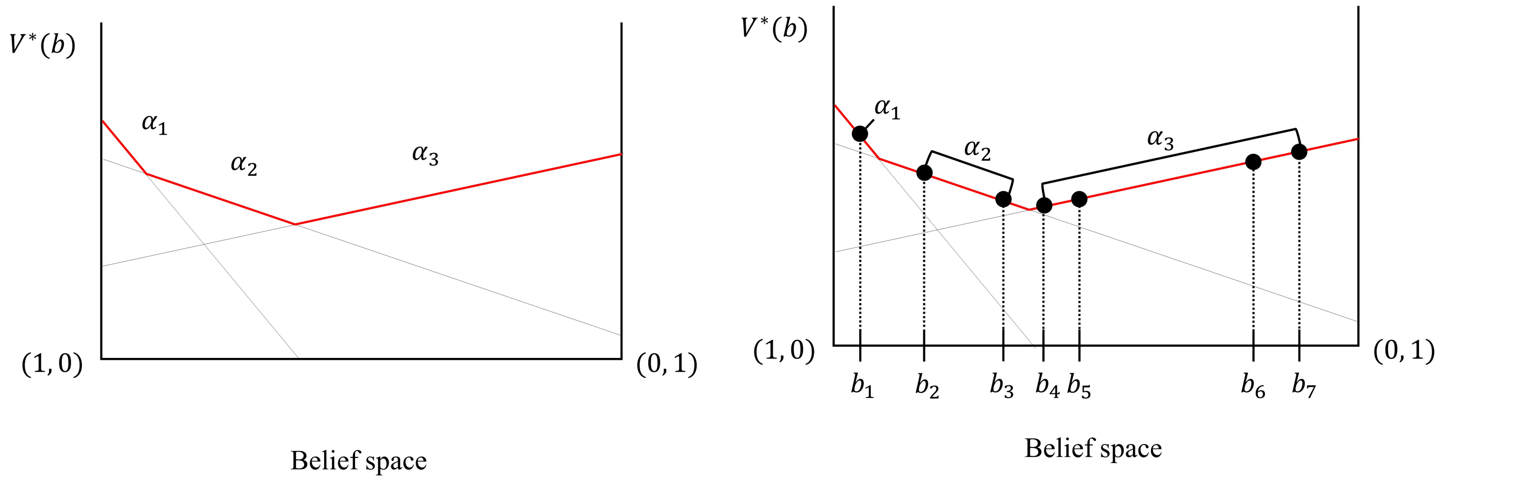
The history-dependent and time-variant transition process improves the accuracy of the model because it can take account of maintenance records and system's age. The time-variant transition model can be obtained by survival analysis with the semi-Markov assumption [97]. History-dependency can be modeled by the state-augmentation [98] or by the concept of periodic replacement [84]. In [99] a large-scale POMDP problem is solved for the system having the aforementioned issues. Whereas the large-scale POMDP could describe more practical systems, the previous studies could not either consider the sensor installation or revise the resulting policy when feedback information is available.

Water distribution pipe is analyzed as an illustrative example, and additional analysis using structural point gives an intuitive result for the value function and system dynamics where the optimal policy is implemented. Monte Carlo simulation result shows that the suggested receding horizon POMDP framework for the infrastructure management gives a better result than previous POMDP, MDP, myopic, and heuristic policies. The suggested POMDP scheduling framework can be generalized to other systems which require the receding horizon policy such as chemical process scheduling or experimental design.

3.2 POMDP solution algorithm

3.2.1 General point based value iteration

Figure 3.1 illustrates how PBVI is performed in a simple two state example. Fig. 3.1(a) shows the exact value function over the entire belief space. It is indicated that $\{\alpha_1, \alpha_2, \alpha_3\}$ are sufficient subset of Γ to represent the exact value function. Although Γ is expected to be an infinite set due to the continuity of the belief space, most of the alphavectors are redundant for the value function representation since the value function is PWLC. Given an arbitrary belief subset $\{b_1, \ldots, b_7\}$ as in Fig. 3.1(b), Γ is estimated through the calculation of the gradients of value function at each belief point. As with the alphavector set reduction, the belief set can be reduced by up to the same size as the alphavector set (i.e., $\{b_1, b_3, b_7\}$ or $\{b_1, b_2, b_5\}$). The PBVI solution algorithm aims to obtain the minimum cardinality of alphavector set and belief set without loss of the value function approximation performance.



(a) (b)

Figure 3.1: Value function over the entire belief space in two states system. (a) Exact alphavector representation. (b) Approximated representation with the finite subset of alphavector.

Generally, the lower bound of the value function $\underline{V}(b)$ is obtained at the parsimonious subset of the belief space $\tilde{B} \subset B$ with the subset of alphavector $\tilde{\Gamma} \subset \Gamma$. Since the alphavector representation is the PWLC function which evaluates the maximizer of the lower bound function approximator set. Given belief subset $\tilde{B}$, the alphavector set $\tilde{\Gamma}$ is evaluated iteratively similar to the value iteration method of MDP. Let the alphavector set $\tilde{\Gamma}_i$ come from the previous iteration step i. New alphavector set $\tilde{\Gamma}_{i+1}$ is calculated by:

$$\begin{aligned}
\alpha_{a,o}(s) &= \arg\max_{\alpha\in\tilde{\Gamma}_i} \sum_{s'\in S} O(a,s',o) \sum_{s\in S} T(s,a,s')b(s)\alpha(s'), \quad b \in \tilde{B} \\
&= \arg\max_{\alpha\in\tilde{\Gamma}_i} b_{a,o}\cdot\alpha \\
\alpha_a(s) &= R(s,a) + \gamma \sum_{o\in\Omega}\sum_{s'\in S} O(a,s',o)T(s,a,s')\alpha_{a,o}(s') \\
\alpha_b &= \arg\max_{\alpha_a} \sum_{s\in S} b(s)\alpha_a(s) \\
\tilde{\Gamma}_{i+1} &= \tilde{\Gamma}_i \cup \{\alpha_b\} \qquad (3.1)
\end{aligned}$$

The calculation of the lower bound $\underline{V}(b_0)$ and alphavector α_{b_0} at an arbitrary belief point $b_0 \in B$ is performed in the same manner with Eq. (2.40):

$$\begin{aligned}
\underline{V}(b_0) &= \max_{\alpha\in\tilde{\Gamma}} \sum_{s\in S} b_0(s)\alpha(s) \quad b_0 \in B \\
\alpha_{b_0} &= \arg\max_{\alpha\in\tilde{\Gamma}} \sum_{s\in S} b_0(s)\alpha(s) \qquad (3.2)
\end{aligned}$$

To distinguish from the original Bellman operator H, the PBVI Bellman operator in Eq. (3.2) is denoted as H_{PBVI}.

The upper bound of the value function $\overline{V}(b)$ is defined as a function that is globally larger than or equal to the exact function. The tightest upper bound is the minimum among the approximator set. However, in the PBVI algorithm, the value function evaluation is limited only at the belief points in the parsimonious belief set $\tilde{B}$. Thus, the convex hull of the approximators is considered as an interpolation between the elements of $\tilde{B}$ for the efficient upper bound approximation. The function $\overline{Q}(s,a)$ is an immediate and naive approximation for the upper bound of the value function, which is referred to as the fast informed bound [81] and calculated as:

$$\begin{aligned}\overline{Q}_0(s,a) &= \max_{s\in S}\max_{a\in A} R(s,a)/(1-\gamma)\\ \overline{Q}_{i+1}(s,a) &= R(s,a)+\gamma\sum_{o\in\Omega}\max_{a'\in A}\sum_{s'\in S} O(a,s',o)T(s,a,s')\overline{Q}_i(a',s'),\\ & \quad s\in S, a\in A \end{aligned}\tag{3.3}$$

where $\overline{Q}_0(s,a)$ is the initial guess and Eq. (3.3) is calculated iteratively until it converges to $\overline{Q}(s,a)$. The upper bounds of the value function at the elements of parsimonious belief set $\tilde{B}$, $v(b)$ are evaluated by:

$$v(b)=\max_{a\in A}\sum_{s\in S} b(s)\overline{Q}(s,a), \quad b\in\tilde{B} \tag{3.4}$$

Let $\tilde{\Upsilon}=\{(b,v(b)) : b\in\tilde{B}\}$ denotes the belief-bound pair set. Then the convex hull is obtained with $\tilde{\Upsilon}$:

$$\mathrm{conv}(b,v)=\{\sum_i c_i(b_i,v_i(b_i)) : \forall c_i\geq 0, \sum_i c_i=1, (b_i,v_i(b_i))\in\tilde{\Upsilon}\} \tag{3.5}$$

The calculation of the upper bound $\overline{V}(b_0)$ at an arbitrary belief point b_0 is obtained as a projection onto the convex hull (i.e., $\overline{V}(b_0)$ =

$\operatorname{proj}_v \operatorname{conv}(b, v)|_{b=b_0})$ which can be formulated as a linear programming problem:

$$\begin{aligned} \overline{V}(b_0) = \quad \min_{c_i} \quad & \sum_i c_i v_i(b_i) \\ s.t. \quad & \sum_i c_i b_i = b_0 \\ & \sum_i c_i = 1, \quad \forall c_i \geq 0 \end{aligned} \tag{3.6}$$

The lower bound and upper bound approximation and their update process are illustrated in Figs. 3.2(a), (b), respectively.

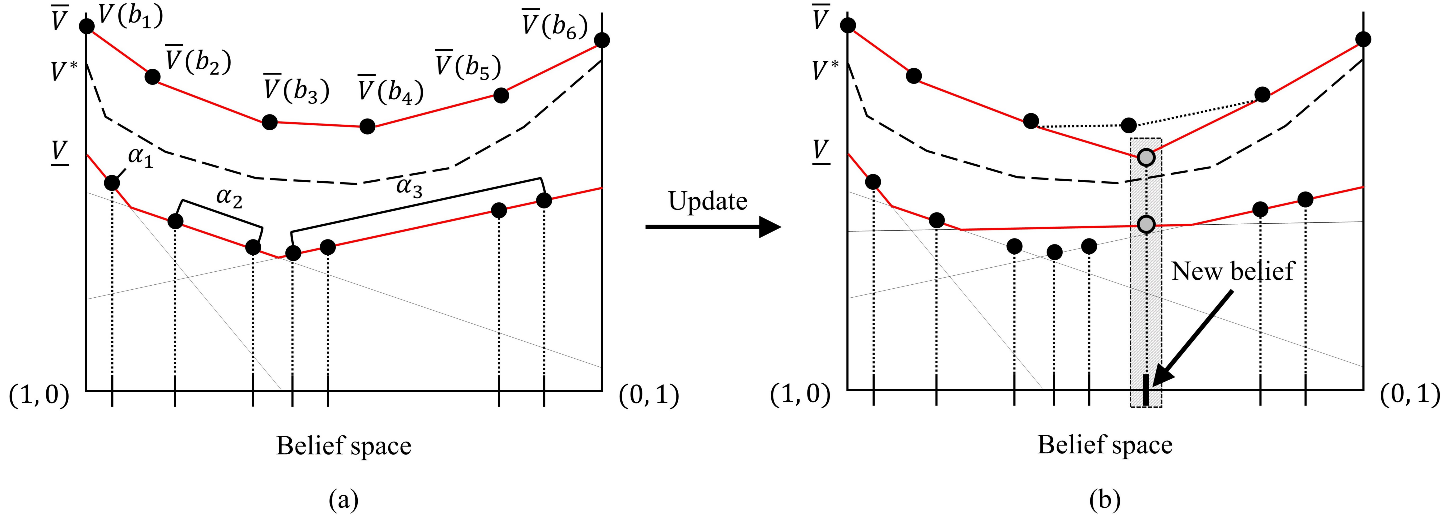


Figure 3.2: (a) Value function approximation using the lower and upper bounds of optimal value function. (b) Lower bound and upper bound update given new belief.

General point-based POMDP solvers involve a two-step approximation scheme consisting of: belief collection step and value iteration step. First, a finite belief subset $\tilde{B}$ is collected. The composition of the finite belief set is conceptually equivalent to the result of pruning down the belief space of infinite size. However, it is actually performed through the bottom-up collection of belief set. One possible method to obtain the belief subset $\tilde{B}$ is to construct the reachable belief set generated from the initial belief b_0. Given the initial belief b_0, reachable belief set is defined as belief points that are generated by simulating several actions and observations following the searching rule. A policy tree in Fig. 3.3 is a visualization of the reachable belief set $\tilde{B}$ which is spanned from the root node b_0. Among the reachable belief set in policy tree, several beliefs which are expected to provide the best value function approximation are collected. The objective of belief collection algorithms is to select trajectory that can approximate a value function with the least number of beliefs.

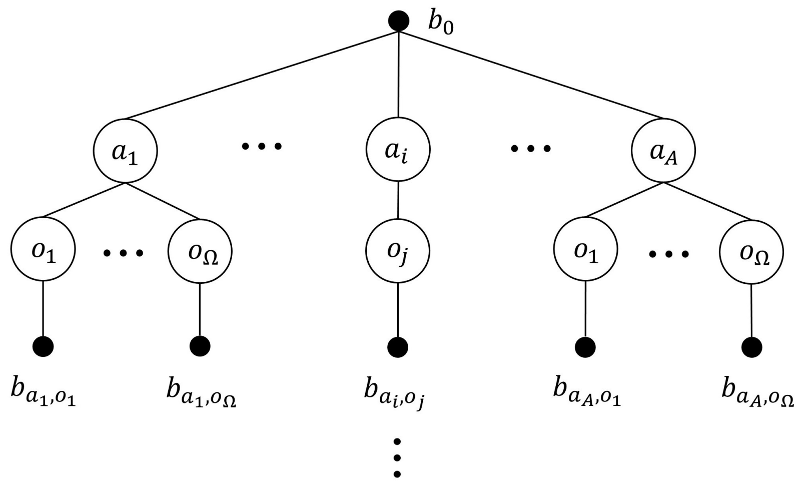


Figure 3.3: Policy tree spanned from the root node, which is generally considered as initial belief b_0. Each solid circle represents a reachable belief and each open circle represents an action a and an observation o.

Second, the lower ($\underline{V}$) and upper ($\overline{V}$) bounds of the value function approximation are improved through the value iteration. With the collected belief subset $\tilde{B}$ from the belief collection step, procedure for the value iteration of the lower bound (Eq. (3.1)) and of the upper bound (Eqs. (3.3)-(3.5)) are performed. These two steps are carried out until the gap between the lower bound and the upper bound meets the convergence criterion. Ideally, the gap should be shrunk to zero at the whole belief space B or at least at the parsimonious belief subset $\tilde{B}$. But practically, [100] reported that it is sufficient to obtain a good solution by examining the convergence only at the target belief point b_0 (i.e., $\overline{V}(b_0) - \underline{V}(b_0)$). $\overline{V}(b_0)$ and $\underline{V}(b_0)$ are evaluated by Eq. (3.2) and Eq. (3.6), respectively.

The idea of PBVI is to consider the 'structure-specific' beliefs only so that the candidate belief space can be greatly reduced. It is shown that the reachable belief set is enough to obtain the value function approximation in the various point-based methods [101, 102, 103, 104, 105]. Searching and collecting rules are important features in determining the convergence performance of the PBVI algorithm.

3.2.2 GapMin algorithm

Among various point-based methods, we adopt a state-of-the-art point-based solver called gapMin proposed in [105]. Like most of the belief collection algorithms of point-based solvers, a heuristic approach for spanning the policy tree is used in gapMin [100]. Assume that the policy tree $\tilde{B}$ spanned from the root node b_0 and a terminal node set $\tilde{B}_\tau$ are given. The algorithm first determines the spanning node $b_\tau \in \tilde{B}_\tau$ defined as the location to which the policy tree spans, and then it spans the tree to acquire a new policy tree $\tilde{B}'$ and a new terminal node set $\tilde{B}'_{\tau}$. Both procedures are performed with the heuristic of 'Improve the worst part first'. By giving priority of the improvement to the belief point that shows the worst approximation performance, the approximation throughout the belief space converges quickly.

Spanning node b_{τ} is selected based on the gap between the lower bound and upper bound of the value function approximation which is inversely proportional to the approximation performance. The spanning node selection rule is then described in Eq. (3.7) where h is the depth of the belief node in the policy tree.

$$b_{\tau} = \arg\max_{b \in \tilde{B}_{\tau}} \gamma^{h}(\overline{V}(b) - \underline{V}(b)) \tag{3.7}$$

Successor action a_{τ} for b_{τ} is chosen to yield the greatest one-step look ahead upper bound:

$$a_{\tau} = \arg\max_{a \in A} R(a, b_{\tau}) + \gamma \sum_{o \in \Omega} p(o|b_{\tau}, a)\overline{V}(b_{\tau,a,o}) \tag{3.8}$$

All observations are sampled to be the candidates of the new terminal node.

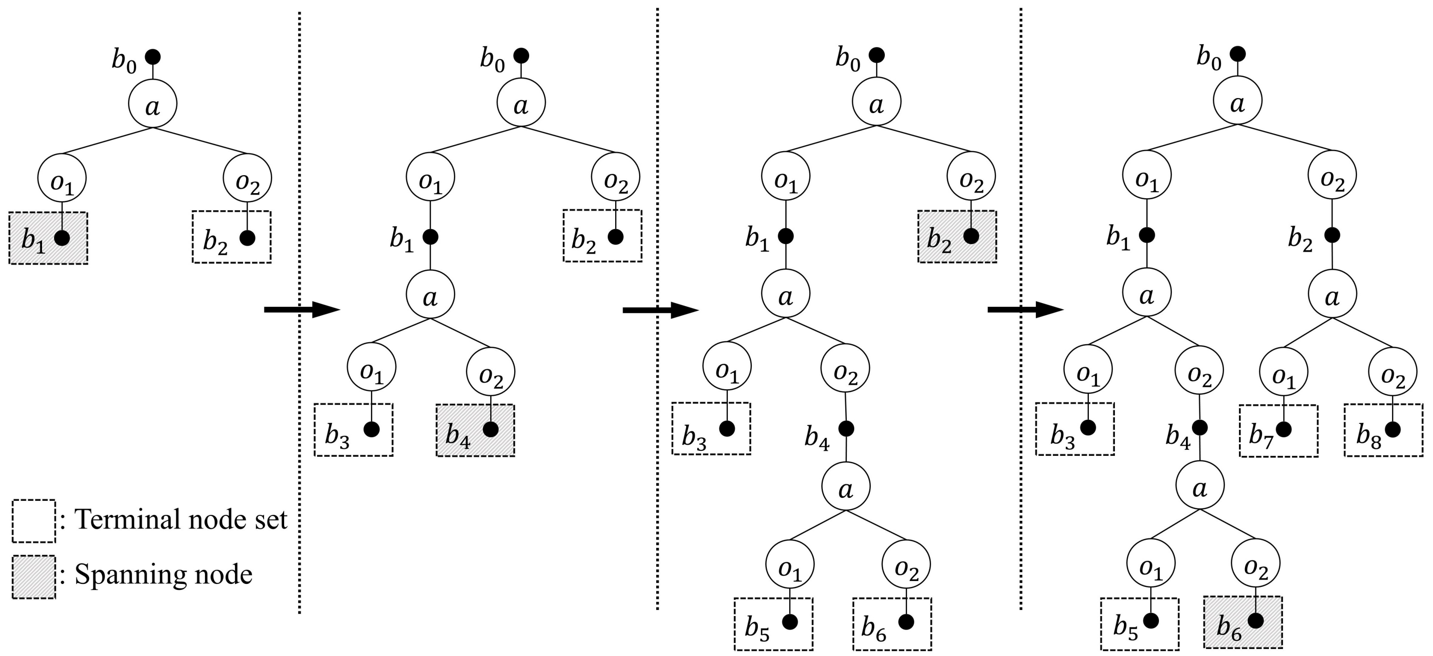


Figure 3.4: Procedure of the belief collection step and the corresponding policy tree spanning. The figure shows the first four steps of the procedure.

Since the previous spanning node b_τ has been spanned to become an internal node, it is deleted from the updated terminal node set. The terminal node set is updated to $\tilde{B}'_\tau$:

$$\tilde{B}'_\tau = (\tilde{B}_\tau - b_\tau) \cup \{b_{\tau,a_\tau,o} : o \in \Omega\} \tag{3.9}$$

Fig. 3.4 shows the procedure of the policy tree spanning where the observation set is $\Omega = \{o_1, o_2\}$. When new belief set $\tilde{B}'$ is collected in the collection step, the lower bound and upper bound of the value function are improved through the value iteration procedure.

The convergence property of the heuristic algorithms for belief collection is discussed in [102]. They proved that the Bellman operator for the PBVI algorithm H_{PBVI}, which is defined only for the finite belief set $\tilde{B}$, is the contraction mapping under the norm measuring the density of $\tilde{B}$. The computational complexity of the algorithm is $\mathcal{O}\left(|O||S|log(|\tilde{B}| + |A||S| + |A||\overline{V}| + |\underline{V}|)\right)$ [100]. Details of the algorithm are shown in [105].

3.2.3 Receding horizon POMDP

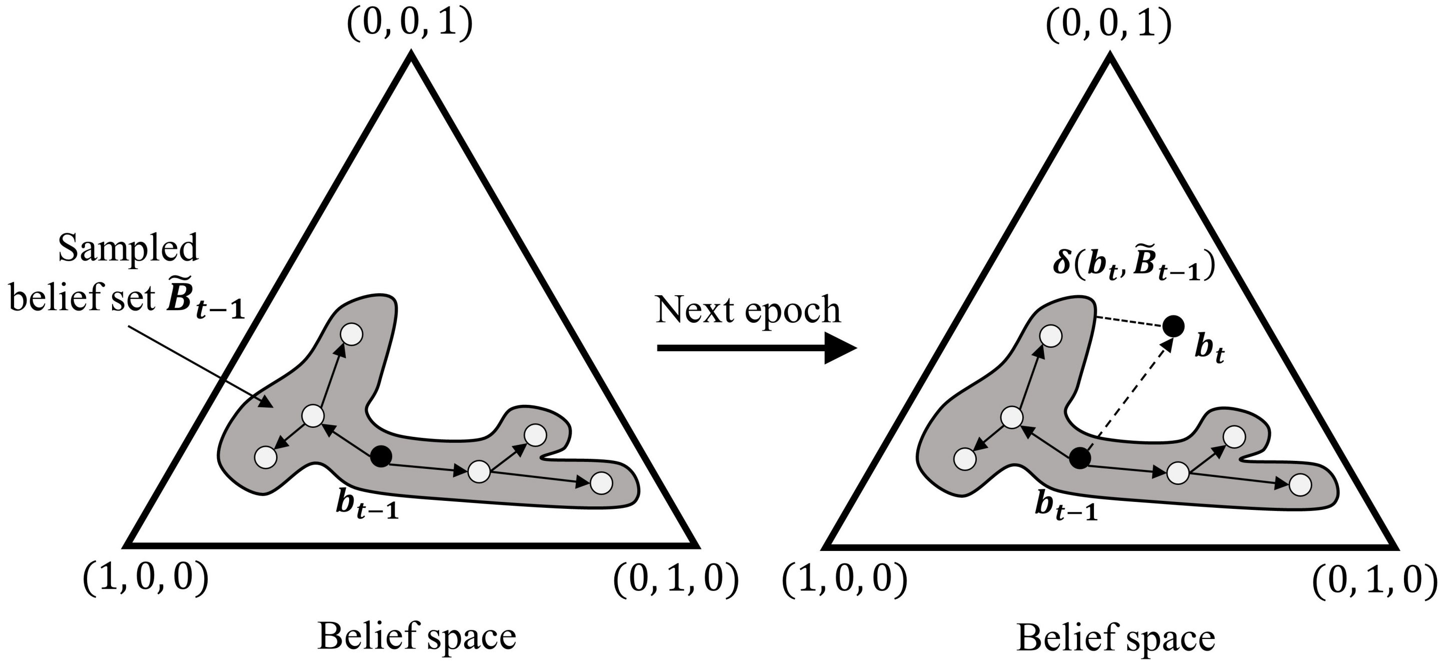


Figure 3.5: The incompatibility between the sampled belief set and the target belief point. Note that a ternary plot is used for representing the belief space of the three-state system.

The solution of a POMDP problem has two challenges for the application to infrastructure scheduling. First, in the case of the finite horizon formulation, the computational load for evaluating the value functions for each decision epoch is high. Moreover, since the infrastructure scheduling problem has no terminal or absorbing state, it is impossible to define the horizon explicitly. Second, as for the infinite horizon problem, although the infrastructure scheduling problem is well formulated, inefficiency or even suboptimality arises due to the algorithmic nature of PBVI. Consider the infinite horizon MDP given a perfect model. A complete closed-loop feedback rule satisfying the Bellman's optimality can be successfully obtained by value or policy iteration, and the resulting policy is time-invariant. However, in the case of the infinite horizon POMDP, the resulting policy is time-variant since the sampled belief space $\tilde{B}$ only covers the adjacent region of the target belief point. Fig. 3.5 shows the time-variant issue of the value function approximation for the three-state system. Suppose that on belief b_{t-1} at decision epoch $t-1$, a well-approximated value function based on the sampled belief set $\tilde{B}_{t-1}$ is given. At the next decision epoch, since the new belief b_t is located outside the previous sample belief set $\tilde{B}_{t-1}$, the approximation performance is significantly degraded. Thus, when calculating the policy, another feedback rule for updating the root node regarding the newly obtained belief should be established.

Receding horizon control (RHC) is a framework that can address these two issues. Instead of solving an infinite horizon POMDP problem, RHC recursively solves the optimization and implements the first action only. The feedback rule for the receding horizon POMDP is to assign the newly obtained belief to the root node for each sub-problem. By reconstructing the policy tree and belief set at each decision epoch, the resulting alphavector can provide a tighter bound for the current belief (i.e., $\underline{V}(b_t)$) than using the fixed root node b_0. This can be explained by the contraction property of the PBVI Bellman operator H_{PBVI} which holds uniformly for $\forall b \in B$. Consider the two cases where the value function approximation at current belief point $\underline{V}(b)$ is directly computed for each of two subsets $\tilde{B}$ and $\tilde{B}' = \tilde{B} \setminus \{b\}$, respectively. Let two PBVI Bellman operators be H_{PBVI} and H'_{PBVI}. Then H_{PBVI} gives a tighter bound at b, i.e., $H'_{PBVI}\underline{V}(b) \leq H_{PBVI}\underline{V}(b) \leq V^*(b)$.

It does not imply that assigning a different value with the target belief b to the root node always leads to the suboptimality. Because the contraction property guarantees the tighter bound only if the two belief sets $\tilde{B}$ and $\tilde{B}'$ are identical except for the current belief. When the belief set collected at the previous decision epoch $\tilde{B}_{t-1}$ still provides a good bound for the current belief b_t, the choice of root node value becomes less significant in terms of the result.

We used the Euclidean distance between the current belief and the previous belief subset as a criterion to determine whether the root node should be reassigned or maintained, which is defined as follows:

$$\delta(b, B) = \min_{b_i \in B} \|b - b_i\|_2, \quad b \notin B \tag{3.10}$$

When $\delta(b_t, \tilde{B}_{t-1}) > d$, where d is a threshold value, the current belief b_t should be assigned to the root node to collect new belief set $\tilde{B}_t$, and vice versa. Fig. 3.5 also illustrates the distance in the three-state system. Various kinds of heuristic criteria are discussed in [106].

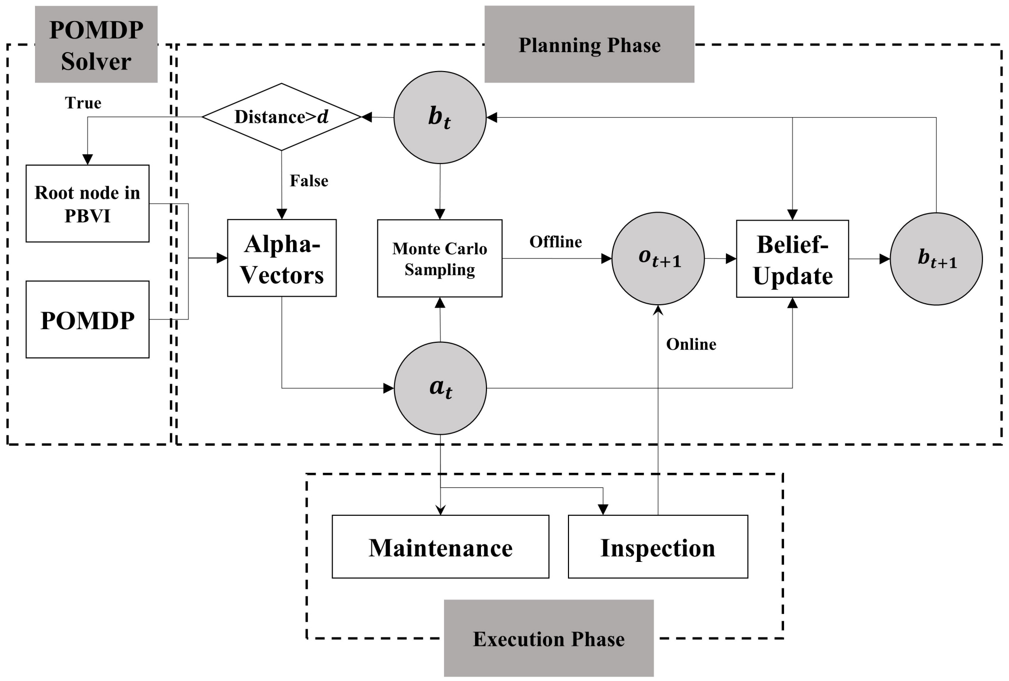


Figure 3.6: Receding horizon POMDP in both offline and online environments.

The proposed framework of receding horizon POMDP can be carried out in both offline and online environments and consists of three parts: POMDP solver, planning phase, and execution phase [107]. Descriptions on implementing the scheme is shown in Fig. 3.6.

With the current belief b_t, a time-variant infinite horizon policy $a_t = \pi_t(b_t)$ is obtained through the alphavectors evaluated from the POMDP solver. Then the root node in PBVI is updated to b_t. In the offline environment the real observation data o_{t+1} is generated by Monte Carlo sampling with the basis probability $p(o_{t+1}|b_t, a_t)$ defined in Eq. (2.6). In the online environment, the maintenance, inspection, and sensor heterogeneous scheduling a_t is implemented in the execution phase. In contrast to the offline phase, the observation data o_{t+1} can be obtained directly by inspection action. Finally, belief at the next decision epoch b_{t+1} is calculated with the Bayesian state estimator in Eq. (2.5). We assume that the decision epoch is periodic, for example, once in a year. Execution phase is performed at the end of every decision epoch, and planning phase can be performed at any point between decision epochs.

3.3 Problem formulation for infrastructure scheduling

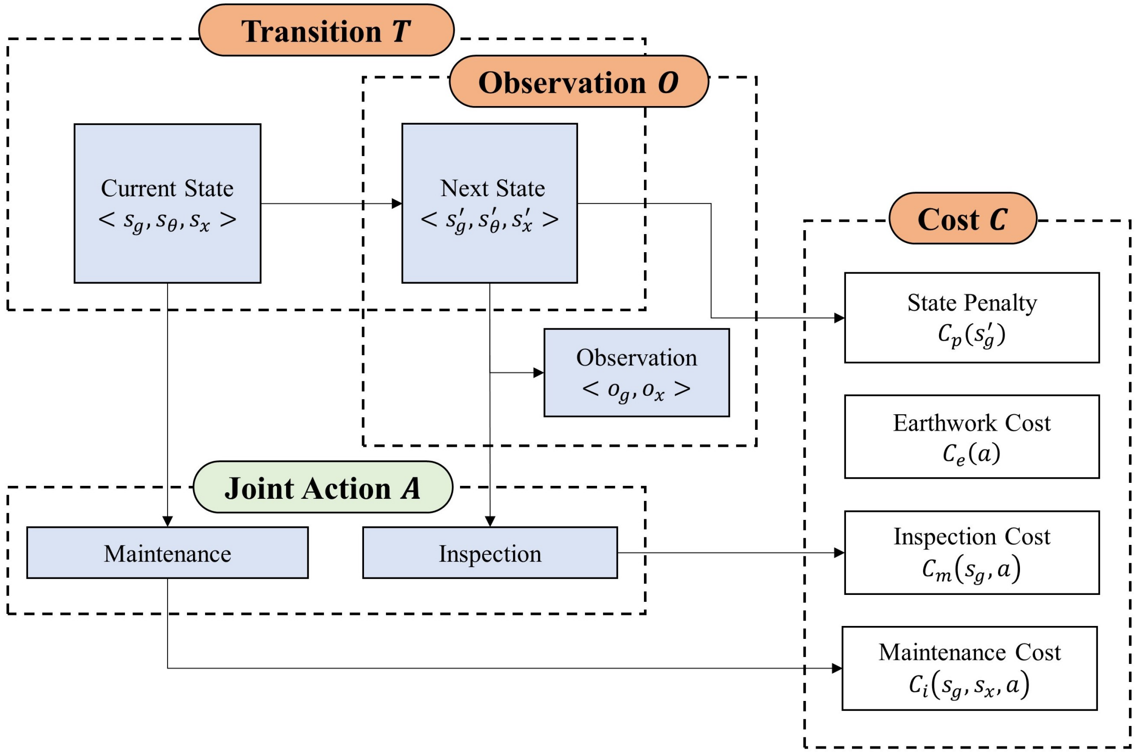

Transition T
Observation O
Cost C
Current State $< s_g, s_\theta, s_x >$
Next State $< s'_g, s'_\theta, s'_x >$
Observation $< o_g, o_x >$
Joint Action A
Maintenance
Inspection
State Penalty $C_p(s'_g)$
Earthwork Cost $C_e(a)$
Inspection Cost $C_m(s_g, a)$
Maintenance Cost $C_i(s_g, s_x, a)$

Figure 3.7: Infrastructure scheduling problem formulation

This section provides how each part of the infrastructure scheduling problem can be interpreted as a POMDP tuple, $< S, A, O, T, R, \Omega, \gamma >$. Fig. 3.7. shows a schematic diagram of the problem formulation.

3.3.1 State

In order to formulate the problem as Markovian, a pertinent set of features should be selected as the state. We specified the following three discrete integer variables as the state: grade (s_g), nominal age (s_θ) and the sensor-installation status (s_x).

The grade (s_g) is defined as the qualitative or quantitative measure of the functionality of a system. The condition rating system is generally used as a grade classification method and the details are discussed in [108, 95, 109]. For $s_g \in G = \{1, \ldots, |G|\}$, the first grade denotes a new system and state $|G|$ denotes the failure. We assume that grade is partially observable and only the belief distribution can be obtained. The belief distribution can be sharpened by inspection.

Second, the nominal age (s_θ) is an index to describe the deterioration rate of the system. Although different systems are in the same grade, their deterioration rates might be different. Without any action implemented on the system, i.e., a natural deterioration process, the real age and the deterioration rate index would be exactly the same. Once the action is implemented, however, the discrepancy between the real age and the deterioration rate index occurs. Thus a fictitious age index called nominal age $s_\theta \in \Theta = \{1, \ldots, |\Theta|\}$ describing the deterioration rate of a system is defined. The unit of nominal age is equal to that of the real age.

Finally, the sensor installation status (s_x) is defined for the scheduling of sensor-installation. s_x is a binary variable indicating whether a sensor is installed or not. It is assumed that a single sensor is installed per system, only one kind of sensor is available, and the sensor itself is invulnerable. The detailed explanation and motivation of sensor scheduling are presented in Sec. 3.3.2.

3.3.2 Maintenance and inspection actions

3.3.2.1 Maintenance actions

Joint action of maintenance, inspection, and sensor-installation is the action set for the infrastructure scheduling problem. Four maintenance actions are considered: No maintenance, patching, rehabilitation, and replacement. The maintenance actions affect the state $< s_g, s_\theta, s_x >$ with different mechanisms and are carried out in different ways.

First, no-maintenance lets the system experience the natural deterioration process. Both nominal age s_θ and real age increase by one time unit at each decision epoch.

Second, patching is a temporary method to affix part of the system. The grade s_g is improved even if a small part is patched. However, the patching action is insufficient to improve the deterioration rate as well. Hence, nominal age s_θ is assumed to be increased by one as if it were under the natural deterioration process.

Third, rehabilitation is an alternative maintenance action to improve the overall functionality and covers the whole system. As a consequence, both grade and deterioration rate are improved. The degree of improvement on the deterioration rate is modeled as reducing the nominal age s_θ by three time units [99].

Finally, system is replaced to a new one through the replacement. Grade and nominal age are initialized to one and the sensor installation status become zero ($< s_g, s_\theta, s_x >=< 1, 1, 0 >$).

3.3.2.2 Inspection actions and sensor installation

Three inspection actions are considered in the infrastructure scheduling: No-inspection, sensor-inspection, and excavation-inspection. In addition, sensor-installation is included in the set of inspection actions. Inspection actions affect the sensor installation status s_x and the observation probability O.

First, no-inspection does not reveal any information on the current state. The observation probability for each grade is uniformly distributed, except for the failure grade $s_g = |G|$.

Second, sensor-inspection is an indirect inspection method. Physical data from sensors such as pressure [30], magnetic flux [110], infrared thermography [111] and wave frequency [112] etc. are utilized and further analyzed for the fault detection, location, and grade estimation. Various kinds of nondestructive tests yield more accurate information and is preferable to the bare-eye inspections [113].

Third, additional sensors are installed through sensor-installation action. The sensor-inspection requires the sensor-installation in advance, and saves inspection cost in the long term. The effect of sensor-installation on the system can be describes by s_x taking the value of one.

Finally, excavation-inspection is a direct inspection method in which an excavation to reveal the buried or hidden system is accompanied. The excavation-inspection not only costs more than the sensor-inspection, but also its performance is lower than that of sensor inspection because no quantitative analysis using the accumulated data is performed. Excavation inspection is only performed when $s_x = 0$.

Table 3.1: Possible candidates of joint action for maintenance and inspection.

	No-inspection	Sensor-inspection	Sensor-installation	Excavation-inspection
No-maintenance	A	A	A	A
Patching	A	A	A	A
Rehabilitation	A	A	A	A
Replacement	A	N/A	N/A	N/A

Note. A = Available. N/A = Not available

All the combinations of maintenance and inspection actions are possible except that only the replacement action can be carried out simultaneously with no-inspection action. As a result, we specified the discrete integer action variable a whose cardinality is thirteen (i.e., $|A| = 13$). The possible joint actions are shown in Table 3.1.

3.3.3 State transition function

The state transition function $T(\cdot, a, \cdot)$ is obtained with the deterioration model and the maintenance model.

3.3.3.1 Deterioration model of water pipe

The pipe would deteriorate naturally if no improvement had been employed. Physical and statistical models are used for predicting the grade and deterioration rate of a system under natural deterioration process. Since the former requires a rigorous understanding of a particular infrastructure, the latter, represented by survival analysis, has been widely used [114]. In [97], survival function is represented as a two-parameter Weibull model which yields a good fitting performance [115]. Natural deterioration process is modeled as a semi-Markov process and series of time-variant deterioration matrices are derived.

Deterioration matrix can be evaluated by the deterioration model of water pipe. The basic idea of the deterioration model is to estimate a survival function or a hazard function for a water pipe. Estimating those functions is called survival analysis which has been widely studied. Models developed by Weibull are the most prominent, but they only considers the two state system ($|S| = 2$); failure or not.

[97] generalizes the deterioration model to n state variables, and provides the methods to evaluate the deterioration matrix.
Let $\{T_1, T_2, \ldots, T_{|S|-1}\}$ be random variables representing the waiting time in states $\{1, 2, \ldots, |S|-1\}$. For example, it takes T_i for the process to go from state i to $i+1$.
When we define the random variable $T_{i\to k}$ as the sum of waiting times in states $\{i, i+1, \ldots, k-1\}$, we can obtain the cumulative waiting time between states i and k. In general, summation of two or more random variables can be calculated analytically by convolution integral. Probability density function (PDF), survival function (SF) of $T_{i\to k}$ are denoted as $f_{i\to k}(T_{i\to k})$, $S_{i\to k}(T_{i\to k})$. Then the transition probability of state i to state $i+1$ is the generalization of hazard function which can be expressed as follows.

$$\begin{aligned} \Pr[s_{t+1} = i+1 | s_t = i] &= p_t(i+1|i, 1) \\ &= \frac{f_{1\to i}(t)}{S_{1\to i}(t) - S_{1\to i-1}(t)} \\ \text{for all} \quad & i = \{1, 2, \ldots, |S|-1\} \end{aligned} \tag{3.11}$$

Once the PDF and SF of waiting time $T_i(t)$ are established, every element of the deterioration matrix can be calculated.

The waiting time T_i of state i follows the Weibull probability distribution. Weibull model is the special case of the proportional hazards model whose physical interpretation is explained by [116]. Weibull model has two parameters and takes the following.

$$\begin{aligned} \text{SF} : S_i(t) = \Pr[T_i \geq t] = \exp[-(\lambda_i t)^{\beta_i}] \\ \text{PDF} : f_i(t) = \lambda_i \beta_i (\lambda_i t)^{\beta_i - 1} \exp[-(\lambda_i t)^{\beta_i}] \end{aligned} \tag{3.12}$$

Parameters λ_i and β_i can be calculated by regression using the survival history of water main system of target region (e.g. $x\%$ probability of being in state i more than t years).
The criteria of classifying the state of water main pipe has been suggested by many researchers [117, 118] and municipal government. The data on which the time a pipe takes to shift from one state to other without any action employed would be recorded. Decision maker uses those historical data to find the parameters of Weibull model and evaluate the deterioration matrix.

3.3.3.2 Grade transition function

As natural deterioration process goes on, the grade belief converges to $[e_{|G|}]_i = \delta(i, |G|), i \in G$ (i.e., $e_{|G|} = [0, 0, \ldots, 1]$). Since Weibull model indicates that grade and deterioration rate are non-decreasing, all the beliefs after a certain time have the same value:

$$\forall \epsilon \geq 0, \quad \exists M \in \mathbb{R} \quad s.t. \quad t \geq M, \quad \left\| b_t - e_{|G|} \right\| < \epsilon \tag{3.13}$$

According to Eq. (3.13), the horizon of model prediction is specified to be bigger than M. The size of nominal age set, $|\Theta|$, defined in Sec. 3.3.1 is set to be equal to M. As a result, a series of $|\Theta|$ deterioration matrices $T_d(s_\theta)$ are obtained:

$$[T_d(t)]_{ij} = p(s'_g = j | s_g = i, s_\theta = t) \quad i, j \in G, \quad t \in \Theta \tag{3.14}$$

The elements of $T_d(t)$ can be further reduced by two assumptions. First, it is assumed that $[T_d(t)]_{ij} = 0$ when $i < j$, meaning that improvement does not take place in the natural deterioration process. Second, $[T_d(t)]_{ij} = 0$ when $i + 2 \leq j$, meaning that system deteriorates only by one grade at each decision epoch. When $|G| = 5$, deterioration matrix is expressed as follows:

$$T_d(t) = \begin{bmatrix} p(1|1,t) & p(2|1,t) & 0 & 0 & 0 \\ 0 & p(2|2,t) & p(3|2,t) & 0 & 0 \\ 0 & 0 & p(3|3,t) & p(4|3,t) & 0 \\ 0 & 0 & 0 & p(4|4,t) & p(5|4,t) \\ 0 & 0 & 0 & 0 & p(5|5,t) \end{bmatrix} \tag{3.15}$$

Maintenance actions improve the state, whereas inspection actions do not. The effect is also stochastic due to the uncertainty on materials, equipment, technical skill of individual workers, and ambient condition [95]. Hence, the maintenance model is also expressed as the probability matrix. Maintenance matrix $T_m(a)$ describes the state transition when action a is taken.

$$[T_m(a)]_{ij} = p(s'_g = j|s_g = i, a) \quad i, j \in G \tag{3.16}$$

Maintenance actions are implemented at the end of decision epoch, and the transition probability with nominal age t and maintenance action a is calculated by Chapman-Kolmogorov equation:

$$\begin{aligned} & p(s'_g = j|s_g = i, s_\theta = t, a) \\ = & \sum_{k\in G} p(s''_g = j|s'_g = k, a)p(s'_g = k|s_g = i, s_\theta = t) \\ = & \sum_{k\in G} [T_m(a)]_{kj}[T_d(t)]_{ik} = [T_d(t)T_m(a)]_{ij} \end{aligned} \tag{3.17}$$

The grade transition function T_g is then simply obtained with multiplying deterioration matrix by maintenance matrix:

$$T_g(s_\theta, a) = T_d(s_\theta)T_m(a) \tag{3.18}$$

3.3.3.3 Transition functions for the nominal age and sensor installation status

Transition functions for the nominal age s_θ and sensor installation status s_x are deterministic. $T_{\theta,x}$ denotes the joint transition function for the augmented state tuple $< s_\theta, s_x >$. Despite its deterministic nature, the transition function is expressed as a probability matrix for compatibility with other functions. In Eq. (3.19), $f_a(\cdot)$ denotes the deterministic transition for $< s_\theta, s_x >$, $\delta(\cdot)$ is the Kronecker delta function.

$$\begin{aligned} [T_{\theta,x}(a)]_{ij} &= p(< s'_\theta, s'_x >= j| < s_\theta, s_x >= i, a) = \delta(f_a(i), j) \\ i, j &\in \{0, 1\} \times \Theta \end{aligned} \tag{3.19}$$

$f_a(\cdot)$ for each action is defined in Table 3.2, and the detailed explanation is given in Sec. 3.3.2.

Table 3.2: $f_a(\cdot)$: Deterministic transition function for $< s_\theta, s_x >$

	No-inspection	Sensor-inspection	Sensor-installation	Excavation-inspection
No-maintenance	$s'_\theta = s_\theta + 1$ $s'_x = s_x$	$s'_\theta = s_\theta + 1$ $s'_x = s_x$	$s'_\theta = s_\theta + 1$ $s'_x = s_x$	$s'_\theta = s_\theta + 1$ $s'_x = s_x$
Patching	$s'_\theta = s_\theta + 1$ $s'_x = s_x$	$s'_\theta = s_\theta + 1$ $s'_x = s_x$	$s'_\theta = s_\theta + 1$ $s'_x = 1$	$s'_\theta = s_\theta + 1$ $s'_x = s_x$
Rehabilitation	$s'_\theta = s_\theta - 3$ $s'_x = s_x$	$s'_\theta = s_\theta - 3$ $s'_x = s_x$	$s'_\theta = s_\theta - 3$ $s'_x = 1$	$s'_\theta = s_\theta - 3$ $s'_x = s_x$
Replacement	$s'_\theta = 1$ $s'_x = 0$	N/A	N/A	N/A

3.3.4 Cost function

From now on, we use the cost function, C, instead of the reward function R for the infrastructure scheduling problem. Cost function includes the cost of action implementation and the state penalty. Action implementation cost is problem-specific. In the infrastructure system, the costs for maintenance and inspection generally consist of earthwork cost, laid-down cost and accessorial cost. Specifically, each cost consists of material cost, labor cost and public expenditure [119].

Earthwork cost $C_e(a)$ occurs when excavation takes place, and is independent of the state. Maintenance cost C_m and inspection cost C_i are defined as the summation of laid-down cost and accessorial cost except for earthwork cost C_e, respectively. Maintenance cost $C_m(s_g, a)$ depends on grade s_g and inspection cost $C_i(s_g, s_x, a)$ depends on both grade s_g and sensor installation status s_x. The dependency on s_x represents the constraints, that the sensor-inspection is impossible when $s_x = 0$, while sensor-installation is impossible when $s_x = 1$. Costs for infeasible cases are set to be infinite so that they could be automatically excluded from the value iteration.

State penalty is the cost incurred immediately when a system occupies a specific state. Agent can give a preference for a certain state by granting different penalties to the states. In extreme cases, if an infinite penalty is given to a particular state, it becomes a forbidden state to which transition probability from any other state is zero. The penalty cost in an infrastructure problem can be understood as a degree of impact caused by structural deterioration. The worse the grade, the worse the structural deterioration and functionality, and the higher the state penalty cost. Specifically, the penalty cost of grade $|G|$ is equal to the failure cost. Since the POMDP solution is sensitive to penalty cost, careful tuning is required.

The overall cost function C is defined in Eq. (3.20), and it is worth noting that the cost function is independent of nominal age s_θ.

$$\begin{aligned} & C(< s_g, s_\theta, s_x >, a, < s_g', s_\theta', s_x' >) \\ & = C_e(a) + C_m(s_g, a) + C_i(s_g, s_x, a) + \gamma C_{sp}(s_g') \end{aligned} \tag{3.20}$$

3.3.5 Observation set and observation function

Among the state tuple $< s_g, s_\theta, s_x >$, s_g and s_x are revealed to the agent and included in the observation set. As mentioned in Sec. 3.3.1, s_g is partially observable except for the failure grade whose observation is always deterministic. s_x is fully observable because it is always known regardless of the inspection method if the exact history of sensor installation is available. Meanwhile, s_θ is unobservable to any inspection method because deterioration rate can only be estimated indirectly with two adjacent grades. In summary, the observation set O is represented by a tuple of grade observation o_g and sensor installation status observation o_x. (i.e. $O =< o_g, o_x >$)

The observation function depends on the inspection action. The observation function of grade $O_g(a)$ is stochastic, whereas that of the sensor installation $O_x(a)$ is deterministic. As with $T_{\theta,x}$, $O_x(a)$ is expressed as a probability matrix for compatibility with other functions:

$$
\begin{aligned}
&[O_g(a)]_{ij} = p(o_g = j|s'_g = i, a) \quad i, j \in G \\
&[O_x(a)]_{ij} = p(o_x = j|s'_x = i, a) = \delta(i, j) \quad i, j \in \{1, 2\}
\end{aligned}
\tag{3.21}
$$

3.3.6 State augmentation

T, C, and O are time-variant and history-dependent. State augmentation can make the functions time-invariant and thus compatible with the infinite horizon formulation. The augmentation encodes the entire dynamics within the prediction horizon Θ into a single matrix. The augmentation yields a state tuple $S =< s_g, s_\theta, s_x >$ and the observation tuple $O =< o_g, o_x >$. According to the computational complexity described in Sec. 3.2.2, considering the sensor installation with the binary variable s_x increases the complexity more than four times.

3.4 Illustrative example and simulation result

A scheduling problem of a water distribution pipe is analyzed as an illustrative example. Water distribution system is one of the core infrastructure facilities, and the systematic scheduling is essential. Most of the water pipes are buried underground, which hinders the state estimation. Therefore, this system is chosen as the representative system that follows the features and difficulties of the infrastructure scheduling described in Sec. 3.1. In particular, a stainless water pipe buried under the ground whose diameter is 20mm, one of the standard size in water main system, is considered. There are five grades and Weibull parameters can be estimated through survival analysis. The parameters avaiable in [97] are used to obtain $T_d(t)$ and $T_m(a)$ are evaluated by referring [99]. The probabilities of observing the right state, $O_g(a)$, of No-inspection, Sensor-inspection, Excavation-inspection are 0.4, 0.63, and 0.92, respectively. Cost functions are adapted from the breakdown cost table of water main construction [119] and summarized in Table 3.3. The sizes of augmented state, action, and augmented observation set are 1000, 13, and 10, respectively, and the model prediction horizon is one hundred years. The POMDP solution algorithm gapMin was implemented in MATLAB R2016a.

Table 3.3: Cost table for the water distribution pipe

		Earthwork	Laid-down	Accessorial	Total
Maintenance	No-maintenance	-	-	-	0
	Patching	22.0	15.6	3.7	41.3
	Rehabilitation	45.0	126.1	47.7	218.8
	Replacement	64.3	265.9	20.0	350.2
Inspection	No-inspection	-	-	-	0
	Sensor-inspection	-	-	0.2	0.2
	Sensor-installation	7.0	3.5	3.5	14.0
	Excavation-inspection	22.0	-	6.1	28.1
Penalty		-	-	-	293.1

Note. Unit of the costs is omitted since the scheduling result is only affected by the relative values.

3.4.1 Structural point for the analysis of a high dimensional belief space

The belief domain is a reachable belief set from the initial belief b_0 and it is a sparse subspace of the convex hull of state space. The sparsity is caused by representing the deterministic transition and observation function for s_θ and s_x. Only partially observable state s_g can make non-zero elements in the belief. In order to analyze the high-dimensional sparse belief space, we first take the non-zero elements of the belief vector and project it onto a scalar. The value of non-zero elements of belief depends only on s_g, whereas s_θ and s_x only affect the index of non-zero elements. Thus, the projection is defined as a weighted average of s_g, with the state penalty cost $C_{sp}(s_g)$ being the weights. Note that the projection has a similarity with the conditional rating system explained in Sec. 3.3.1. Belief b is projected onto a single point in the one dimensional space which is called structural point $SP(b)$:

$$SP(b) = \sum_{s \in S} C_{sp}(s) b(s) \tag{3.22}$$

3.4.2 Infinite horizon policy under the natural deterioration process

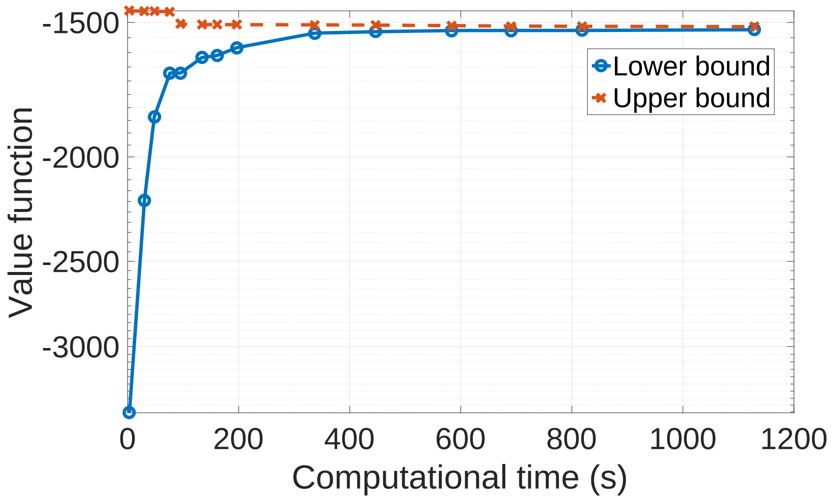
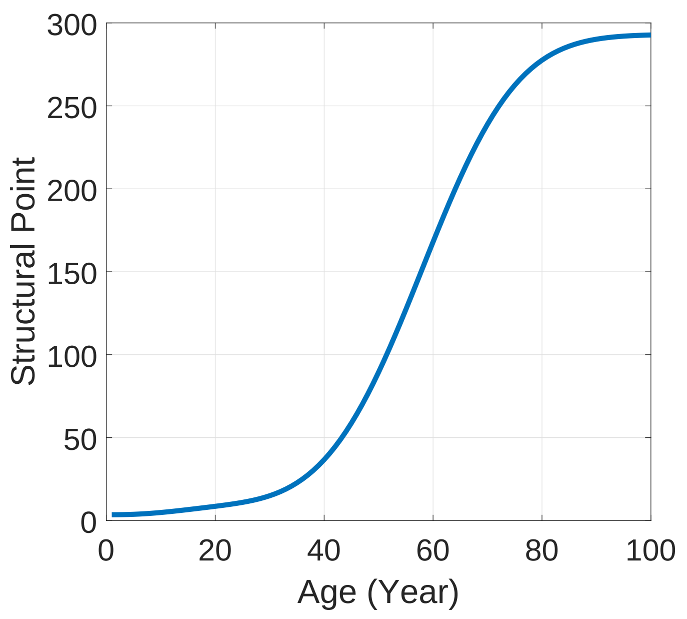


Figure 3.8: Convergence of POMDP problem

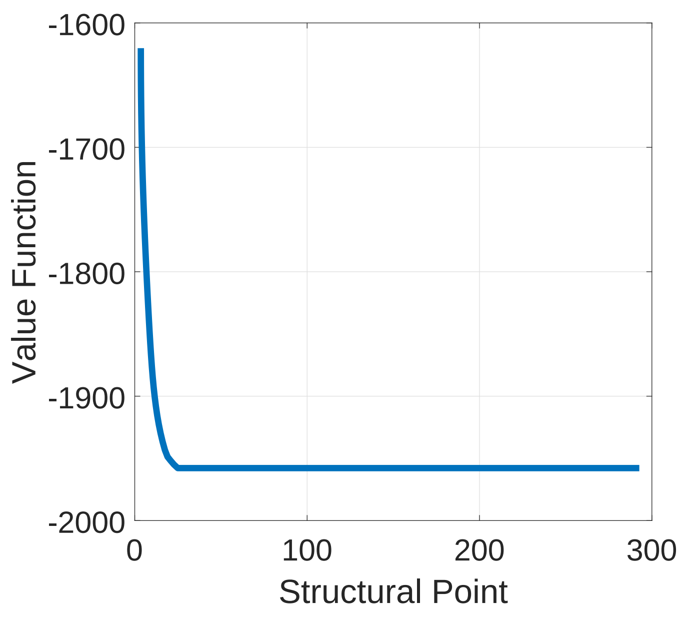
To observe the effect of deterioration on the optimal policy, infinite horizon POMDP problems by varying the age of water pipe are obtained. This problem is the same as the new system goes through the natural deterioration process during the model prediction horizon (i.e., one hundred years). The solution to the infinite horizon POMDP problem is represented as an alphavector set. Fig. 3.8 illustrates the lower bound and upper bound of the value function approximation converged by PBVI algorithm, and Table 3.4 shows the detailed result of PBVI algorithm. The convergence criterion is that the gap is one hundred times smaller than the absolute value of the upper bound approximation. The cardinality of the alphavector set $|\Gamma|$ and the belief-value pair set $|\Upsilon|$ are 191 and 435, respectively. The algorithm was terminated after 1129 seconds.

Table 3.4: Results of POMDP problem by PBVI algorithm

Gap	LB	UB	$\lvert\Gamma\rvert$	$\lvert\Upsilon\rvert$	Time (s)
10.082	-1523.7	-1513.7	191	435	1129



(a)



(b)

Figure 3.9: (a) $SP(b)$ variation over system age. (b) Value function as function of $SP(b)$

Beginning with an age-one-pipe, beliefs under the natural deterioration process for one hundred years are obtained. To observe only the effect of deterioration on the $SP(b)$, we assume that the observation probability is one. Each belief is projected to $SP(b)$, and the lower bound approximation of value functions are calculated by Eqs. (3.2) and (3.1). Fig. 3.9(a) shows the variations in $SP(b)$ over the age of system. Structural points are monotonically increasing since the pipe naturally deteriorates. Fig. 3.9(b) shows the lower bound approximation of value function over $SP(b)$. A piecewise linear, convex function $\underline{V}(b)$ retains its shape after the projection.

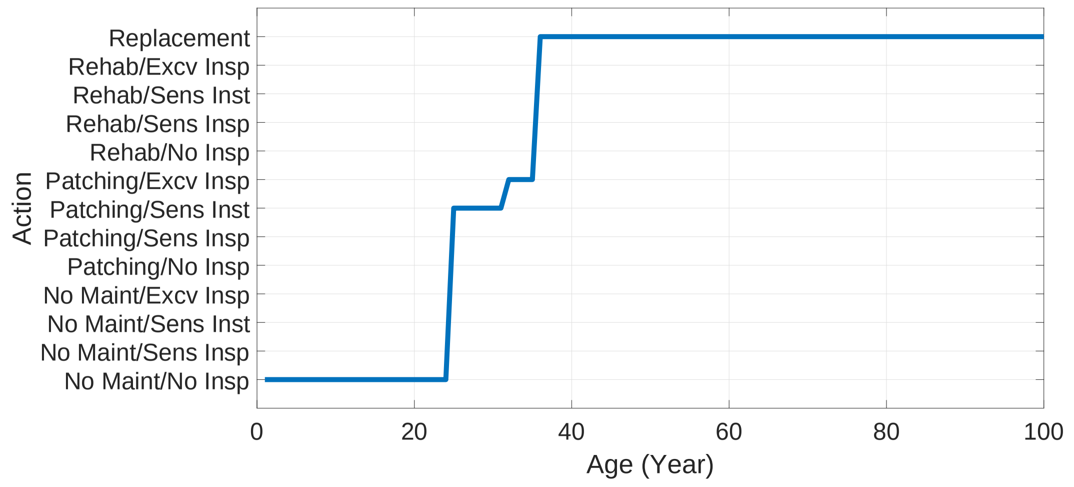


Figure 3.10: Infinite horizon policy result for each pipe age

Fig. 3.10 shows the optimal action for the system at each age. It is shown that patching/sensor-installation action is optimal after age 25, patching/excavation-inspection is optimal after age 32, and replacement is optimal after age 36. As age increases, more rigorous maintenance and inspection actions are required. In the area where the sensor-installation is optimal, the economic advantage caused by sensor-inspection in the future is greater than the loss caused by installing the sensor at the current decision epoch. This tendency is reversed in older ages.

3.4.3 Receding horizon POMDP

Fig. 3.11 shows the result of the finite horizon optimal policy for control horizon of 120 years following the steps of receding horizon POMDP explained in Fig. 3.6. The initial belief is assumed to be the age one pipe with no sensor installed. The most prominent point of the result is that the replacement action of optimal policy is pseudo-periodic, since the replacement action initializes s_g, s_θ, s_x to 1, and the belief at the next decision epoch is identical with the initial belief b_0, i.e., $b_{t+1} = b_0, a_t =$ Replacement. However, in a simulation environment, the behavior is sampled from $p(o|b, a)$, and hence the behavior is different every cycle. Under the pseudo-periodic replacement, a complex combination of maintenance and inspection actions occurs.

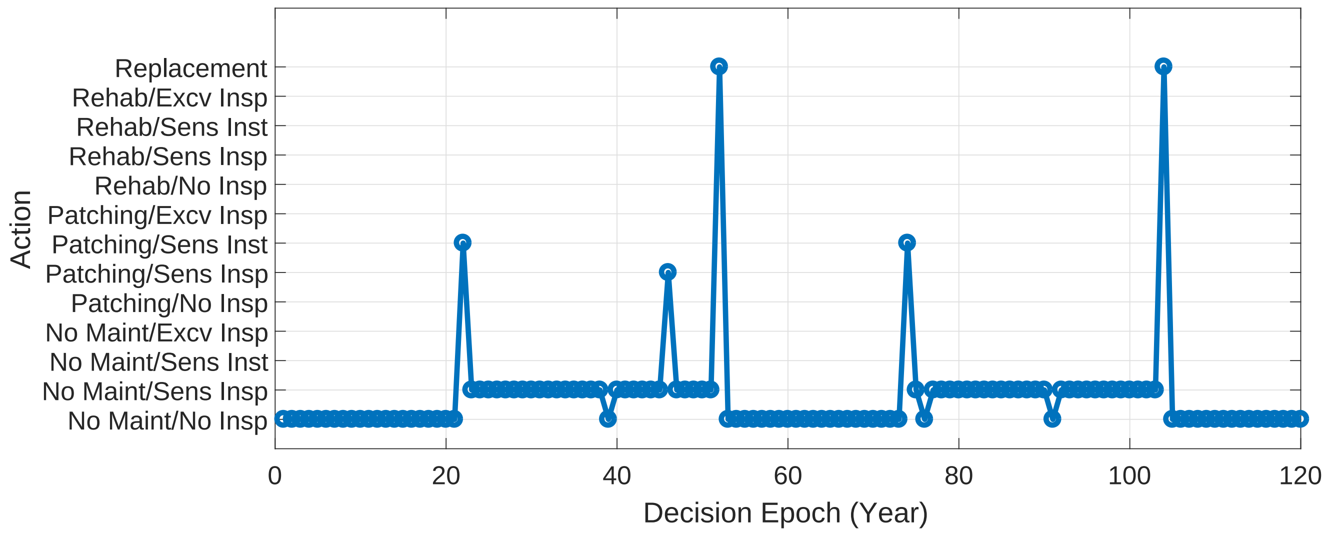
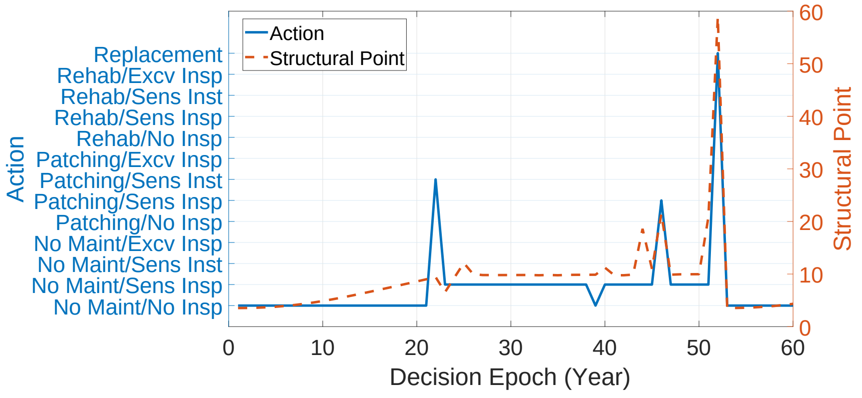
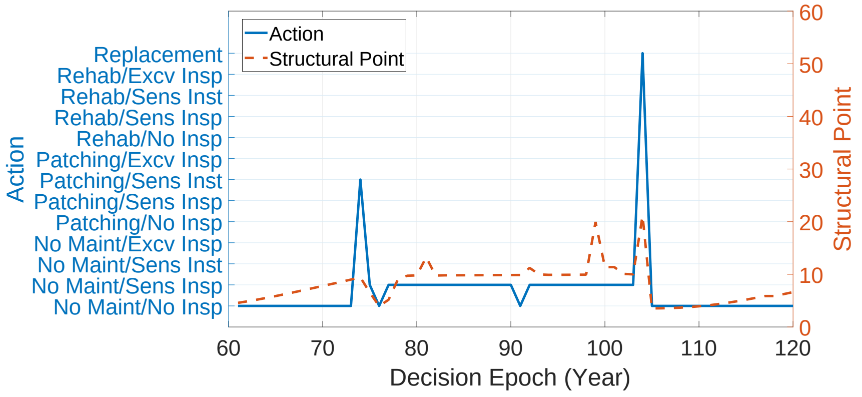


Figure 3.11: Receding horizon policy computed in offline environment

Fig. 3.12 shows the variation of structural points $SP(b)$ under the optimal policy until the first and second replacements are encountered. Until age 21 in the first period and 73 in the second period, no action is implemented and the structural points are increasing. Patching/sensor-installation action is implemented on the pipe at age 21 and 73, respectively. After sensor is installed after 22 years from replacement actions, sensor-inspection becomes available to the system. Thus, sensor-inspections such as patching/sensor-inspection or no-maintenance/sensor-inspection are implemented to obtain the state information. It is shown that the reliability is underestimated in situations where no inspection method is implemented in that the $SP(b)$ value is increased by the inspection methods. Thus, the 'fully observable' assumption of MDP provides more optimistic policy than that of POMDP, resulting in a higher cost.



(a)



(b)

Figure 3.12: Optimal policy of a single period compared with the structural points. (a) First period and (b) second period

3.4.4 Validation of POMDP policy via Monte Carlo simulation

The scheduled policy is then compared with other policies including POMDP policy without feedback rule, MDP policy, heuristic policy and myopic policy. In the case of POMDP policy without feedback, the root node is fixed with the initial belief b_0 throughout the planning horizon. MDP policy is calculated by value iteration in Eq. (2.2). In MDP framework, only maintenance actions can be considered. With the assumption of full observability, the state is not expressed as a probability distribution, and the result can be expressed as a finite size state-action mapping table shown in Fig. 3.13. The augmented state tuple in MDP is $< s_g, s_\theta >$, each of which is displayed on the x-axis and y-axis of the plot. Similar to the result of POMDP policy in Fig. 3.11, more rigorous maintenance action is required as pipe becomes less reliable.

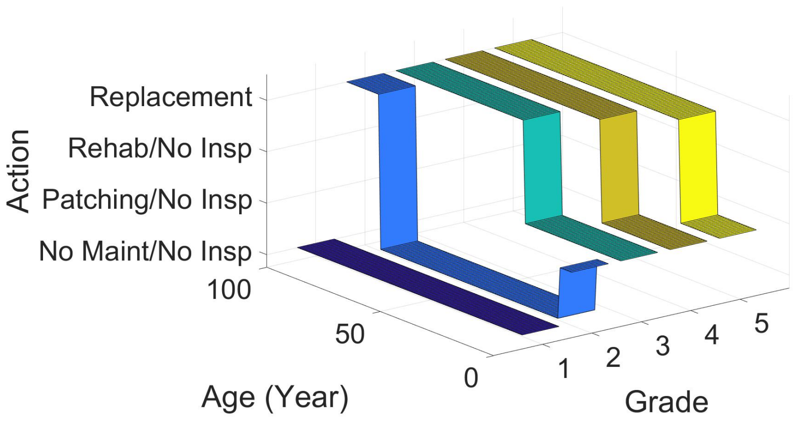


Figure 3.13: MDP policy

Heuristic policy does not utilize any optimization methodology, but specifies replacement or inspection periods a priori. Here, we intentionally set the periods to be close to those of POMDP policy so that more meaningful comparison can be made. As a result, replacement and sensor inspection periods are 40 years and 3 years, respectively, and sensor is installed after 5 years of replacement. Myopic policy is defined as a policy that replaces the infrastructure only after a failure occurs.

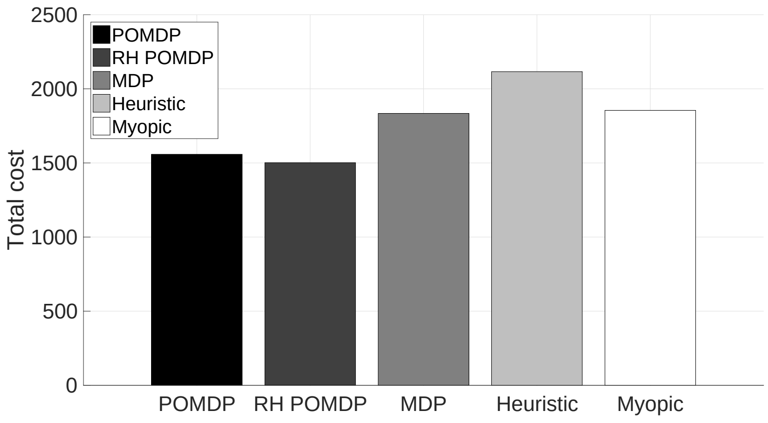


Figure 3.14: Total cost comparison

The discounted total cost within 120 years is estimated by generating the random scenario using Monte Carlo sampling. $T(s, a, s')$ and $p(o|b, a)$ are used to sample the successor state s' and observation o, respectively. An optimal policy is determined with respect to the belief b for POMDP and observation o for MDP. Each experiment was repeated 5000 times to reduce the effect of randomness. Fig. 3.14 shows that the receding horizon POMDP policy yields the minimum cost compared to other policies. The feedback rule for the root node in receding horizon scheme improves the previous POMDP algorithms. Under the observation uncertainty, even MDP is a worse choice than heuristic and myopic policies, which indicates that 'fully observable' assumption is in fact unrealistic.

Chapter 4

A model-based deep reinforcement learning method applied to finite-horizon optimal control of nonlinear control-affine system [2]

4.1 Introduction

In the optimal control theory, the Hamilton-Jacobi-Bellman (HJB) equation plays a key role, not only in verifying the optimality of a given policy, but also in constructing one. The HJB equation is a partial differential equation (PDE) and its solution is referred to as the value function, which measures the 'cost-to-go' of a starting state [13]. However, since it seldom can be solved exactly for nonlinear systems, various approaches to build or implement approximate solutions have been tried. When the system dynamics is control-affine and the cost function quadratic, the HJB equation can be transformed and decomposed into simpler equations with respect to the value function, its first-order derivative (called costate function), and the policy function [62]. Globalized dual heuristic programming (GDHP), a model-based RL method is based on these surrogate equations, uti-

[2]This chapter is an adapted version of J. W. Kim, B. J. Park, H. Yoo, J. H. Lee, and J. M. Lee, "A model-based deep reinforcement learning method applied to finite-horizon optimal control of nonlinear control-affine system," *Journal of Process Control*. Under review. [120]

lizing all such information about the system dynamics and the cost function [61, 34].

Finite horizon optimal tracking control (FHOC) is the main problem of interest in this study. RL approaches for FHOC have widely been studied [121, 122, 123, 124, 125]. The FHOC problem has a certain feature that distinguishes it from the infinite horizon problem. The solution involves time-varying value, costate, and policy functions with terminal boundary conditions. In the case that time-varying trajectories must be tracked, the usual approach has been to reformulate the tracking problem as a regulation problem by introducing the offset dynamics [122, 123, 126]. However, this method requires the steady-state values of the state and the control input to be known for each target reference value in order to achieve offset-free setpoint tracking. Instead, this study adopts state space augmentation including the integral action and the time variable in order to express the time dependency and to be free from obtaining such prior information [26].

One of the key elements of these learning-based methods is a function approximator, which is used to represent the value, policy, or other related function in a parametric form. A popular choice has been the neural networks given its property as a universal approximator: A neural network with a single hidden layer can represent any bounded continuous function defined in any compact subset of the real space [127]. Hence, thus far, most studies have employed a single-layer neural networks (SNNs) [62, 128, 129, 130]. The learning method for SNNs can be formulated as a least-squares problem, by taking advantage of the linearity of the approximator. In the case of FHOC, in order to construct a time-varying approximator, Cheng

[131] and Adhyaru [121] used time-varying weights, whereas Heydari [129], Zhao [124], and Mu [125] employed time-varying activation functions. Both approaches showed good results for obtaining feedback policies of the FHOC problem in small scale problems for which the functions to represent are relatively simple.

The latest trend is to explore the potential benefits of deep neural networks (DNNs) which have multiple hidden layers [132]. Though SNNs is a universal approximator, it does not scale very well, showing exponential complexity growth for certain types of functions. Having multiple hidden layers has been shown to result in better scalability, meaning there is a right choice of depth for each function from the viewpoint of statistical efficiency. Use of DNNs has allowed model-free RL methods to be applied successfully to problems with high-dimensional continuous state and action spaces [43], and the state-of-the-art deep RL (DRL) has shown some remarkable performances in certain applications such as robotics, games, spacecrafts, etc. [37, 39, 133]. Motivated by the recent successes, this study adopts DNNs as function approximators. In the context of the proposed model-based RL method, DNNs will be shown give more robust approximations in the presence of varying initial state and state noise, compared to SNNs.

Whether shallow or deep, neural networks always present the problem of potential overfitting, which can lead to undesirable consequences using a learned policy, because the optimization step involved in RL is based on the function approximators. The DNNs framework even exacerbate the overfitting issue. Particularly in process control, where the envelope of the data distribution is small compared to the dimension of state space due to the correlations and state dynamics, overfitting can easily result [134]. Moreover, the amount of data is small due to the cost and time limitation and the safety constraints. Hence, rather than the model-free and data-oriented algorithm, available model knowledge on the system dynamics and cost should be fully used to solve the optimal process control problem. Apart from the model-based setting, we adopt various machine learning techniques which decorrelate the data and increase the stability in learning to alleviate the overfitting issue [43, 135].

4.2 Function approximation and learning with deep neural networks

4.2.1 GDHP with a function approximator

Value, costate, and policy functions are typically highly complex nonlinear functions defined in the state space Ω. One of the main difficulties in RL is the type selection and structure design of the function approximator [136]. This difficult stems from several factors. First, the optimal value, costate, and policy functions are defined globally in Ω, not for a local region generated by a single trajectory. Thus the approximated functions should also be defined over the entire state space. Second, nonlinearities of the functions depend on the system dynamics and the cost functions, and thus can be arbitrarily complex. Finally, the complexity may grow even larger in the stochastic cases where the optimality equations Eqs. (2.18) - (2.22) contain the expectation operator.

As a universal function approximator, deep neural networks (DNNs) has received much attention [137]. Reinforcement learning with DNNs, often called deep RL (DRL), is solving the RL problems in a very high-dimensional state space [43, 135]. The recent success of DNNs can be attributed to the following: First, DNNs are known to provide highly scalable structure with respect to the dimension of the function input, as it can perform automatic feature selections [138]. Second, the development of automatic differentiation (AD), pretraining for good initialization, and GPU-based computing have significantly lowered the barrier of training DNNs.

The robust interpolation capabilities of DNNs can compensate for the weakness of the GDHP algorithm in estimating the HJB equation, which is basically a sample-based and bootstrapping method. Specifically, the right-hand-sides of the GDHP update rules, i.e., Eqs. (2.48) - (2.50), contain the recursive function evaluations for each noisy argument x_{k+1}. Hence, the approximators should represent the value, costate, and policy functions by interpolating between such noisy, transient data. Since the overfitting problem is inevitable with any complex function approximator with a large number of fitting parameters, the performance of the learned control policy can degrade significantly when the uncertainties encountered on-line are significantly different from those seen during the offline training. Discussions of the methods to control the trade-off between generalizability and being prone to overfitting are presented in the following section. Section 4.3 provides the result of DNNs' approximations, when obtained using the state-of-the-art DRL stable learning methods, significantly outperform the SNNs approximation used in the previous studies of [139], in terms of robustness to stochastic uncertainties.

In the GDHP algorithm, the measurement tuple $(x_k, \Delta u_k, r_k, x_{k+1})$, which is composed of the current augmented state, control, cost, and the successor augmented state, is obtained at every time step. The elements of tuple for the terminal time are just the terminal augmented state and terminal cost as (x_T, ϕ). The measurement tuple set are expressed as

$$\mathcal{D} = \{S_n | S_n = (x_{k_n}, \Delta u_{k_n}, r_{k_n}, x_{k_n+1}), n \in \mathbb{N}\} \tag{4.1}$$

$$\mathcal{D}_T = \{S_n | S_n = (x_{T_n}, \phi_n), n \in \mathbb{N}\} \tag{4.2}$$

where S_n is the n^{th} measurement tuple of the set. The augmented states within the set $\mathcal{D}$ can be alternatively expressed as a compact matrix form, $X_{\mathcal{D}}, X_{\mathcal{D}}^+ \in \mathbb{R}^{(S+A+1)\times N}$ such that $X_{\mathcal{D}} = [x_{k_1}, x_{k_2}, \ldots, x_{k_N}]$ and $X_{\mathcal{D}}^+ = [x_{k_1+1}, x_{k_2+1}, \ldots, x_{k_N+1}]$, respectively. Denote the function index set of value, costate, and policy functions as $\psi \in \{V, \lambda, \pi\}$. DNNs approximations of value, costate, and policy functions are written as

$$\hat{\psi}(X_{\mathcal{D}}) = \hat{W}_{\psi,H+1}\sigma_{\psi,H}(\hat{W}_{\psi,H}\sigma_{\psi,H-1}(\cdots \hat{W}_{\psi,2}\sigma_{\psi,1}(\hat{W}_{\psi,1}X_{\mathcal{D}})\cdots)) \tag{4.3}$$

We assume that the three networks have same number of layers, i.e., H, and same widths at each layer. The weight matrices and the nonlinear activation functions of the l^{th} layers of the function $\hat{\psi}$ are denoted as $\hat{W}_{\psi,l}$ and $\sigma_{\psi,l}(\cdot)$, respectively.

In accordance with the recursion in Eqs. (2.48) - (2.50), the 'targets' that the current estimates of the value, costate, and policy functions (i.e., $\hat{\psi}(x_k)$) should satisfy regarding the optimality equations, denoted as $\tau_{V,k}$, $\tau_{\lambda,k}$, and $\tau_{\pi,k}$, respectively, are evaluated each time when a single sample $(x_k, \Delta u_k, r_k, x_{k+1})$ becomes available [140]:

$$\tau_{V,k} = r_k + \hat{V}(x_{k+1}) \tag{4.4}$$

$$\begin{aligned}\tau_{\lambda,k} &= 2\left(\frac{\partial h(x_k)}{\partial x_k}\right)^T Q\left(h(x_k) - \rho_k\right) \\ &\quad + \left(\frac{\partial x_{k+1}}{\partial x_k}\right)^T \hat{\lambda}(x_{k+1})\end{aligned} \tag{4.5}$$

$$\tau_{\pi,k} = -\frac{1}{2}R^{-1}g^T(x_k)\hat{\lambda}(x_{k+1}) \tag{4.6}$$

In addition to the optimality targets, we used the residuals which measures the deviation of the definition of costate function in Eq. (2.19),

$$e_{V\lambda,k} = \frac{\partial \hat{V}(x_k)}{\partial x_k} - \hat{\lambda}(x_k) \tag{4.7}$$

This equation forces the value and the costate functions to satisfy the definition explicitly. Note that the previous GDHP algorithms make the relationship between the value and the costate functions implicitly by sharing the first layer of their neural networks [61, 140], which turns out to be impractical in the DNNs case. The targets associated with the boundary conditions of the value and costate functions $\tau_{V,T}$ and $\tau_{\lambda,T}$ are also defined as

$$\tau_{V,T} = \phi(x_T) \tag{4.8}$$

$$\tau_{\lambda,T} = \frac{\partial \phi(x_T)}{\partial x_T} \tag{4.9}$$

When the algorithm proceeds online, the data collection and training can proceed concurrently. The DNNs weights $\hat{W}_{\psi,l}$ become incrementally and adaptively optimized to minimize the residuals by gradient based optimization as the measurements are obtained. Denote the vectors τ_V, τ_λ, τ_π and $e_{V\lambda}$ as the stacked vectors of Eqs (4.4) - (4.7) for the set $\mathcal{D}$, i.e., $\tau_V = vec\big([\tau_{V,k_1}, \tau_{V,k_2}, \dots, \tau_{V,k_{|\mathcal{D}|}}]\big)$. Similarly, the vectors $\tau_{V,T}$, and $\tau_{\lambda,T}$ are the stacked vectors of Eqs. (4.8) and (4.9) for the set $\mathcal{D}_T$.

Then we can define the loss functions with the square of the Euclidean norms as,

$$\begin{aligned}
&L_V(\mathcal{D} \cup \mathcal{D}_T) = \\
&\|\hat{V}(X_{\mathcal{D}}) - \tau_V\|_2^2 + \beta_{V,1}\|e_{V\lambda}\|_2^2 + \beta_{V,2}\|\hat{V}(X_{\mathcal{D}_T}) - \tau_{V,T}\|_2^2 \qquad (4.10)\\
&L_\lambda(\mathcal{D} \cup \mathcal{D}_T) = \\
&\|\hat{\lambda}(X_{\mathcal{D}}) - \tau_\lambda\|_2^2 + \beta_{\lambda,1}\|e_{V\lambda}\|_2^2 + \beta_{\lambda,2}\|\hat{\lambda}(X_{\mathcal{D}_T}) - \tau_{\lambda,T}\|_2^2 \qquad (4.11)\\
&L_\pi(\mathcal{D}) = \|\hat{\pi}(X_{\mathcal{D}}) - \tau_\pi\|_2^2 \qquad (4.12)
\end{aligned}$$

where $\beta_{V,1}$, $\beta_{V,2}$, $\beta_{\lambda,1}$, and $\beta_{\lambda,2}$ are the scalar weighting factors. The weights of the DNNs $\hat{W}_{\psi,l}$ are updated by the gradient descent rule as

$$\begin{aligned}
vec(\hat{W}^+_{\psi,l}) &= vec(\hat{W}_{\psi,l}) - \eta_\psi \frac{\partial L_\psi}{\partial vec(\hat{W}_{\psi,l})}\\
&l \in \{1, \dots, H+1\}, \quad \psi \in \{V, \lambda, \pi\} \qquad (4.13)
\end{aligned}$$

where $\hat{W}_{\psi,l}$ and $\hat{W}^+_{\psi,l}$ denote the current and updated l^{th} layer of the DNNs weight estimates of function ψ, respectively, η_ψ denotes the learning rate. The gradient of the networks with respect to the DNNs weights is calculated by the backpropagation rule.

4.2.2 Stable learning of DNNs

Employing the gradient descent based optimization in its bare form can lead to problems since DNNs with a large number of weight parameters are vulnerable to overfitting and the intrinsically unstable behavior of bootstrap methods. The state-of-the-art deep RL algorithms have tried to alleviate these issues with some tricks like replay buffer, target networks, etc.[43, 135, 141]. In addition to those tricks, we propose pre-training with some known policies and discretization to prevent the overfitting and to accelerate the training speed.

4.2.2.1 Replay buffer

DNNs have a significantly larger number of weights compared to other function approximators, which can cause an overfitting problem. Various regularization techniques such as minibatch training, weight regularization, and drop-out method are utilized to avoid the overfitting problem and reduce the generalization error [137]. Especially, in minibatch training, a batch dataset is divided into different subsets (i.e., $\mathcal{D} = \bigcup_{i=1}^{m} \mathcal{D}_i$) and the gradient descent method is applied to each $\mathcal{D}_i$. This method compromises between a batch training and the single-data training, and concretely, it is more robust than the single-data training. The gradient learning method using minibatch is the well-known stochastic gradient descent (SGD) method and is justified by the Robbins-Monro stochastic approximation method [13].

Replay buffer is a specific batch RL method which stores simulated data [142]. The illustration is provided in Fig. 4.1. At each iteration, minibatches of $\mathcal{D}$ and $\mathcal{D}_T$ are prepared independently by uniform sampling from the replay buffer to estimate gradients in Eq. (4.13).

The rationale for constructing the minibatch data by uniform sampling of the replay buffer is to reduce the correlations between RL data sets and make them independently and identically distributed (i.i.d.). First, contrary to the supervised learning where a fixed training data set are given, time-varying dataset is received in RL. By uniformly sampling from the replay buffer, overfitting with a recent data set is prevented, and thus temporal correlation is reduced. Second, there also exist correlations within a data set, because it contains state trajectories generated by a control policy that maps a state to an action. Decorrelated minibatch data satisfies the i.i.d. condition which is a crucial assumption made on machine learning training data sets [43]. Note that the size of $\mathcal{D}$ should be calibrated by observing the trade-off between stability and training speed.

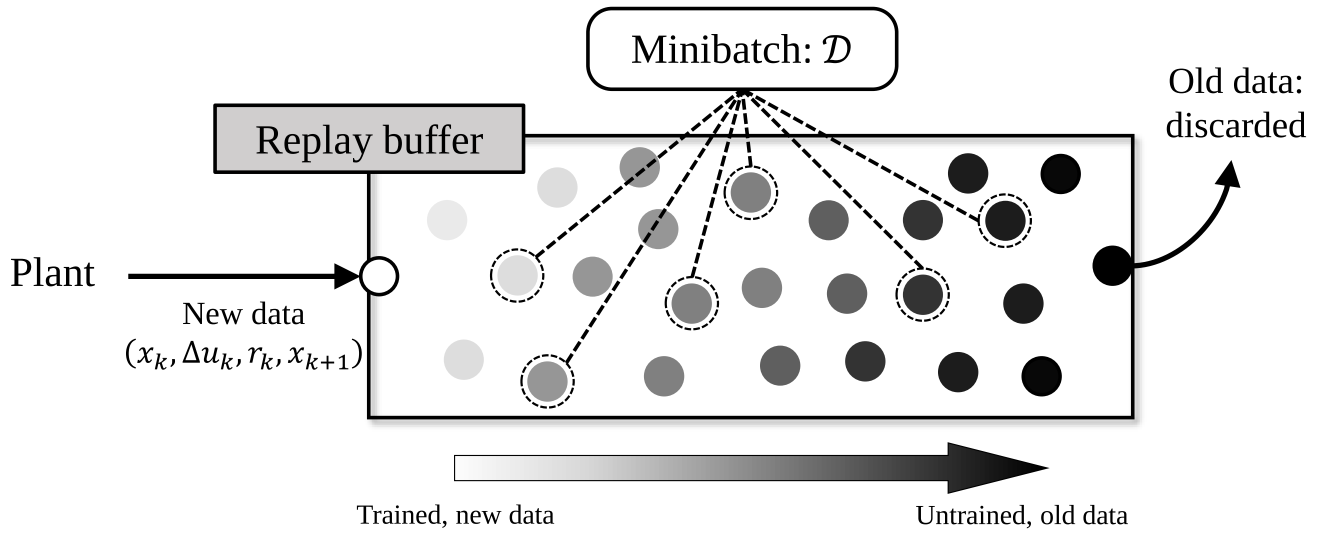


Figure 4.1: Illustration of replay buffer. The data structure of the replay buffer takes first-in-first-out (FIFO) form, while the total size of the replay buffer is kept the same. Minibatch $\mathcal{D}$ for training is randomly sampled from the replay buffer.

4.2.2.2 Target network

GDHP uses bootstrapping to estimate the value and costate functions. When computing the residuals in value and costate networks (Eqs. (4.4) and (4.5)), the weights at time steps k (current) and $k+1$ (target) are both used. In the training process, as the values of the weights for the value, costate, and policy networks change to match the bootstrapped target values, the target values also vary concurrently. For example, the gradient of $\|\hat{V}(X_{\mathcal{D}}) - \tau_V\|_2^2$, the first term of Eq. (4.10), is

$$2\left(\frac{\partial vec(\hat{V}(X_{\mathcal{D}}) - \hat{V}(X_{\mathcal{D}}^+))}{\partial vec(\hat{W}_{V,l})}\right)^T \left(\hat{V}(X_{\mathcal{D}}) - R_{\mathcal{D}} - \hat{V}(X_{\mathcal{D}}^+)\right) \quad (4.14)$$

where $R_{\mathcal{D}}$ denotes the stacked vector of $[r_{k_1}, \ldots r_{k_{|\mathcal{D}|}}]$. Since $X_{\mathcal{D}}$ and $X_{\mathcal{D}}^+$ both consisted of noisy measurements, the gradient Eq. (4.14) has high variance due to the arithmetic operations of stochastic terms, which can cause instability.

The target network method alleviates the side-effects from the bootstrapping method [143] by evaluating the target values τ_V, τ_λ and τ_π from the separate networks called target networks. The weights are set differently in the current and target networks while maintaining the same network structures. Denote $\hat{\psi}'$ as the target networks, then the modified targets for Eqs. (4.4) - (4.6) are written as

$$\begin{aligned} \tau'_{V,k} &= r_k + \hat{V}'(x_{k+1}) \quad (4.15) \\ \tau'_{\lambda,k} &= 2\left(\frac{\partial h(x_k)}{\partial x_k}\right)^T Q\left(h(x_k) - \rho_k\right) \\ &\quad + \left(\frac{\partial x_{k+1}}{\partial x_k}\right)^T \hat{\lambda}'(x_{k+1}) \quad (4.16) \\ \tau'_{\pi,k} &= -\frac{1}{2}R^{-1}g^T(x_k)\hat{\lambda}'(x_{k+1}) \quad (4.17) \end{aligned}$$

In this case, the gradient Eq. (4.14) becomes

$$2\left(\frac{\partial vec(\hat{V}(X_{\mathcal{D}}))}{\partial vec(\hat{W}_{V,l})}\right)^T \left(\hat{V}(X_{\mathcal{D}}) - R_{\mathcal{D}} - \hat{V}'(X^+_{\mathcal{D}})\right) \quad (4.18)$$

since the dependency of $\hat{W}_{V,l}$ with respect to the target value function $\hat{V}'(X^+_{\mathcal{D}})$ is removed, hence the variance of the gradient estimate becomes smaller. Let $\hat{W}'_{\psi,l}$ as the weights of the l^{th} layer of target networks. Then the learning rule for the target networks is expressed as a weighted sum of the target and current networks

$$\hat{W}'^{+}_{\psi,l} = \tau\hat{W}^{+}_{\psi,l} + (1-\tau)\hat{W}'_{\psi,l}, \quad \tau \ll 1 \quad (4.19)$$

where $\hat{W}'^{+}_{\psi,l}$ is the updated target network weights, and τ is the weighting factor which indicates how soft the update should be [43]. The rule intentionally slows down the learning rate by slowing the changes in the target value, which greatly improves the stability of the learning.

4.2.2.3 Pre-training

Eqs. (4.10) - (4.12) show that the value, costate, and policy networks are all functions of each other. This cyclic relationship makes the GDHP method prone to divergence. Consequently, choosing a good initialization of the policy network and adding exploration are important for stable learning [144]. Existing control methods such as the linear quadratic regulator (LQR) or model predictive controller (MPC) provide good starting suboptimal policies which can be used for initialization. In the ensuing examples, the LQR integrator (LQI) is implemented as an initial control policy. Let us denote the initial policy as $l(x_k, k)$. Furthermore, a noise sequence $\mathcal{N}$ can be added to the controller's output for exploration (i.e., $u_k = u_k^* + \mathcal{N}(k)$) in order to explore the vicinity neighborhoods of the suboptimal trajectories.

4.2.2.4 Discretization

Obtaining the discrete-time state space model in Eq. (4.20) involves the numerical integration method. To reduce the numerical error, rather than using the first-order Euler method (i.e., zero-order hold(ZOH), a higher-order discretization method is implemented. We implemented Dormand-Prince (DOPRI5) method which is the most widely used among Runge-Kutta family [145, 146]. Given the system dynamics $\dot{x} = f(t, x, u)$ with the initial condition, $x(t_0) = x_0$, the explicit Runge-Kutta method is given by

$$x_{k+1} = x_k + h \sum_{i=1}^{s} b_i p_i \tag{4.20}$$

, where

$$\begin{aligned}
p_1 &= f(t_k, x_k, u_k), \\
p_2 &= f\left(t_k + c_2 h, x_k + h(a_{21}p_1), u_k\right), \\
p_3 &= f\left(t_k + c_3 h, x_k + h(a_{31}p_1 + a_{32}p_2), u_k\right) \\
&\vdots \\
p_j &= f\left(t_k + c_j h, x_k + h\left(\sum_{i=1}^{j-1} a_{ji}p_i\right), u_k\right) \\
&= f(t_k^{(j)}, x_k^{(j)}, u_k) \\
&\vdots \\
p_s &= f\left(t_k + c_d h, x_k + h\left(\sum_{i=1}^{d-1} a_{di}p_i\right), u_k\right)
\end{aligned} \tag{4.21}$$

p_j is the j^{th} interpolated function value at an intermediate time and state points $t_k^{(j)}$ and $x_k^{(j)}$, respectively. h denotes the discrete time step and d denotes the number of discretization stages. Coefficients a_{ij}, b_i and c_i are given by the Butcher tableau. The DOPRI5 method uses discretization stages of $d = 5$. Then the derivative of the successor state with respect to the current state can be evaluated by the following explicit formula:

$$\begin{aligned}
\frac{\partial x_{k+1}}{\partial x_k} &= I + h\sum_{i=1}^{d} b_i \frac{\partial p_i}{\partial x_k} \\
&= I + h\sum_{i=1}^{d} b_i \left(\frac{\partial f(t_k^{(i)}, x_k^{(i)}, u_k)}{\partial x_k^{(i)}}\right)^T \frac{\partial x_k^{(i)}}{\partial x_k}
\end{aligned} \tag{4.22}$$

, where

$$\frac{\partial x_k^{(j)}}{\partial x_k} = I + h \sum_{i=1}^{j-1} \left(\frac{\partial x_k^{(i)}}{\partial x_k} \right)^T a_{ji} \tag{4.23}$$

The control derivative is computed in the same way:

$$\begin{aligned} \frac{\partial x_{k+1}}{\partial u_k} &= h \sum_{i=1}^{d} b_i \frac{\partial p_i(x_k)}{\partial u_k} \\ &= h \sum_{i=1}^{d} b_i \frac{\partial f(t_k^{(i)}, x_k^{(i)}, u_k)}{\partial u_k} \end{aligned} \tag{4.24}$$

4.2.3 Overall algorithm

The overall algorithm is summarized in Algorithm 1. Structures and training schemes of the value, costate, and policy networks with the target networks incorporated are illustrated in Fig. 4.2. The overall schematic diagram is illustrated in Fig. 4.3.

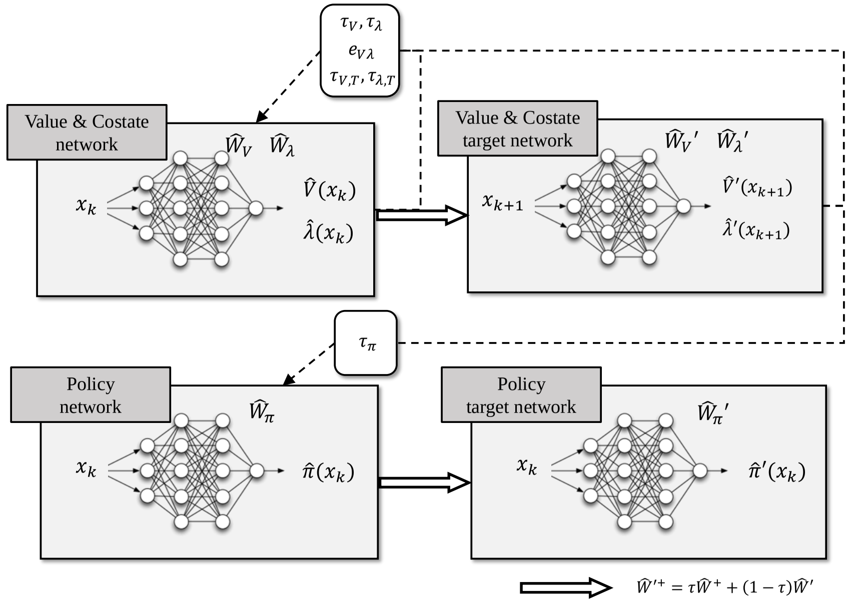


Figure 4.2: Structures and training schemes of the value, costate, and policy networks. Original networks are trained with the signals from the target networks, while the target networks weights are obtained by the soft update rule.

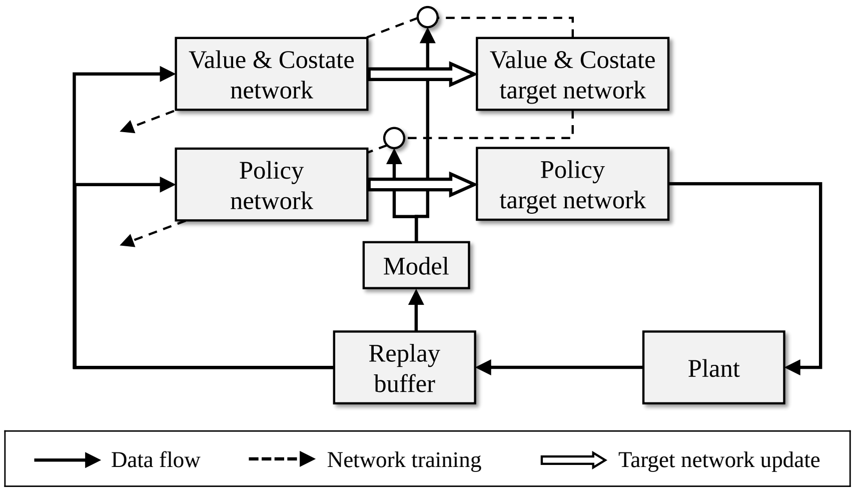


Figure 4.3: A schematic diagram of GDHP algorithm

Algorithm 1 GDHP

```
1: procedure GDHP WITH DNNs
   Initialize: Initialize critic, costate, and policy networks $\hat{V}$, $\hat{\lambda}$ and $\hat{\pi}$
   Initialize critic, costate, and policy target networks $\hat{V}'$, $\hat{\lambda}'$, and $\hat{\pi}'$
   Initialize replay buffer $\mathcal{D}$
   Initialize terminal replay buffer $\mathcal{D}_T$
   Initialize noise process $\mathcal{N}$
   Discretize and augment the system dynamics as Eq. (2.25) and obtain
   derivatives by Eqs. (4.20), (4.22) and (4.24)
2:     for $i = 1$ to $E$ do
3:         for $k = 1$ to $T$ do
4:             if $i \leq$ Initial policy then
5:                 $\Delta u_k = l(x_k) + \mathcal{N}(k)$
6:             else
7:                 $\Delta u_k = \hat{\pi}^{(i)}(x_k) + \mathcal{N}(k)$
8:             Get next augmented state $x_{k+1}$ and stage-wise cost $r_k$ with
   control input $\Delta u_k$ by Eqs. (2.25) and (2.15)
9:             if $k == T$ then
10:                Store $(x_T, \phi_T)$ to $\mathcal{D}_T$
11:            else
12:                Store $(x_k, u_k, r_k, x_{k+1})$ to $\mathcal{D}$
13:            Uniform sample of minibatch $\tilde{\mathcal{D}}$ from $\mathcal{D} \cup \mathcal{D}_T$
14:            Compute residual functions $L_V(\tilde{\mathcal{D}})$, $L_\lambda(\tilde{\mathcal{D}})$, and $L_\pi(\tilde{\mathcal{D}})$ by
   Eqs. (4.10)-(4.12)
15:            Update the weights $\hat{W}^+_{V,l}$, $\hat{W}^+_{\lambda,l}$, and $\hat{W}^+_{\pi,l}$ by Eq. (4.13)
16:            Update the weights $\hat{W}'^+_{V,l}$, $\hat{W}'^+_{\lambda,l}$, and $\hat{W}'^+_{\pi,l}$ by Eq. (4.19)
```

4.3 Results and discussions

In this section, numerical results of applying the GDHP algorithm with DNNs on simulated systems are presented. The main objective for this section is to show that the benefit of using DNNs over SNNs in the cases where state space dimension is high and initial states as well as the state transition are subject to stochastic uncertainties.

4.3.1 Example 1: Semi-batch reactor

A benchmark semi-batch reactor presented in [147] is used. In the reactor, the following parallel second-order reactions occur:

$$\begin{aligned} A + B &\xrightarrow{k_1} C \\ B + C &\xrightarrow{k_2} D \end{aligned} \tag{4.25}$$

System equations are derived from mass and energy balances as follows:

$$\begin{aligned}
\dot{C}_A &= -\frac{Q_f}{V} C_A - k_1(T) C_A C_B \\
\dot{C}_B &= \frac{Q_f}{V}(C_{B_f} - C_B) - k_1(T) C_A C_B - k_2(T) C_B C_C \\
\dot{C}_C &= -\frac{Q_f}{V} C_C + k_1(T) C_A C_B - k_2(T) C_B C_C \\
\dot{T} &= \frac{Q_f}{V}(T_f - T) - \frac{1}{\rho C_p}\big(\Delta H_1 k_1(T) C_A C_B \\
&\quad + \Delta H_2 k_2(T) C_B C_C\big) - \frac{UA}{\rho C_p V}(T - T_K) \\
\dot{V} &= Q_{fr}, \quad Q_{fr} = \begin{cases} 0 & \text{if} \quad t < 31 \text{ min} \\ Q_f & \text{otherwise} \end{cases}
\end{aligned} \tag{4.26}$$

where C_A, C_B, and C_C denote the molar concentrations of substances A, B, and C, and T and V are the temperature and volume of the reactor, respectively. The kinetic parameters are functions of the temperature following the Arrhenius law

$$k_i(T) = k_{i_0} \exp\left(-\frac{E_i}{RT}\right), \quad i \in \{1, 2\} \tag{4.27}$$

Initial conditions of states are $C_{A_0} = 1$ mol/L, $C_{B_0} = 0$ mol/L, $C_{C_0} = 0$ mol/L, $T_0 = 298$ K, and $V_0 = 50$ L. The manipulate variables (MV) are the flow rate of substance B ($Q_f(t)$) and the jacket temperature ($T_K(t)$). The controlled variables (CV) are the reactor temperature ($T(t)$) and product yield ($V(t)C_C(t)$). Other physical parameters are specified as: $T_f = 308$ K, $C_{B_f} = 0.9$ mol/L, $UA/\rho C_p = 3.75$ L/min, $k_{1_0} = 5.0969 \times 10^{16}$ L/mol min, $k_{2_0} = 2.2391 \times 10^{17}$ L/mol min, $E_1/R = 12305$ K, $E_2/R = 13450$ K, $\Delta H_1/\rho C_p = -28.5$ KL/mol, and $\Delta H_2/\rho C_p = -20.5$ KL/mol. The compact subsets of state, MVs, and CVs are described in Table 4.1.

Table 4.1: Minimum and maximum values of the state, MVs, and CVs in the operating region

Variables	Minimum	Maximum
C_A (mol/L)	0	1
C_B (mol/L)	0	1
C_C (mol/L)	0	1
T (K)	293.15	323.15
V (L)	50	150
Q_f (L/min)	0	1.5
T_K (K)	293.15	318.15
VC_C (mol)	0	90

The reactor is operated in the following scheme: While the reactant A is initially charged the temperature is maintained with a room temperature 298.15 K. After the start-up process until $t = 30$ min, the second reactant B is fed and reaction occurs. The temperature is regulated to 308.15 K until $t = 80$ min and gradually cooled down to 303.15 K until the batch terminal time. Finally, the product C is discharged with the target yield (42 mol). The objective is formally represented as

$$J = \|V(t_T)C_C(t_T) - \varphi\|_H^2 + \sum_{i=0}^{T-1}\Big(\|T(t_i) - \rho(t_i)\|_Q^2 + \|\Delta u_i\|_R^2\Big) \tag{4.28}$$

where the temperature reference trajectory $\rho(t)$ and the target yield φ are specified as

$$\rho(t) = \begin{cases} 298.15 \text{ K}, & 0 \le t < 3 \\ 308.15 \text{ K}, & 30 \le t < 80 \\ 303.15 \text{ K}, & 98 \le t \end{cases}, \qquad \varphi = 42 \text{ mol}$$

We used weight matrices as $Q = \text{diag}(0.1, 0.1)$, $R = \text{diag}(0.01, 0.01)$, and $H = \text{diag}(1, 1)$, where diag denotes the diagonalization operator. Sampling time was set as 0.5 minute and the horizon was 100 minutes. The state space dynamics is normalized into [-1, 1] by applying the linear transformation $X = \dfrac{2x - x_M - x_m}{x_M - x_m}$, where X is the normalized state and control values and x_M and x_m are the maximum and minimum values for corresponding variables, respectively.

4.3.1.1 Specification of hyperparameters

Five-layer DNNs were used for the value and costate networks and four-layer DNNs were used for policy network. The number of nodes in the value, costate, and policy networks were ($S + A + 1$, 75, 75, 45, 1), ($S + A + 1$, 75, 75, 75, $S + A + 1$), and ($S + A + 1$, 150, 90, A), respectively, referring the network structure of [135]. Leaky ReLU activation functions were used for hidden layers. The size of the replay buffer $|\mathcal{D}|$ was set to be 10 times of a single episode length and terminal replay buffer size $|\mathcal{D}_T|$ was 10. Learning rates η_V, η_λ, and η_π were all 10^{-4}. Loss functions weighting factors were set to be $\beta_{V,1}, \beta_{\lambda,1} = 10^{-2}$, and $\beta_{V,2}, \beta_{\lambda,2} = 1$. The target network weighting factor τ was 0.05. We measured the episode loss functions $\mathcal{L}_\psi^{(i)} = \sum_{k=0}^{T-1} L_\psi(\tilde{\mathcal{D}}_{k,i})$, where $\tilde{\mathcal{D}}_{k,i}$ denotes the sampled minibatch at time step k and iteration number i, to determine the convergence criteria: $\|\mathcal{L}_\psi^{(i+1)} - \mathcal{L}_\psi^{(i)}\|/\|\mathcal{L}_\psi^{(i)}\| < 10^{-5}$, $\psi \in \{V, \lambda\}$.

In order to have an efficient initialization of the DNNs, the linear quadratic integral control (LQI), which is a variant of LQR for the tracking problem, was used as the initial policy $l(x_k)$. At every time step, the system dynamics were linearized by computing the discretized Jacobian with respect to the current state x_k, and the LQI gain was obtained by solving the steady-state Riccati equation. The LQI controller was employed until episode 25 to update just the value and costate networks. After that, the policy function resulting from the iteration with the data from the previous episode was implemented until convergence.

4.3.1.2 Tracking results in the deterministic system case

The tracking results for the deterministic system case are discussed in this section. The trajectories of the states and CVs under several control polices at different training stages of the algorithm are presented in Figs. 4.4 and 4.5. In particular, plotted trajectories are for the 20th, 75th, 120th, and 200th episodes, each of which is being operated under different conditions, i.e., the LQI policy , intermediate GDHP policies yet to converge, and the final converged policy, respectively. The trajectories of the CVs in Fig. 4.5 indicate that the final GDHP policy performs quite well in both the tracking of the reactor temperature and satisfying the boundary condition for the product yield, while the LQI policy in episode 25 fails to do so. Figure. 4.6 shows the log vlau episode loss function of value and costate networks and the episode cost. The episode loss functions continue to decrease throughout the scenario, except for having high values right after the control policy being switched from initial LQI policy to the GDHP policy. Since the episode loss functions $\mathcal{L}_V$ and $\mathcal{L}_\lambda$ measures the deviations of optimality equations Eqs. (2.48) - (2.50), costate function definition Eq. (2.19), and the boundary conditions Eq. (2.24), we can conclude that all deviations become zero. Figure. 4.6 indicates that the converged GDHP policy shows smaller episode cost than that of the initial LQI polcy.

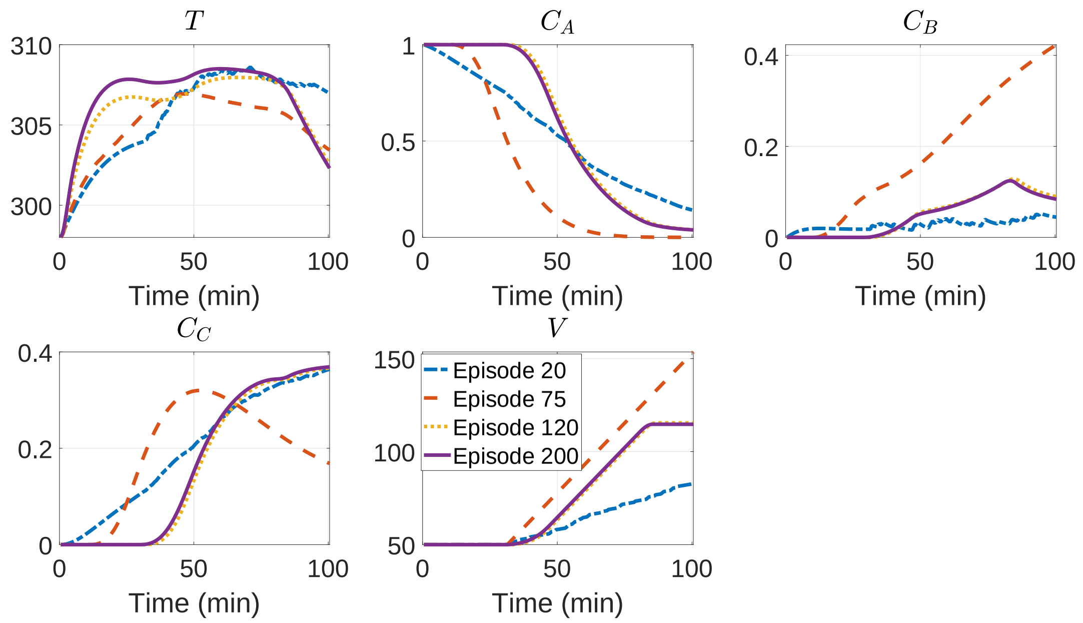


Figure 4.4: State trajectories of GDHP at episodes 20, 75, 120, and 200.

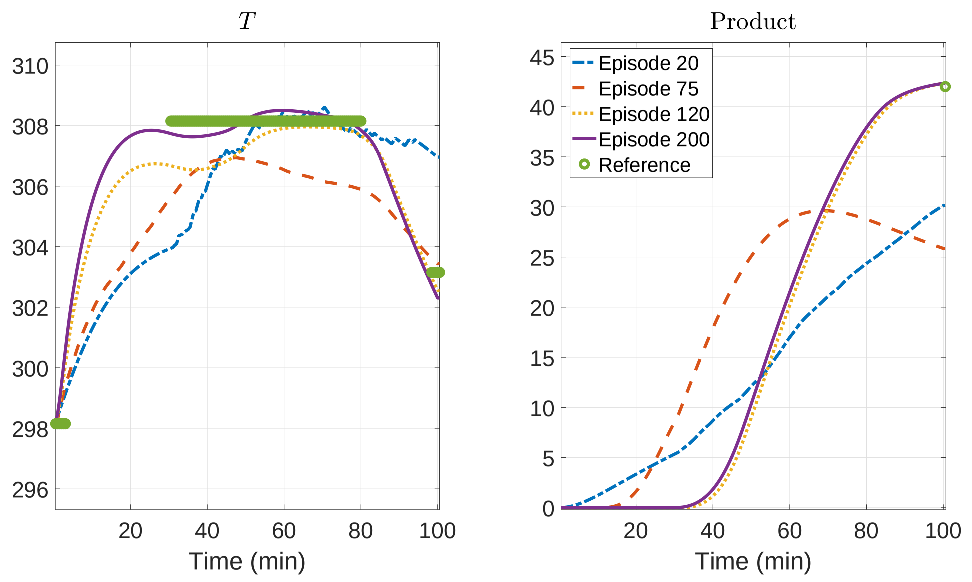


Figure 4.5: Trajectories of the CVs of GDHP at episodes 20, 75, 120, and 200. Circle mark for the first graph denotes the reference trajectory of the reactor temperature, and the second graph denotes the boundary condition for the product yield.

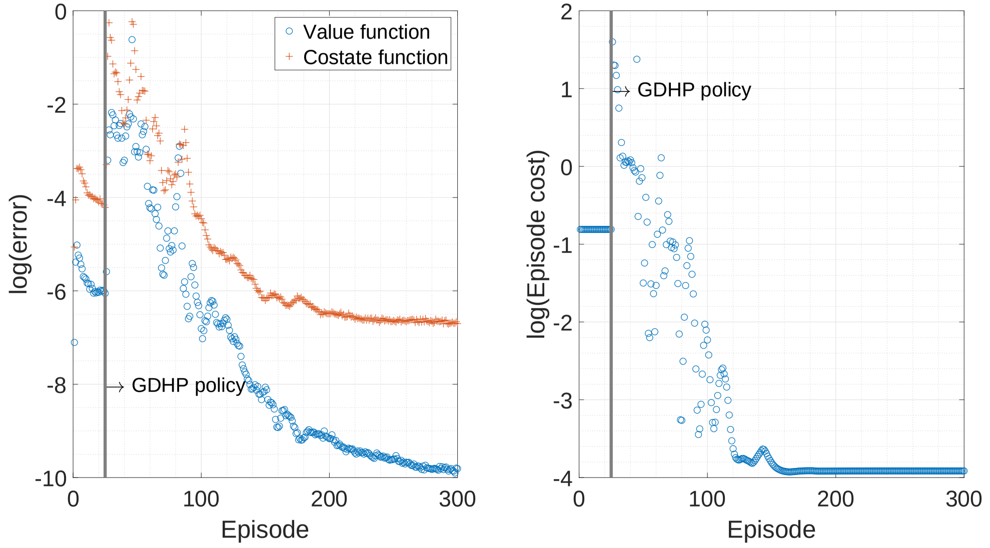


Figure 4.6: Log values of episode loss functions $\mathcal{L}_V$ and $\mathcal{L}_\lambda$ and episode cost. Initial LQI controller was implemented before episode 25 and GDHP policy was applied after episode 25.

4.3.1.3 Tracking results under state noise

This section addresses the tracking control problem in the presence of the state noise. We compare the tracking results under four different policies represented by DNNs or SNNs trained with data containing Gaussian state noise of $\mathcal{N}(0, 0.005)$ and by DNNs and SNNs trained with data from the deterministic system, respectively. In the test case, four scenarios with Gaussian noise of variances range from 0.002 to 0.008 were considered. Note that 20 episodes were simulated for each of the scenarios. We assume that the system starts from a fixed initial state and uncertainties arise from the state noise only. The mean episode costs for the four test cases are illustrated in Fig. 4.7.

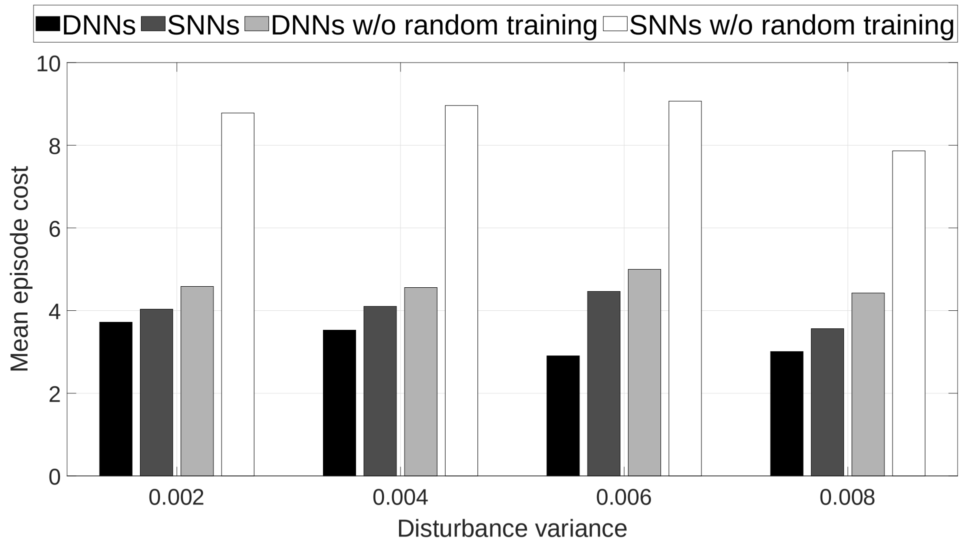


Figure 4.7: Mean episode costs for 20 scenarios under policies trained with DNNs and SNNs with state noises, and DNNs and SNNs with the deterministic system. Mean episode costs were obtained based on the results from the simulation of 20 episodes.

The control policy from the DNNs trained with the data containing state noises shows the best performance among the four policies. Note that in the case with state noise, both of the SNNs policies result in failures compared to the DNNs case, since the state trajectories show noisy behavior throughout the entire horizon and require an approximator with a larger number of parameters. The generalizability of the DNNs lead to a better feedback policy in the presence of uncertainty, while SNNs cannot.

4.3.1.4 Tracking results under initial state uncertainty

The model based GDHP algorithm should be able to give a control policy, that works well with multiple initial states. Let us assume that the initial state $p(x_0)$ follows a certain probability distribution, and the initial state realized in the off-line training process is sampled from this distribution. We assumed that the initial state is subject to the uniform distribution whose support is 0.3 times of the minimum and the maximum values of the operating region (Table 4.1). After 200 episodes training, the resulting policy was tested for the scenarios with twenty different initial state conditions which were randomly generated from the uniform distribution which has significantly larger supports (i.e., 0.7 times of minimum and maximum values of the operating regions) than that used in the training process. Figure 4.8 shows the mean episode costs with respect to four different policies from the DNNs and SNNs trained with randomly sampled initial states and DNNs and SNNs with a single fixed initial state, respectively.

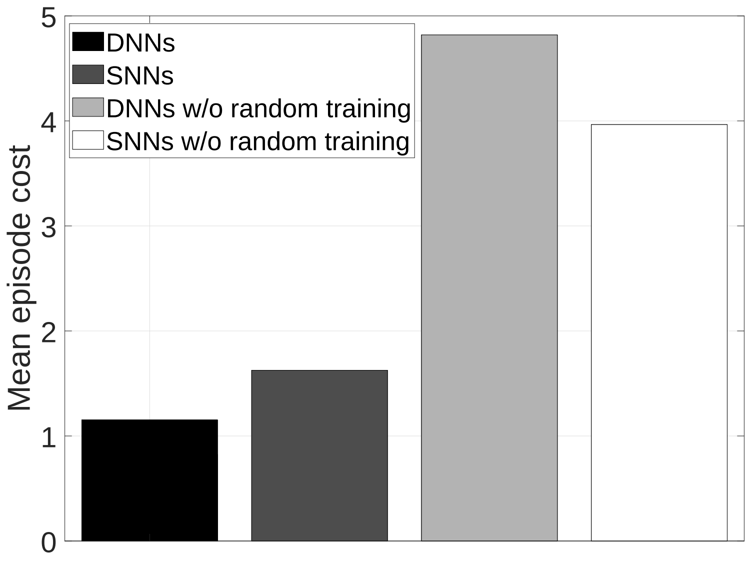


Figure 4.8: Mean episode costs for episodes with 20 randomly sampled initial states under policies trained with DNNs and SNNs with random initial states, and DNNs and SNNs with a deterministic initial state.

DNNs trained with random initial states show the best result among the four policies. However, the DNNs trained without the random initial states setting showed the worst result. The results indicate that DNNs are more vulnerable to overfitting than SNNs, hence give worse performance when trained with a single fixed initial condition while the initial state in actual on-line varies.

In summary, GDHP with DNNs provide robustness when initial state uncertainties are modelled and taken into account during the training process. When the online uncertainty pushes the system outside the region experienced offline, the larger parameter space of the DNNs has problems from overfitting. Thus the suggested algorithm requires careful generations of simulation data coupled with uncertainty quantifications.

4.3.2 Example 2: Diffusion-Convection-Reaction (DCR) process

Performance of the GDHP algorithm with DNNs is examined in the context of a system with high-dimensional state and control space, such as a spatially distributed parameter system expressed by PDEs. We use the example of a typical deterministic diffusion-convection-reaction (DCR) process in one-dimensional rod [148]. Consider an exothermic reaction $A \rightarrow B$, and the temperature is to be controlled by a distributed actuator along the same geometry. The dimensionless temperature profile denoted as $x(z,t)$ is described by the following

PDE:

$$\begin{aligned}\frac{\partial x}{\partial t} &= (1-\alpha_{dc})\frac{\partial}{\partial z}\left(k(z)\frac{\partial x}{\partial z}\right) + \alpha_{dc}\frac{\partial x}{\partial z} \\ &\quad + \beta_T(z)\left(e^{-\gamma/(1+x)} - e^{-\gamma}\right) + \beta_U(u(z,t) - x)\end{aligned} \tag{4.29}$$

with the Dirichlet boundary conditions and the initial condition of

$$x(0,t) = 0, \quad x(\pi,t) = 0, \quad x(z,0) = 0.5 \tag{4.30}$$

where z is the one-dimensional spatial coordinate, and $u(z,t)$ denotes the control input profile. Terms in the PDE denote the diffusion, convection, reaction, and control related terms, respectively. α_{dc} is the weight between the and convection terms; $k(z)$ expresses the diffusion coefficient; $\beta_T(z)$ is the heat of reaction along the coordinate; γ is the activation energy, and β_U is the heat transfer coefficient. All parameters have dimensionless values. Specifically, $\alpha_{dc} = 0.5$, $k(z) = 0.5 + 0.7/(1+z)$, $\beta_T(z) = 16(\cos(z) + 1)$, $\gamma = 4$, and $\beta_U = 1$.

The objective is to control the state $\bar{x}(z,t)$, $z \in [0,\pi]$ to the zero setpoint. The objective is written as:

$$J = \int_0^{\pi}\left(\|\bar{x}(z,t_T)\|_H^2 + \sum_{i=0}^{T-1}\left(\|\bar{x}(z,t_i)\|_Q^2 + \|\Delta u(z,t_i)\|_R^2\right)\right)dz \tag{4.31}$$

The positive definite control weighting matrices Q and R were set to be identity matrices with appropriate dimensions. This system has been examined in [149] where the HDP algorithm was applied to a reduced-order model, but the control dimension was limited to one.

For numerical simulations and Jacobian calculation of the PDE, we applied the finite difference discretization for the spatial derivative and used the method-of-line for the time derivative. Specifically, the central difference method was used for the second-order derivative in the diffusion term, and the upwind difference scheme (UDS) for the first-order derivative in the convection term. The spatial coordinate z was discretized into 40 equidistant intervals, and the sampling time and the horizon were chosen as 0.005 second and 4 seconds, respectively. Since it took too much computation time to solve the Riccati equation for the system dimensions $S = 40$, $A = 40$, and $O = 40$, which needed to construct the LQI, the following manually designed policy was used as the initial policy instead [149]:

$$u(z,t) = \begin{cases} -2, & 0.2\pi \le z \le 0.4\pi \\ 0, & \text{otherwise} \end{cases}$$

The GDHP algorithm was applied for 200 episodes with the same algorithmic hyperparameters as the previous example. The state and control trajectories at episode 10, 30, 35, and 200 are shown in Figs. 4.9 and 4.10. The plot of the state profile shows a suboptimal trajectory when the GDHP policy is first implemented at episode 25, but it quickly stabilized to the zero setpoint by episode 35. According to Fig. 4.11, episode loss functions of value and costate networks continue to decrease by episode 200. It also indicates the converged GDHP policy at episode 200 shows improved episode cost than that of the initial LQI policy. The GDHP algorithm with DNNs is shown to be effective in handling high-dimensional state space systems in this example.

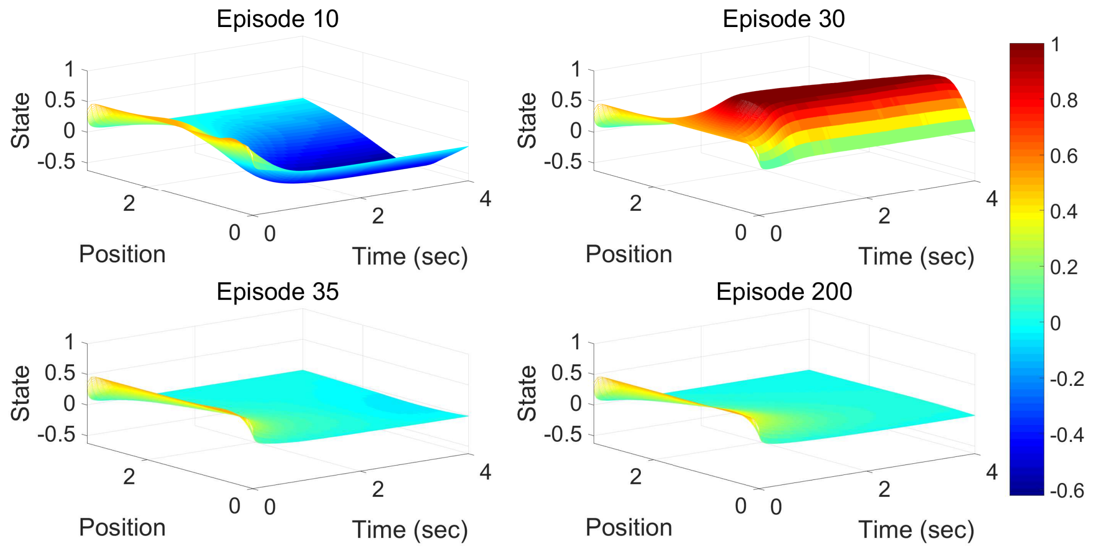


Figure 4.9: State trajectories of the DCR process under the GDHP algorithm at episodes 10, 30, 35, and 200.

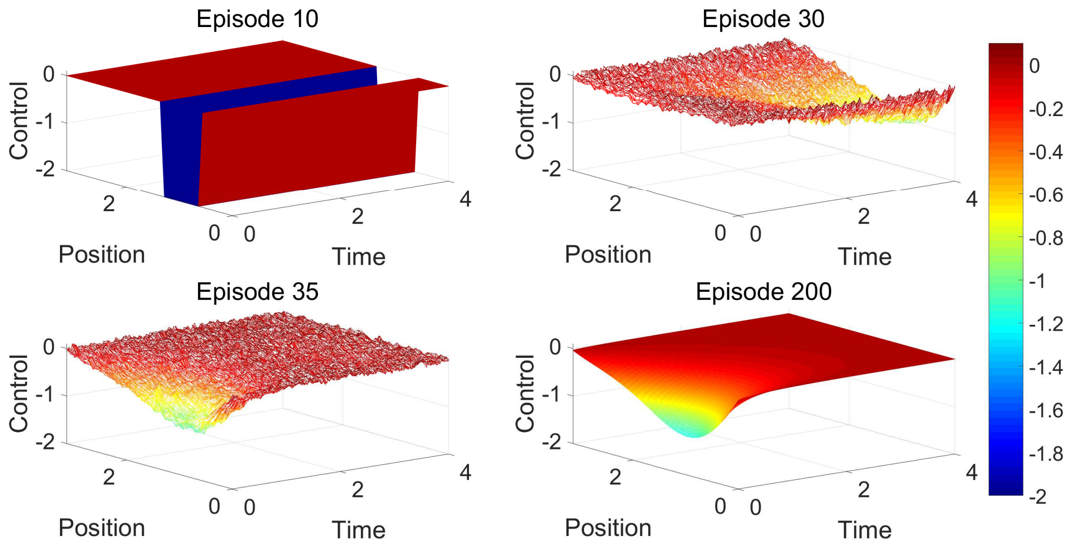


Figure 4.10: Control trajectories of the DCR process under the GDHP algorithm at episodes 10, 30, 35, and 200.

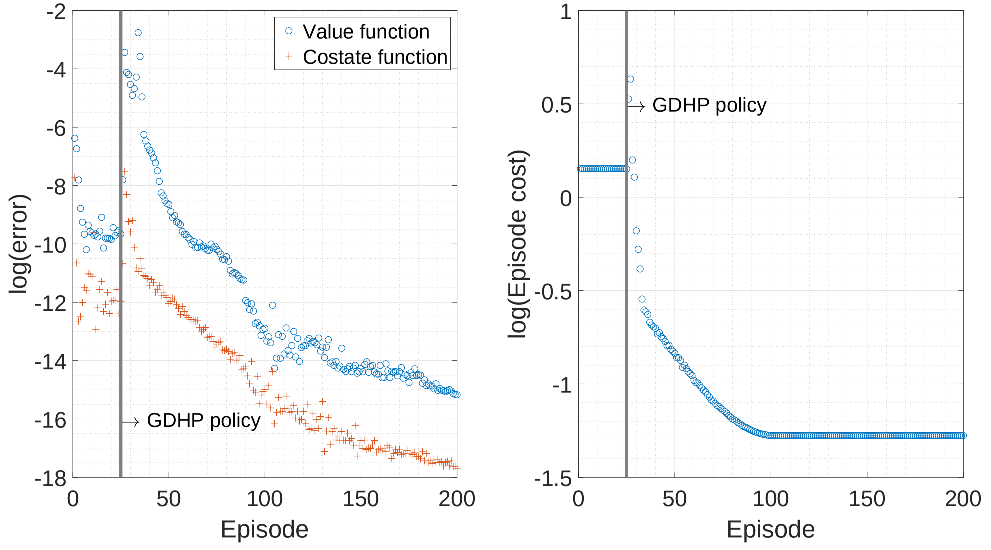


Figure 4.11: Log values of episode loss functions $\mathcal{L}_V$ and $\mathcal{L}_\lambda$ and episode cost. Initial LQI controller was implemented before episode 25 and the GDHP policy was applied after episode 25.

Chapter 5

Convergence analysis of the model-based deep reinforcement learning for optimal control of nonlinear control-affine system [3]

5.1 Introduction

In the case of control-affine system and quadratic cost function, the solutions of HJB equation are expressed with respect to the value function and its first-order derivative (i.e., costate function). The use of model knowledge not only enables the analysis of the closed-loop stability [34] but also reduces the amount of samples in the learning-based algorithm [37]. The model-based methods vary with the level of incorporating the state-space model, which are heuristic dynamic programming (HDP), dual HDP (DHP), and globalized DHP (GDHP) [61]. The HDP algorithm approximates the value function, for which the convergence analysis is studied in [151, 152] and its optimality bound is provided in [153]. In the case of DHP and GDHP, where the costate function is approximated, the convergence is only studied for the linear system [154]. It is also shown that the

[3]This chapter is an adapted version of J. W. Kim, T. H. Oh, S. H. Song, D. W. Jeong, and J. M. Lee, “Convergence analysis of the model-based deep reinforcement learning for optimal control of nonlinear control-affine system," *Automatica*. Under review. [150]

GDHP algorithm taking both the value and the costate functions into account yields the best result among the three methods [61]. Hence, this study performs the convergence analysis of GDHP algorithm for the nonlinear control-affine system.

The choice of function approximator is one of the key elements of the learning-based methods. The value, costate, and policy are expressed in a parametric form, for which the neural networks (NN) is the most widely incorporated function. The single-layer NN (SNN) has been applied in most studies because its linearity brings the simplicity to both the algorithm design and the stability analysis. The NN parameters are obtained by the least-squares method in the early literature [152, 155, 156, 157], while gradient descent method is implemented recently [158, 159]. The uniformly ultimately boundedness (UUB) results with respect to the NN parameters and closed-loop stability in HDP algorithm are provided [160, 159]. This study provides an equivalent UUB result for GDHP algorithm combined with function approximator.

Recently, the development of deep NNs (DNNs) allows for constructing the high-dimensional function approximator, which is applied to the optimal control problem to develop the deep RL (DRL) algorithm. The success of DRL motivates us to propose a novel algorithm which utilizes the DNNs structure within the GDHP algorithm and to provide the novel convergence analysis regrading the deep approximator.

5.2 Convergence proof of globalized dual heuristic programming (GDHP)

In this section, an iterative algorithm for solving the HJB equation, referred to as GDHP algorithm, is presented. The value, costate, and policy functions are sequentially updated to solve Eq. (2.30) numerically as presented in Section 2.2.2.

Lemma 5.1. *If $V^{(1)}(x_k) \leq V^{(0)}(x_k)$ holds for all x_k, the sequence $\{V^{(i)}(x_k)\}$ is a nonincreasing sequence satisfying that $V^{(i+1)}(x_k) \leq V^{(i)}(x_k)$ for all x_k.*

Proof 5.1. *Define a new sequence $Z^{(i)}(x_k)$ generated by an arbitrary control policies μ_k initialized by $Z^0(\cdot) = V^{(0)}(\cdot)$. Then the sequence satisfies following equation.*

$$Z^{(i+1)}(x_k) = r(x_k, \mu_k) + \gamma Z^{(i)}\big(f(x_k) + g(x_k)\mu_k\big) \tag{5.1}$$

$V^{(0)}(x_k) \leq Z^{(0)}(x_k)$ and assume that $V^{(i)}(x_k) \leq Z^{(i)}(x_k), \forall x_k$. Then

$$\begin{aligned} Z^{(i+1)}(x_k) &= r(x_k, \mu(x_k)) + \gamma Z^{(i)}\big(f(x_k) + g(x_k)\mu_k\big) \\ &\geq r(x_k, \mu(x_k)) + \gamma V^{(i)}\big(f(x_k) + g(x_k)\mu_k\big) \\ &\geq r(x_k, \pi_k^{(i)}) + \gamma V^{(i)}(f(x_k) + g(x_k)\pi_k^{(i)}) \\ &= V^{(i+1)}(x_k) \end{aligned} \tag{5.2}$$

Thus, $V^{(i+1)}(x_k) \leq Z^{(i+1)}(x_k), \forall i$ is obtained by mathematical induction.

Consider the sequence in which $\mu(x_k)$ is substituted by $\pi_k^{(i-1)}$ as,

$$\begin{aligned} Z^{(i+1)}(x_k) &= r(x_k, \pi_k^{(i-1)}) + \gamma Z^{(i)}(x_{k+1}) \\ Z^{(1)}(x_k) &= r(x_k, \pi_k^{(0)}) + \gamma Z^{(0)}(x_{k+1}) \end{aligned} \tag{5.3}$$

Then $Z^{(1)}(x_k) = V^{(1)}(x_k) \leq V^{(0)}(x_k)$ *by assumption. Assume that* $V^{(i-1)}(x_k) - Z^{(i)}(x_k) \geq 0, \quad \forall x_k$ *holds, then*

$$\begin{aligned} & V^{(i)}(x_k) - Z^{(i+1)}(x_k) \\ & = \gamma\big(V^{(i-1)}(x_{k+1}) - Z^{(i)}(x_{k+1})\big) \geq 0 \end{aligned} \tag{5.4}$$

we have $Z^{(i+1)}(x_k) \leq V^{(i)}(x_k), \forall i$ *by mathematical induction.*

From $V^{(i+1)}(x_k) \leq Z^{(i+1)}(x_k)$ *and* $Z^{(i+1)}(x_k) \leq V^{(i)}(x_k)$*, the result,* $V^{(i+1)}(x_k) \leq V^{(i)}(x_k), \forall i$ *is satisfied.* $\square$

The following theorem implies the uniform convergence of the value function sequence, and provides the upper bound of the suboptimality at each iteration step.

Theorem 5.1. *Assume that the following inequalities hold,*

$$0 \leq \gamma V^*(f(x_k) + g(x_k)u_k) \leq \theta r(x_k, u_k) \tag{5.5}$$

for all x_k*, where* $0 \leq \theta$ *and* $\alpha_V V^* \leq V^{(0)} \leq \beta_V V^*$ *for* $0 \leq \alpha_V \leq 1$ *and* $1 \leq \beta_V$*. Then the sequence* $\{V^{(i)}\}$ *generated from the GDHP algorithm satisfies the following inequalities:*

$$\begin{aligned} & \left(1 + \frac{\alpha_V - 1}{(1+\theta^{-1})^i}\right) V^*(x_k) \leq V^{(i)}(x_k) \\ & \qquad \leq \left(1 + \frac{\beta_V - 1}{(1+\theta^{-1})^i}\right) V^*(x_k) \end{aligned} \tag{5.6}$$

In addition, the sequence $\{V^{(i)}\}$ converges to V^* uniformly on the state space Ω.

Proof 5.2. The proof follows the procedure in [153]. First, consider the lower bound of the inequality by the mathematical induction. When $i = 1$,

$$\begin{aligned} V^{(1)}(x_k) &= r(x_k, \pi_k^{(0)}) + \gamma V^{(0)}(x_{k+1}) \\ &\geq r(x_k, \pi_k^{(0)}) + \alpha_V \gamma V^*(x_{k+1}) \end{aligned}$$

By assumption Eq. (5.5),

$$\frac{\alpha_V - 1}{1+\theta}\big(\theta r(x_k, \Delta u_k) - \gamma V^*(x_{k+1})\big) \leq 0$$

Then, the lower bound inequality holds in $i = 1$ as,

$$\begin{aligned} V^{(1)}(x_k) &\geq \left(1 + \theta\frac{\alpha_V - 1}{1+\theta}\right) r(x_k, \pi_k^{(0)}) \\ &\quad + \left(\alpha_V - \frac{\alpha_V - 1}{1+\theta}\right)\gamma V^*(x_{k+1}) \\ &= \left(1 + \frac{\alpha_V - 1}{1+\theta^{-1}}\right)\big(r(x_k, \pi_k^{(0)}) + \gamma V^*(x_{k+1})\big) \\ &\geq \left(1 + \frac{\alpha_V - 1}{1+\theta^{-1}}\right) V^*(x_k) \end{aligned} \tag{5.7}$$

Assume that the lower bound inequality holds for i, then

$$\begin{aligned}
V^{(i+1)}(x_k) &= r(x_k, \pi_k^{(i)}) + \gamma V^{(i)}(x_{k+1}) \\
&\geq r(x_k, \pi_k^{(i)}) + \left(1 + \frac{\alpha_V - 1}{(1+\theta^{-1})^i}\right)\gamma V^*(x_{k+1}) \\
&= \left(1 + \frac{(\alpha_V - 1)\theta^i}{(\theta+1)^i}\right) r(x_k, \pi_k^{(i)}) \\
&\quad + \left(1 + \frac{\alpha_V - 1}{(1+\theta^{-1})^{i-1}} - \frac{(\alpha_V - 1)\theta^{i-1}}{(\theta+1)^i}\right)\gamma V^*(x_{k+1}) \\
&= \left(1 + \frac{(\alpha_V - 1)\theta^i}{(\theta+1)^i}\right)\big(r(x_k, \pi_k^{(i)}) + \gamma V^*(x_{k+1})\big) \\
&\geq \left(1 + \frac{\alpha_V - 1}{(1+\theta^{-1})^i}\right) V^*(x_k)
\end{aligned} \tag{5.8}$$

Thus, the lower bound inequality of Eq. (5.6) is proved. Second, the upper bound can be proved similarly by the mathematical induction. When $i = 1$,

$$\begin{aligned}
V^{(1)}(x_k) \leq Z^{(1)}(x_k) & \\
&= r(x_k, \nu_k) + \gamma Z^{(0)}(x_{k+1}) \\
&= r(x_k, \nu_k) + \gamma V^{(0)}(x_{k+1}) \\
&\leq r(x_k, \nu_k) + \beta_V \gamma V^*(x_{k+1})
\end{aligned} \tag{5.9}$$

where $Z^{(i+1)}(x_k) = r(x_k, \nu_k) + \gamma Z^{(i)}(x_{k+1})$ with an arbitrary policy ν_k. By assumption Eq. (5.5),

$$\frac{\beta_V - 1}{1+\theta}\big(\theta r(x_k, \Delta u_k) - \gamma V^*(x_{k+1})\big) \geq 0 \tag{5.10}$$

, which can be added to Eq. (5.9) as,

$$
\begin{aligned}
V^{(1)}(x_k) &\leq \left(1 + \theta\frac{\beta_V - 1}{1+\theta}\right) r(x_k, \nu_k) \\
&\quad + \left(\beta_V - \frac{\beta_V - 1}{1+\theta}\right)\gamma V^*(x_{k+1}) \\
&= \left(1 + \frac{\beta_V - 1}{1+\theta^{-1}}\right)\big(r(x_k, \nu_k) + \gamma V^*(x_{k+1})\big)
\end{aligned} \tag{5.11}
$$

Since above holds for an arbitrary policy ν_k, it also holds for the optimal policy π_k^* as,

$$
\begin{aligned}
V^{(1)}(x_k) &\leq \left(1 + \frac{\beta_V - 1}{1+\theta^{-1}}\right)\big(r(x_k, \pi_k^*) + \gamma V^*(x_{k+1})\big) \\
&= \left(1 + \frac{\beta_V - 1}{1+\theta^{-1}}\right) V^*(x_k)
\end{aligned} \tag{5.12}
$$

Assume that the upper bound inequality holds for i, then

$$
\begin{aligned}
&V^{(i+1)}(x_k) \leq Z^{(i+1)}(x_k) \\
&= r(x_k, \nu_k) + \gamma Z^{(i)}(x_{k+1}) \\
&= r(x_k, \nu_k) + \gamma V^{(i)}(x_{k+1}) \\
&\leq r(x_k, \nu_k) + \left(1 + \frac{\beta_V - 1}{(1+\theta^{-1})^i}\right)\gamma V^*(x_{k+1}) \\
&= \left(1 + \frac{(\beta_V - 1)\theta^i}{(\theta+1)^i}\right) r(x_k, \nu_k) \\
&\quad + \left(1 + \frac{\beta_V - 1}{(1+\theta^{-1})^{i-1}} - \frac{(\beta_V - 1)\theta^{i-1}}{(\theta+1)^i}\right)\gamma V^*(x_{k+1}) \\
&= \left(1 + \frac{(\beta_V - 1)\theta^i}{(\theta+1)^i}\right)\big(r(x_k, \nu_k) + \gamma V^*(x_{k+1})\big)
\end{aligned} \tag{5.13}
$$

Since above holds for arbitrary policy ν_k, it also holds for the optimal policy π_k^* as,

$$\begin{aligned} V^{(i+1)}(x_k) &\leq \left(1 + \frac{(\beta_V - 1)\theta^i}{(\theta+1)^i}\right)\big(r(x_k, \pi_k^*) + \gamma V^*(x_{k+1})\big) \\ &= \left(1 + \frac{\beta_V - 1}{(1+\theta^{-1})^i}\right) V^*(x_k) \end{aligned} \tag{5.14}$$

The upper bound inequality of the sequence $\{V^{(i)}\}$ is proved by the mathematical induction. When $i \to \infty$, the lower bound and upper bound both converge to $V^*(x_k)$, thus the uniform convergence of the sequence $\{V^{(i)}\}$ exists and equal to the optimal value function, i.e., $V^{(\infty)}(x_k) = V^*(x_k)$. $\square$

Lemma 5.2. *The sequence of the costate functions $\{\lambda^{(i)}\}$ defined in Eq. (2.49) satisfies the following equality.*

$$\lambda^{(i+1)}(x_k) - \lambda^*(x_k) = \Lambda(x_k)\big(\lambda_{k+1}^{(i)} - \lambda_{k+1}^*\big) \tag{5.15}$$

for the function $\Lambda \in \mathbb{R}^{S\times S}$ defined in Eq. (5.23), where $\lambda_{k+1}^{(i)} = \lambda^{(i)}\big(f(x_k) + g(x_k)\pi_k^{(i)}\big)$ and $\lambda_{k+1}^* = \lambda^*\big(f(x_k) + g(x_k)\pi_k^*\big)$.

Proof 5.3. *By the definition of the costate function update in Eq. (2.49),*

$$\begin{aligned} \lambda^{(i+1)}(x_k) - \lambda^*(x_k) &= \frac{\partial}{\partial x_k}\big(r(x_k, \pi_k^{(i)}) - r(x_k, \pi_k^*)\big) \\ &+ \gamma \frac{\partial}{\partial x_k}\big(f(x_k) + g(x_k)\pi_k^{(i)}\big)^T \lambda_{k+1}^{(i)} \\ &- \gamma \frac{\partial}{\partial x_k}\big(f(x_k) + g(x_k)\pi_k^*\big)^T \lambda_{k+1}^* \end{aligned} \tag{5.16}$$

We now use $\nabla_{x_k}(\cdot)$ to express the $\dfrac{\partial}{\partial x_k}(\cdot)$. Denote $vec(\cdot)$ the vector-

ization operation for which the matrix is converted into a column vector. Then, using the chain rule the above equation is expressed as

$$\begin{aligned}
&\lambda^{(i+1)}(x_k) - \lambda^*(x_k) \\
&= 2\big(\nabla_{x_k}\pi_k^{(i)}\big)^T R\pi_k^{(i)} - 2\big(\nabla_{x_k}\pi_k^*\big)^T R\pi_k^* \\
&+ \gamma\Big(\nabla_{x_k} f + (\pi_k^{(i)T}\otimes I_S)\nabla_{x_k} vec(g) + g\nabla_{x_k}\pi_k^{(i)}\Big)^T \lambda_{k+1}^{(i)} \\
&- \gamma\Big(\nabla_{x_k} f + (\pi_k^{*T}\otimes I_S)\nabla_{x_k} vec(g) + g\nabla_{x_k}\pi_k^*\Big)^T \lambda_{k+1}^*
\end{aligned} \tag{5.17}$$

where I_n denotes the square identity matrix with the dimension of n. Since π_k^* and $\pi_k^{(i)}$ satisfy Eqs. (2.29) and (2.50), respectively, it can be simplified as

$$\begin{aligned}
&\lambda^{(i+1)}(x_k) - \lambda^*(x_k) \\
&= \gamma\big(\nabla_{x_k} f\big)^T\big(\lambda_{k+1}^{(i)} - \lambda_{k+1}^*\big) \\
&+ \gamma\Big(\nabla_{x_k} vec(g)\Big)^T vec\Big(I_S\lambda_{k+1}^{(i)}\pi_k^{(i)T} - I_S\lambda_{k+1}^*\pi_k^{*T}\Big)
\end{aligned} \tag{5.18}$$

with the property of the vec operator, $vec(AXB) = (B^T\otimes A)vec(X)$. The second term of the RHS becomes,

$$\begin{aligned}
&vec\Big(I_S\lambda_{k+1}^{(i)}\pi_k^{(i)T} - I_S\lambda_{k+1}^*\pi_k^{*T}\Big) \\
&= vec\Big(\big(\lambda_{k+1}^{(i)} - \lambda_{k+1}^*\big)\big(\pi_k^{(i)} + \pi_k^*\big)^T\Big) + \Xi \\
&= \Big(\big(\pi_k^{(i)} + \pi_k^*\big)\otimes I_S\Big)\big(\lambda_{k+1}^{(i)} - \lambda_{k+1}^*\big) + \Xi
\end{aligned} \tag{5.19}$$

where $\Xi = vec\big(\lambda^*_{k+1}\pi_k^{(i)T} - \lambda^{(i)}_{k+1}\pi_k^{*T}\big)$. *Then Eq. (5.18) becomes,*

$$\begin{aligned}
&\lambda^{(i+1)}(x_k) - \lambda^*(x_k) \\
&= \gamma\Big(\nabla_{x_k} f + \big((\pi_k^{(i)T} + \pi_k^{*T}) \otimes I_S\big)\nabla_{x_k} vec(g)\Big)^T \\
&\quad \times \big(\lambda^{(i)}_{k+1} - \lambda^*_{k+1}\big) + \gamma\nabla_{x_k} vec(g)^T \Xi
\end{aligned} \tag{5.20}$$

We can modify term Ξ *to:*

$$\begin{aligned}
\Xi &= -\frac{\gamma}{2} vec\Big(\big(\lambda^*_{k+1}\lambda^{(i)T}_{k+1} - \lambda^{(i)}_{k+1}\lambda^{*T}_{k+1}\big) g R^{-1}\Big) \\
&= -\frac{\gamma}{2} vec\Big(\big(\lambda^*_{k+1}(\lambda^{(i)}_{k+1} - \lambda^*_{k+1})^T \\
&\qquad\qquad - (\lambda^{(i)}_{k+1} - \lambda^*_{k+1})\lambda^{*T}_{k+1}\big) g R^{-1}\Big) \\
&= -\frac{\gamma}{2}\Big((gR^{-1})^T \otimes \lambda^*_{k+1} - (\lambda^{*T}_{k+1} g R^{-1})^T \otimes I_S\Big) \\
&\qquad \times vec\big(\lambda^{(i)}_{k+1} - \lambda^*_{k+1}\big)
\end{aligned} \tag{5.21}$$

where the first equality is obtained by Eqs. (2.29) and (2.50) and the third equality comes from the property of $vec(AXB) = (B^T \otimes A)vec(X)$. *Substituting* Ξ *into Eq. (5.20), we obtain*

$$\lambda^{(i+1)}(x_k) - \lambda^*(x_k) = \Lambda(x_k)\big(\lambda^{(i)}_{k+1} - \lambda^*_{k+1}\big) \tag{5.22}$$

with

$$\Lambda = \gamma\left[\nabla_{x_k} f + \left(\pi_k^{(i)T} \otimes I_S + \frac{\gamma}{2} g R^{-1} \otimes \lambda^{*T}_{k+1}\right)\nabla_{x_k} vec(g)\right]^T \tag{5.23}$$

$\square$

The next step is to pose a bound condition for which the function

$\Lambda(x_k)$ becomes a contraction operator.

Theorem 5.2. *The costate sequence* $\{\lambda^{(i)}\}$ *converges to* λ^* *uniformly on the state space* Ω *when* $\Lambda_M = \sup_{x_k \in \Omega} ||\Lambda(x_k)||_2 < 1$. *Moreover, the policy sequence* $\{\pi^{(i)}\}$ *converges to* π^* *uniformly in* Ω.

Proof 5.4. *The convergence of the sequence* $\{\lambda^{(i)}\}$ *to the optimal costate function* λ^* *can be demonstrated by*

$$\begin{aligned} &\left\|\lambda^{(i+1)}(x_k) - \lambda^*(x_k)\right\|_2 \\ &\leq \left\|\Lambda(x_k)\right\|_2 \left\|\lambda^{(i)}(x_{k+1}^{(i)}) - \lambda^*(x_{k+1}^*)\right\|_2 \\ &\leq \Lambda_M^i \left\|\lambda^{(1)}(x_{k+i}^{(1)}) - \lambda^*(x_{k+i}^*)\right\|_2 \end{aligned} \tag{5.24}$$

According to the assumption $\Lambda_M < 1$, *the costate sequence converges to the optimal costate function as* $i \to \infty$. *In the case of the policy sequence, it satisfies the same equality with Eq. (5.15) as*

$$\pi^{(i+1)}(x_k) - \pi^*(x_k) = \Lambda\big(\pi^{(i)}(x_{k+1}^{(i)}) - \pi^*(x_{k+1}^*)\big) \tag{5.25}$$

Hence $\{\pi^{(i)}\}$ *converges to the optimal policy function* π^*. $\square$

Remark 5.1. In the discrete-time linear system $x_{k+1} = Ax_k + Bu_k$, the convergence condition can be simply expressed as $\gamma\|A\|_2 < 1$, which is analogous to the convergence result presented in [154]. The condition can also be interpreted as the stability of linear system: the spectral radius of A is less than one. The second term of $\Lambda(x_k)$ in Eq. (5.23) addresses the effect of non-zero gradient of $g(x_k)$ of nonlinear control-affine system. Denote the second term as Λ_c,

$$\begin{aligned}\Lambda_c &= \left(\pi_k^{(i)T} \otimes I_S + \frac{\gamma}{2} g R^{-1} \otimes \lambda_{k+1}^{*T}\right) \nabla_{x_k} vec(g) \\ &= -\frac{\gamma}{2}\left(\left(\lambda_{k+1}^{(i)T} g R^{-1}\right) \otimes I_S - g R^{-1} \otimes \lambda_{k+1}^{*T}\right) \nabla_{x_k} vec(g)\end{aligned} \tag{5.26}$$

Then the nonlinearity effect can be measured by the spectral norm of Λ_c as

$$\|\Lambda_c\|_2 \le \frac{\gamma}{2} \|\nabla_{x_k} vec(g)\|_2 \|gR^{-1}\|_2 \|\lambda_{k+1}^{(i)} - \lambda_{k+1}^{*}\|_2 \tag{5.27}$$

which becomes zero as $i \to \infty$.

5.3 Function approximation with deep neural networks

5.3.1 Function approximation and gradient descent learning

With the same motivation to Chapter 4, DNNs is used as the function approximator of the value, costate, and policy functions.

Assumption 5.1. *Without loss of generality, three networks (i.e., value, costate, and policy networks) are assumed to have a same number of layers, denoted as $H + 1$, and have a same widths at each layer.*

Denote a function set of value, costate, and policy as $\chi \in \{V, \lambda, \pi\}$. Following the Assumption 5.1, three networks are defined as

$$\begin{aligned}&\overline{\chi}(X) \\ &= W_{\chi,H+1}\sigma_{\chi,H}(W_{\chi,H}\sigma_{\chi,H-1}(\cdots W_{\chi,2}\sigma_{\chi,1}(W_{\chi,1}X)\cdots))\end{aligned} \tag{5.28}$$

where $\overline{\chi}(X)$ denote the value ($\chi = V$), costate ($\chi = \lambda$), and policy ($\chi = \pi$) networks which optimally approximate the corresponding target functions, evaluated at the batch state data X, respectively. The optimal weight matrices of the l^{th} layer of networks are denoted as $W_{\chi,l}$, for which the dimensions are given as: $W_{V,H+1} \in \mathbb{R}^{1\times d_H}$, $W_{\lambda,H+1} \in \mathbb{R}^{S\times d_H}$, $W_{\pi,H+1} \in \mathbb{R}^{A\times d_H}$, ..., $W_{\chi,l} \in \mathbb{R}^{d_l\times d_{l-1}}$, ..., $W_{\chi,1} \in \mathbb{R}^{d_1\times S}$, where d_l represents the width of l^{th} layer. The batch state matrix $X = [x_1, x_2, \dots x_N]$ has dimension $\mathbb{R}^{S\times N}$, where N is the number of data. The element-wise nonlinear activation functions of the l^{th} layer of networks are denoted as $\sigma_{\chi,l}$.

The DNNs approximation takes place in the other dimensional space than the original function, thus there exist reconstruction errors for the networks denoted as ϵ_χ,

$$\chi(X) = \overline{\chi}(X) + \epsilon_\chi \tag{5.29}$$

The optimal DNNs models (i.e., weight matrices) are unknown, and the estimated DNNs networks, denoted as $\hat{\chi}(X)$ are used during the learning process. The corresponding approximate weight matrices of the l^{th} layer at the iteration number i are denoted as $\hat{W}^{(i)}_{\chi,l}$.

The learning procedure aims to make the network sequence (i.e., $\{\hat{\chi}^{(i)}\}$) to be the optimal approximation of the optimal function (i.e., $\overline{\chi}^{*}$). However, since $\overline{\chi}^{*}$ is unknown, it is designed to follow the intermediate values of the iteration path of the GDHP algorithm (i.e., $\{\chi^{(i)}\}$). Hence, the following per-sample residuals which measures the deviations of optimality conditions of GDHP in Eqs. (2.27), (2.28) and (2.29) are defined.

$$\begin{aligned}
e_{V,n}^{(i)} &= r_n + \gamma \hat{V}^{(i)}(x_n^+) - \hat{V}^{(i)}(x_n) \\
e_{\lambda,n}^{(i)} &= \nabla_{x_n} r_n + \gamma \big(\nabla_{x_n} x_n^+\big)^T \hat{\lambda}^{(i)}(x_n^+) - \hat{\lambda}^{(i)}(x_n) \\
e_{\pi,n}^{(i)} &= \hat{\pi}^{(i)}(x_n) + \frac{\gamma}{2} R^{-1} g^T(x_n) \hat{\lambda}^{(i)}(x_n^+)
\end{aligned} \tag{5.30}$$

where (x_n, u_n, r_n, x_n^+) is the n^{th} data ($n \in \{1, \ldots, N\}$) data of a sample batch.

The scalar GDHP residual functions are calculated as, $E_\chi = \frac{1}{2} e_\chi^{(i)T} e_\chi^{(i)}$, where $e_\chi^{(i)} = vec\big([e_{\chi,1}^{(i)}, e_{\chi,2}^{(i)}, \ldots, e_{\chi,N}^{(i)}]\big)$. The gradient descent method for the value network is expressed as,

$$vec(\hat{W}_{\chi,l}^{(i+1)}) = vec(\hat{W}_{\chi,l}^{(i)}) - \eta_\chi \big(\nabla_l e_\chi^{(i)}\big)^T e_\chi^{(i)} \tag{5.31}$$

where η_χ is the learning rate of the networks and $\nabla_l e_\chi^{(i)} = \dfrac{\partial e_\chi^{(i)}}{\partial vec(\hat{W}_{\chi,l}^{(i)})}$.

5.3.2 Forward and backward propagations of DNNs

The training and evaluation of the feed-forward DNNs are performed by forward and backward propagation rule [161]. In the forward propagation step, the vectorized output value of the layer is transferred from its inner layer as

$$\begin{aligned}
vec(p_{\chi,l}) &= vec\big(\hat{W}_{\chi,l}^{(i)} \sigma(p_{\chi,l-1})\big) \\
&= (I_N \otimes \hat{W}_{\chi,l}^{(i)}) \sigma(vec(p_{\chi,l-1})), \quad l \in \{2, \ldots, H+1\} \\
vec(p_{\chi,1}) &= (I_N \otimes \hat{W}_{\chi,1}^{(i)}) vec(X)
\end{aligned} \tag{5.32}$$

where $p_{\chi,l} \in \mathbb{R}^{d_l \times N}$ is the value evaluated at the l^{th} hidden layer. In the backward propagation step, the gradient of DNNs with respect to the l^{th} weights is expressed with that of the outer layer $(l+1)^{\text{th}}$ as

$$\begin{aligned} \delta_{\chi,l} &= \delta_{\chi,l+1}\big(I_N \otimes \hat{W}^{(i)}_{\chi,l+1}\big)\nabla\sigma\big(vec(p_{\chi,l})\big), \\ \delta_{\chi,H+1} &= I_{Nd_{H+1}}, \quad l = \{1,\dots,H\} \end{aligned} \tag{5.33}$$

where $\delta_{\chi,l} \in \mathbb{R}^{Nd_{H+1}\times Nd_l}$ is the accumulated gradient signal evaluated in the l^{th} layer. $\nabla\sigma\big(vec(p_{\chi,l})\big)$ is the diagonal matrix of $\mathbb{R}^{Nd_l\times Nd_l}$ whose n^{th} element being $\nabla\sigma\big([vec(p_{\chi,l})]_n\big)$. The gradient of the NN output with respect to the vectorized weight is computed as,

$$\frac{\partial vec\big(\hat{\chi}^{(i)}(X)\big)}{\partial vec(\hat{W}^{(i)}_{\chi,l})} = \delta^{(i)}_{\chi,l}(X)\big(\sigma(p^{(i)}_{\chi,l-1}(X))^T \otimes I_{d_l}\big) \tag{5.34}$$

According to the residuals defined in Eq. (5.30) and the backward propagation rule, $\nabla_l e^{(i)}_{\chi} \in \mathbb{R}^{Nd_{H+1}\times d_l d_{l-1}}$ in Eq. (5.31) becomes

$$\begin{aligned} \nabla_l e^{(i)}_V &= \gamma\delta^{(i)}_{V,l}(X^+)\big(\sigma(\hat{p}^{(i)}_{V,l-1}(X^+))^T \otimes I_{d_l}\big) \\ &\quad - \delta^{(i)}_{V,l}(X)\big(\sigma(\hat{p}^{(i)}_{V,l-1}(X))^T \otimes I_{d_l}\big) \\ \nabla_l e^{(i)}_\lambda &= \gamma\big(\nabla_X X^+\big)^T\delta^{(i)}_{\lambda,l}(X^+)\big(\sigma(\hat{p}^{(i)}_{\lambda,l-1}(X^+))^T \otimes I_{d_l}\big) \\ &\quad - \delta^{(i)}_{\lambda,l}(X)\big(\sigma(\hat{p}^{(i)}_{\lambda,l-1}(X))^T \otimes I_{d_l}\big) \\ \nabla_l e^{(i)}_\pi &= \delta^{(i)}_{\pi,l}(X)\big(\sigma(\hat{p}^{(i)}_{\pi,l-1}(X))^T \otimes I_{d_l}\big) \\ &\quad + \left(\frac{\gamma}{2}R^{-1}g^T(X)\right)\delta^{(i)}_{\pi,l}(X^+)\big(\sigma(\hat{p}^{(i)}_{\pi,l-1}(X^+))^T \otimes I_{d_l}\big) \end{aligned} \tag{5.35}$$

where $\nabla_X X^+$ and $R^{-1}g^T(X)$ are diagonal block matrices whose n^{th} blocks are $\nabla_{x_n} x_n^+$ and $R^{-1}g^T(x_n)$, respectively.

5.4 Convergence analysis in the deep neural networks space

In this section, the Lyapunov functions of the DNNs weight errors and the closed-loop systems are constructed. The ultimate boundedness is obtained as a result.

5.4.1 Lyapunov analysis of the neural network parameter errors

Recall that the ultimate goal of the algorithm is to find the optimal DNNs weights $W^*_{\chi,l}$. The convergence analysis starts from expressing the residual $e^{(i)}_{\chi,n}$ with respect to $\hat{\chi}^{(i)} - \overline{\chi}^*$, which can be decomposed into three steps as

$$\overline{\chi}^* \underset{\epsilon^*_\chi}{\longrightarrow} \chi^* \underset{\text{GDHP}}{\longrightarrow} \chi^{(i)} \underset{e^{(i)}_\chi}{\longrightarrow} \hat{\chi}^{(i)} \tag{5.36}$$

where each step refers to the reconstruction error which measures the irreducible neural network approximation error (Eq. (5.29)), GDHP iteration error in the value function space (Eqs. (5.6), (5.15), and (5.25)), and GDHP residual in the neural network space (Eq. (5.30)), respectively. Three steps of error decomposition is illustrated in Fig. 5.1.

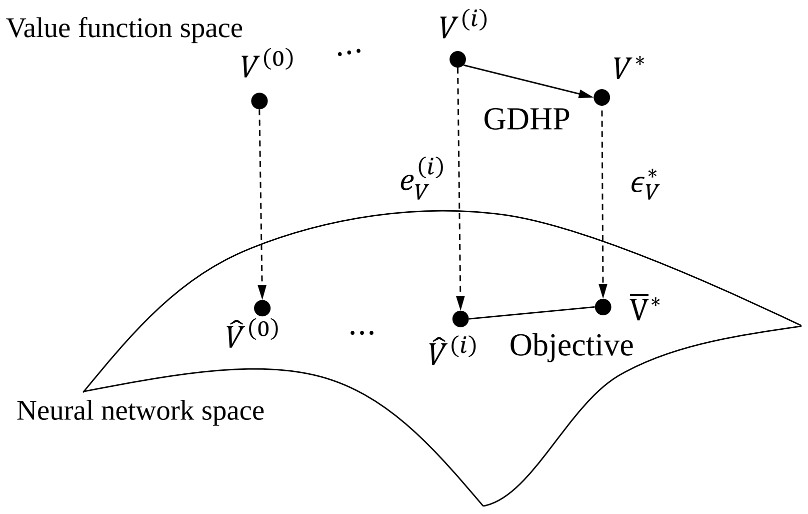


Figure 5.1: Illustration of three-step decomposition in $\hat{\chi}^{(i)} - \overline{\chi}^*$.

Note that this decomposition is not considered in the previous convergence analysis [160, 159, 162]. The designed residual functions make network sequence $\{\hat{\chi}^{(i)}\}$ to follow $\{\overline{\chi}^{(i)}\}$, which is the moving target regression problem. Whereas, previous results assumed that $\{\hat{\chi}^{(i)}\}$ is directly steered toward $\overline{\chi}^{*}$.

The residual $e_{V,n}^{(i)}$ is modified by considering three errors: GDHP value function iteration rule Eq. (2.48), GDHP iteration error Eq. (5.6), and the reconstruction error Eq. (5.29) as

$$\begin{aligned} e_{V,n}^{(i)} &= \gamma \hat{V}^{(i)}(x_n^+) - \gamma V^{(i)}(x_n^+) - \hat{V}^{(i)}(x_n) + V^{(i+1)}(x_n) \\ &= \gamma \hat{V}^{(i)}(x_n^+) - \gamma V^{*}(x_n^+) - \hat{V}^{(i)}(x_n) + V^{*}(x_n) \\ &\quad - \frac{\gamma \alpha_{V,n}^{(i)}}{(1+\theta^{-1})^{i}} V^{*}(x_n^+) + \frac{\alpha_{V,n}^{(i+1)}}{(1+\theta^{-1})^{i+1}} V^{*}(x_n) \\ &= \gamma \hat{V}^{(i)}(x_n^+) - \gamma \overline{V}^{*}(x_n^+) - \hat{V}^{(i)}(x_n) + \overline{V}^{*}(x_n) + M_{V,n}^{(i)} \end{aligned} \tag{5.37}$$

where $\alpha_{V,n}^{(i)}$ is the constant satisfying $\alpha_V - 1 \leq \alpha_{V,n}^{(i)} \leq \beta_V - 1$ according to Eq. (5.6). The overall error term $M_{V,n}^{(i)}$ is defined as

$$\begin{aligned} M_{V,n}^{(i)} = &\frac{\alpha_{V,n}^{(i+1)} V^{*}(x_n) - \gamma \alpha_{V,n}^{(i)} (1+\theta^{-1}) V^{*}(x_n^+)}{(1+\theta^{-1})^{i+1}} \\ &- \gamma \epsilon_{V,n}^{+*} + \epsilon_{V,n}^{*} \end{aligned} \tag{5.38}$$

The GDHP residuals for costate $e_{\lambda,n}^{(i)}$ and policy $e_{\pi,n}^{(i)}$ are evaluated similarly

$$\begin{aligned}
e_{\lambda,n}^{(i)} &= \gamma\big(\nabla_{x_n} x_n^+\big)^T \left(\hat{\lambda}^{(i)}(x_n^+) - \overline{\lambda}^*(x_n^+)\right) \\
&\quad - \hat{\lambda}^{(i)}(x_n) + \overline{\lambda}^*(x_n) + M_{\lambda,n}^{(i)} \\
e_{\pi,n}^{(i)} &= \hat{\pi}^{(i)}(x_n) - \overline{\pi}^*(x_n) \\
&\quad + \frac{\gamma}{2} R^{-1} g^T \left(\hat{\lambda}^{(i)}(x_n^+) - \overline{\lambda}^*(x_n^+)\right) + M_{\pi,n}^{(i)}
\end{aligned} \tag{5.39}$$

According to Theorem 5.2, $\alpha_{\lambda,n}^{(i)} \in \mathbb{R}^{S\times 1}$ and $\alpha_{\pi,n}^{(i)} \in \mathbb{R}^{A,1}$ satisfy $\lambda^{(i)}(x_n) - \lambda^*(x_n) = \|\Lambda\|_2^i \alpha_{\lambda,n}^{(i)}(x_n)$ and $\pi^{(i)}(x_n) - \pi^*(x_n) = \|\Lambda\|_2^i \alpha_{\pi,n}^{(i)}(x_n)$, respectively. The overall error terms $M_{\lambda,n}^{(i)}$ and $M_{\pi,n}^{(i)}$ are defined as

$$\begin{aligned}
M_{\lambda,n}^{(i)} &= -\gamma\big(\nabla_{x_n} x_n^+\big)^T \|\Lambda\|_2^i \alpha_{\lambda,n}^{(i)}(x_n^+) + \|\Lambda\|_2^{i+1} \alpha_{\lambda,n}^{(i+1)}(x_n) \\
&\quad - \gamma\big(\nabla_{x_n} x_n^+\big)^T \epsilon_{\lambda,n}^{+*} + \epsilon_{\lambda,n}^* \\
M_{\pi,n}^{(i)} &= -\|\Lambda\|_2^i \alpha_{\pi,n}^{(i)}(x_n) + \frac{\gamma}{2} R^{-1} g^T(x_n) \|\Lambda\|_2^i \alpha_{\lambda,n}^{(i)}(x_n^+) \\
&\quad - \epsilon_{\pi,n}^* + \frac{\gamma}{2} R^{-1} g^T \epsilon_{\lambda,n}^*
\end{aligned} \tag{5.40}$$

The vectorized terms $\hat{\chi}^{(i)}(X^+) - \overline{\chi}^*(X^+)$ in Eqs. (5.37) and (5.39) are written with respect to the DNNs weight errors $\tilde{W}_{\chi,l}^{(i)}$. Since Eq. (5.32) can be rewritten as $vec(p_{\chi,l}) = \big(\sigma(p_{\chi,l-1})^T \otimes I_{d_l}\big) vec(\hat{W}_{\chi,l}^{(i)})$, Eq. (5.41) holds.

$$\begin{aligned}
& vec\big(\hat{\chi}^{(i)}(X^+) - \overline{\chi}^*(X^+)\big) \\
&= \big(\sigma(\hat{p}_{\chi,H}^{(i)})^T \otimes I_{d_{H+1}}\big) vec(\hat{W}_{\chi,H+1}^{(i)}) \\
&- \big(\sigma(\hat{p}_{\chi,H}^{(i)})^T \otimes I_{d_{H+1}}\big) vec(W_{\chi,H+1}^*) \\
&+ \big(\sigma(\hat{p}_{\chi,H}^{(i)})^T \otimes I_{d_{H+1}}\big) vec(W_{\chi,H+1}^*) \\
&- \big(\sigma(p_{\chi,H}^*)^T \otimes I_{d_{H+1}}\big) vec(W_{\chi,H+1}^*)
\end{aligned} \tag{5.41}$$

Since the activation function σ is the element-wise operation, it is commutative with the vec operator [163]. Also, the mean value theorem for σ can be implemented element-wise to obtain the diagonal matrix $\xi^+_{\chi,H}$, whose n^{th} element is $\nabla\sigma([\xi^+_{\chi,H}]_n)$, where $[\xi^+_{\chi,H}]_n \in [vec(p^*_{\chi,H})_n, vec(\hat{p}^{(i)}_{\chi,H})_n]$.

$$\begin{aligned}
&vec\big(\hat{\chi}^{(i)}(X^+) - \overline{\chi}^*(X^+)\big) \\
&= \big(\sigma(\hat{p}^{(i)}_{\chi,H})^T \otimes I_{d_{H+1}}\big)vec(\tilde{W}^{(i)}_{\chi,H+1}) \\
&\qquad + (I_N \otimes W^*_{\chi,H+1})\Bigg(\sigma\big(vec(\hat{p}^{(i)}_{\chi,H})\big) - \sigma\big(vec(p^*_{\chi,H})\big)\Bigg) \\
&= \big(\sigma(\hat{p}^{(i)}_{\chi,H})^T \otimes I_{d_{H+1}}\big)vec(\tilde{W}^{(i)}_{\chi,H+1}) \\
&\qquad + (I_N \otimes W^*_{\chi,H+1})\xi^+_{\chi,H}vec(\hat{p}^{(i)}_{\chi,H} - p^*_{\chi,H})
\end{aligned} \tag{5.42}$$

The procedure in Eqs. (5.41) and (5.42) is performed recursively from the outermost layer throughout the inner layers to yield

$$\begin{aligned}
&vec\big(\hat{\chi}^{(i)}(X^+) - \overline{\chi}^*(X^+)\big) \\
&= \sum_{m=1}^{H+1} K^+_{\chi,m}\big(\sigma(\hat{p}^{(i)}_{\chi,m-1}(X^+))^T \otimes I_{d_m}\big)vec\big(\tilde{W}^{(i)}_{\chi,m}\big)
\end{aligned} \tag{5.43}$$

The coefficients $K^+_{\chi,m} \in \mathbb{R}^{Nd_{H+1}\times Nd_m}$ are defined recursively with respect to $K^+_{\chi,m+1}$ and the $\xi^+_{\chi,m}$ are acquired from the mean value theorem.

$$\begin{aligned}
&\sigma\big(vec(\hat{p}^{(i)}_{\chi,m})\big) - \sigma\big(vec(p^*_{\chi,m})\big) = \xi^+_{\chi,m}vec\big(\hat{p}^{(i)}_{\chi,m} - p^*_{\chi,m}\big) \\
&K^+_{\chi,m} = K^+_{\chi,m+1}(I_N \otimes W^*_{\chi,m+1})\xi^+_{\chi,m} \\
&K^+_{\chi,H+1} = I_{Nd_{H+1}}
\end{aligned} \tag{5.44}$$

Finally, the Lyapunov candidate function of the weight errors $\tilde{W}_{\chi}^{(i)} = \hat{W}_{\chi}^{(i)} - W_{\chi}^{*}$ can be constructed, for the later purpose to check the convergence of the neural networks. Lemma 5.3 provides an explicit expression of the total difference.

Lemma 5.3. *Define the Lyapunov candidate functions of weight errors as* $\mathcal{L}_{\chi} = \sum_{l=1}^{H+1} \left\| vec(\tilde{W}_{\chi,l}^{(i)}) \right\|^2$. *Then* $\Delta\mathcal{L}_{\chi}$ *are expressed as quadratic forms with respect to* $\tilde{W}_{\chi}^{(i)} = \left[vec(\tilde{W}_{\chi,1}^{(i)})^T, \ldots, vec(\tilde{W}_{\chi,H+1}^{(i)})^T \right]^T$ *and* $M_{\chi}^{(i)} = vec\big([M_{\chi,1}^{(i)}, \ldots, M_{\chi,N}^{(i)}] \big)$.

Proof 5.5. *The weight errors dynamics are expressed as,*

$$vec(\tilde{W}_{\chi,l}^{(i+1)}) = vec(\tilde{W}_{\chi,l}^{(i)}) - \eta_{\chi} \big(\nabla_l e_{\chi}^{(i)} \big)^T e_{\chi}^{(i)} \tag{5.45}$$

The total differences can be evaluated by substituting the weight error dynamics as,

$$\begin{aligned} \Delta\mathcal{L}_{\chi} &= \sum_{l=1}^{H+1} \left(\left\| vec(\tilde{W}_{\chi,l}^{(i+1)}) \right\|^2 - \left\| vec(\tilde{W}_{\chi,l}^{(i)}) \right\|^2 \right) \\ &= \sum_{l=1}^{H+1} \Bigg(-\eta_{\chi} vec(\tilde{W}_{\chi,l}^{(i)})^T \big(\nabla_l e_{\chi}^{(i)} \big)^T e_{\chi}^{(i)} \\ &\quad - \eta_{\chi} e_{\chi}^{(i)T} \nabla_l e_{\chi}^{(i)} vec(\tilde{W}_{\chi,l}^{(i)}) + \eta_V^2 \left\| \big(\nabla_l e_{\chi}^{(i)} \big)^T e_{\chi}^{(i)} \right\|^2 \Bigg) \end{aligned} \tag{5.46}$$

The GDHP residuals $e_V^{(i)}$ *in Eqs. (5.37) and (5.39) are expressed by using Eq. (5.43) as*

$$e_V^{(i)} = \sum_{m=1}^{H+1} K_{V,m} vec\big(\tilde{W}_{V,m}^{(i)} \big) + M_V^{(i)} \tag{5.47}$$

The coefficients $K_{V,m} \in \mathbb{R}^{Nd_{H+1} \times d_m d_{m-1}}$ *are given as*

$$\begin{aligned} K_{V,m} = \gamma K_{V,m}^{+}\big(\sigma(\hat{p}_{V,m-1}^{(i)}(X^{+}))^{T} \otimes I_{d_m}\big) \\ - K_{V,m}^{-}\big(\sigma(\hat{p}_{V,m-1}^{(i)}(X))^{T} \otimes I_{d_m}\big) \end{aligned} \tag{5.48}$$

where the coefficients $K_{V,m}^{-}$ *satisfy the equality which is similar to the definition of* $K_{V,m}^{+}$ *in Eqs. (5.43) and (5.44).*

$$\begin{aligned} & vec\big(\hat{V}^{(i)}(X) - \overline{V}^{*}(X)\big) \\ & = \sum_{m=1}^{H+1} K_{V,m}^{-}\big(\sigma(\hat{p}_{m-1}^{(i)}(X))^{T} \otimes I_{d_m}\big) vec\big(\tilde{W}_{V,m}^{(i)}\big) \end{aligned} \tag{5.49}$$

The GDHP residuals for costate and policy are defined similarly as

$$\begin{aligned} e_{\lambda}^{(i)} &= \sum_{m=1}^{H+1} K_{\lambda,m} vec\big(\tilde{W}_{\lambda,m}^{(i)}\big) + M_{\lambda}^{(i)} \\ e_{\pi}^{(i)} &= \sum_{m=1}^{H+1}\bigg(K_{\pi,m} vec\big(\tilde{W}_{\pi,m}^{(i)}\big) + K_{\pi\lambda,m} vec\big(\tilde{W}_{\lambda,m}^{(i)}\big)\bigg) + M_{\pi}^{(i)} \end{aligned} \tag{5.50}$$

The coefficients $K_{\lambda,m}, K_{\pi,m}$ *and* $K_{\pi\lambda,m} \in \mathbb{R}^{Nd_{H+1} \times d_m d_{m-1}}$ *are defined as*

$$\begin{aligned} K_{\lambda,m} &= \gamma\big(\nabla_X X^{+}\big)^{T} K_{\lambda,m}^{+}\big(\sigma(\hat{p}_{\lambda,m-1}^{(i)}(X^{+}))^{T} \otimes I_{d_m}\big) \\ &\quad - K_{\lambda,m}^{-}\big(\sigma(\hat{p}_{\lambda,m-1}^{(i)}(X))^{T} \otimes I_{d_m}\big) \\ K_{\pi,m} &= K_{\pi,m}^{-}\big(\sigma(\hat{p}_{\pi,m-1}^{(i)}(X))^{T} \otimes I_{d_m}\big) \\ K_{\pi\lambda,m} &= \bigg(\frac{\gamma}{2} R^{-1} g^{T}(X)\bigg) K_{\lambda,m}^{+}\big(\sigma(\hat{p}_{\lambda,m-1}^{(i)}(X^{+}))^{T} \otimes I_{d_m}\big) \end{aligned} \tag{5.51}$$

with the coefficients $K_{\lambda,m}^{+}, K_{\lambda,m}^{-}$ *and* $K_{\pi,m}^{-} \in \mathbb{R}^{Nd_{H+1} \times d_m d_{m-1}}$ *are* *defined similar to Eqs. (5.43) and (5.44).*

Finally, $\Delta\mathcal{L}_V$ is expressed with the weight error $\tilde{W}_{V,l}^{(i)}$ by substituting Eq. (5.47) to Eq. (5.46) as,

$$\begin{aligned}
&\Delta\mathcal{L}_V \\
&= -\eta_V \sum_{l=1}^{H+1}\sum_{m=1}^{H+1}\Bigg(vec(\tilde{W}_{V,l}^{(i)})^T \nabla_l e_V^{(i)T} K_{V,m} vec(\tilde{W}_{V,m}^{(i)}) \\
&+ vec(\tilde{W}_{V,m}^{(i)})^T K_{V,m}^T \nabla_l e_V^{(i)} vec(\tilde{W}_{V,l}^{(i)}) \Bigg) - \eta_V \times \\
&\sum_{l=1}^{H+1}\Bigg(vec(\tilde{W}_{V,l}^{(i)})^T \nabla_l e_V^{(i)T} M_V^{(i)} + M_V^{(i)T} \nabla_l e_V^{(i)} vec(\tilde{W}_{V,l}^{(i)}) \Bigg) \\
&+ \eta_V^2 \sum_{l=1}^{H+1} \left\| \nabla_l e_V^{(i)T} \left(\sum_{m=1}^{H+1} K_{V,m} vec(\tilde{W}_{V,m}^{(i)}) + M_V^{(i)} \right) \right\|^2 \Bigg]
\end{aligned} \tag{5.52}$$

Then $\Delta\mathcal{L}_V$ can be written in a compact quadratic form as,

$$\begin{aligned}
&\Delta\mathcal{L}_V \\
&= -\tilde{W}_V^{(i)T} A_V \tilde{W}_V^{(i)} + \tilde{W}_V^{(i)T} B_V M_V^{(i)} + M_V^{(i)T} B_V^T \tilde{W}_V^{(i)} \\
&+ M_V^{(i)T} C_V M_V^{(i)}
\end{aligned} \tag{5.53}$$

The coefficient matrices are defined as,

$$\begin{aligned}
[A_V]_{lm} &= \eta_V \left(\nabla_l e_V^{(i)T} K_{V,m} + K_{V,l}^T \nabla_m e_V^{(i)} \right) \\
&\quad - \eta_V^2 K_{V,l}^T C_V K_{V,m} \\
[B_V]_l &= -\eta_V \nabla_l e_V^{(i)T} + \eta_V^2 K_{V,l}^T C_V \\
C_V &= \sum_{l=1}^{H+1} \nabla_l e_V^{(i)} \nabla_l e_V^{(i)T}, \quad l, m \in \{1, \dots H+1\}
\end{aligned} \tag{5.54}$$

where the $[\cdot]_{lm}$ *refers to the block matrix of location* (l, m). *Similarly,* $\Delta\mathcal{L}_\lambda$ *is computed as,*

$$\begin{aligned}
&\Delta\mathcal{L}_\lambda \\
&= -\tilde{W}_\lambda^{(i)T} A_\lambda \tilde{W}_\lambda^{(i)} + \tilde{W}_\lambda^{(i)T} B_\lambda M_\lambda^{(i)} + M_\lambda^{(i)T} B_\lambda^T \tilde{W}_\lambda^{(i)} \\
&+ M_\lambda^{(i)T} C_\lambda M_\lambda^{(i)}
\end{aligned} \tag{5.55}$$

with the coefficient matrices,

$$\begin{aligned}
[A_\lambda]_{lm} &= \eta_\lambda \bigg(\nabla_l e_\lambda^{(i)T} K_{\lambda,m} + K_{\lambda,l}^T \nabla_m e_\lambda^{(i)} \bigg) \\
&\quad - \eta_\lambda^2 K_{\lambda,l}^T C_\lambda K_{\lambda,m} \\
[B_\lambda]_l &= -\eta_\lambda \nabla_l e_\lambda^{(i)T} + \eta_\lambda^2 K_{\lambda,l}^T C_\lambda \\
C_\lambda &= \sum_{l=1}^{H+1} \nabla_l e_\lambda^{(i)} \nabla_l e_\lambda^{(i)T}, \qquad l, m \in \{1, \ldots, H+1\}
\end{aligned} \tag{5.56}$$

Finally, the quadratic form of $\Delta\mathcal{L}_\pi$ *is written as*

$$\begin{aligned}
&\Delta\mathcal{L}_\pi \\
&= -\tilde{W}_\pi^{(i)T} A_{\pi,1} \tilde{W}_\pi^{(i)} - \tilde{W}_\pi^{(i)T} A_{\pi,2} \tilde{W}_\lambda^{(i)} \\
&- \tilde{W}_\lambda^{(i)T} A_{\pi,2}^T \tilde{W}_\pi^{(i)} - \tilde{W}_\lambda^{(i)T} A_{\pi,3} \tilde{W}_\lambda^{(i)} \\
&+ \tilde{W}_\pi^{(i)T} B_{\pi,1} M_\pi^{(i)} + M_\pi^{(i)T} B_{\pi,1}^T \tilde{W}_\pi^{(i)} \\
&+ \tilde{W}_\lambda^{(i)T} B_{\pi,2} M_\pi^{(i)} + M_\pi^{(i)T} B_{\pi,2}^T \tilde{W}_\lambda^{(i)} \\
&+ M_\pi^{(i)T} C_\pi M_\pi^{(i)}
\end{aligned} \tag{5.57}$$

where the coefficient matrices are defined as,

$$
\begin{aligned}
[A_{\pi,1}]_{lm} &= \eta_\pi \left(\nabla_l e_\pi^{(i)T} K_{\pi,m} + K_{\pi,l}^T \nabla_m e_\pi^{(i)} \right) \\
&\quad - \eta_\pi^2 K_{\pi,l}^T C_\pi K_{\pi,m} \\
[A_{\pi,2}]_{lm} &= \eta_\pi \nabla_l e_\pi^{(i)T} K_{\pi\lambda,m} - \eta_\pi^2 K_{\pi,l}^T C_\pi K_{\pi\lambda,m} \\
[A_{\pi,3}]_{lm} &= \eta_\pi^2 K_{\pi\lambda,l}^T C_\pi K_{\pi\lambda,m} \\
[B_{\pi,1}]_l &= -\eta_\pi \nabla_l e_\pi^{(i)T} + \eta_\pi^2 K_{\pi,l}^T C_\pi \\
[B_{\pi,2}]_l &= \eta_\pi^2 K_{\pi\lambda,l}^T C_\pi \\
C_\pi &= \sum_{l=1}^{H+1} \nabla_l e_\pi^{(i)} \nabla_l e_\pi^{(i)T}, \qquad l, m \in \{1, \dots, H+1\}
\end{aligned}
\tag{5.58}
$$

5.4.2 Lyapunov analysis of the closed-loop stability

The state Lyapunov function is defined as the square norm of the state, $\mathcal{L}_x = \|X\|^2$.

Assumption 5.2. *Under the closed-loop dynamics with the optimal policy $\pi^*(x_n)$, there exists a constant $0 < \alpha_{x_n} < 0.5$ such that $\left\| f(x_n) + g(x_n)\pi^*(x_n) \right\|^2 \le \alpha_{x_n} \|x_n\|^2$* [159].

Lemma 5.4. *When the closed-loop dynamics is subject to Assumption 5.2, the total difference of the Lyapunov candidate function $\Delta\mathcal{L}_x$ is upper bounded by the quadratic function given in Eq. (5.63).*

Proof 5.6. *Total difference $\Delta\mathcal{L}_x$ is written as*

$$\begin{aligned}\Delta\mathcal{L}_x &= \|X^+\|^2 - \|X\|^2 \\ &= \sum_{n=1}^{N}\bigg(\big\|f(x_n) + g(x_n)\pi^*(x_n) - g(x_n)\pi^*(x_n) \\ &\qquad + g(x_n)\hat{\pi}^{(i)}(x_n)\big\|^2 - \big\|x_n\big\|^2\bigg) \\ &\leq \sum_{n=1}^{N}\bigg(-(1-2\alpha_{x_n})\big\|x_n\big\|^2 + \big\|g(x_n)(\hat{\pi}^{(i)}(x_n) - \pi^*(x_n))\big\|^2\bigg)\end{aligned} \tag{5.59}$$

by using Assumption 5.2. The second term of $\Delta\mathcal{L}_x$ is modified by following the similar procedure of the value, costate, and policy residual values as,

$$\begin{aligned} e_{x,n}^{(i)} &= g(x_n)\big(\hat{\pi}^{(i)}(x_n) - \pi^*(x_n)\big) \\ &= g(x_n)\big(\hat{\pi}^{(i)}(x_n) - \overline{\pi}^*(x_n)\big) + M_{x,n}^{(i)} \\ M_{x,n}^{(i)} &= -g(x_n)\epsilon_{\pi,n}^{*}\end{aligned} \tag{5.60}$$

The residual value of the state $e_x^{(i)} := vec\big([e_{x,1}^{(i)}, e_{x,2}^{(i)}, \dots, e_{x,N}^{(i)}]\big)$, is given as,

$$e_x^{(i)} = \sum_{m=1}^{H+1} K_{x,m} vec\big(\tilde{W}_{\pi,m}^{(i)}\big) + M_x^{(i)} \tag{5.61}$$

and $K_{x,m}$ is defined as,

$$K_{x,m} = g(X)K_{\pi,m}^{-}\big(\sigma(\hat{p}_{\pi,m-1}^{(i)}(X))^T \otimes I_{d_m}\big) \tag{5.62}$$

where the coefficient $K^{-}_{\pi,m}$ is defined in Eq. (5.51). The upper bound of the total difference is given in the quadratic form as,

$$\begin{aligned}\Delta\mathcal{L}_x \leq &-X^T A_{x,1} X + \tilde{W}_{\pi}^{(i)T} A_{x,2} \tilde{W}_{\pi}^{(i)} \\ &+ \tilde{W}_{\pi}^{(i)T} B_x M_x^{(i)} + M_x^{(i)T} B_x^T \tilde{W}_{\pi}^{(i)} + M_x^{(i)T} C_x M_x^{(i)}\end{aligned} \tag{5.63}$$

with the following coefficient matrices

$$\begin{aligned}[A_{x,1}]_{nn} &= 1 - 2\alpha_{x_n}, \quad n \in \{1, \dots, N\} \\ [A_{x,2}]_{lm} &= K_{x,l}^T K_{x,m} \\ [B_x]_l &= K_{x,l}^T \\ C_x &= I_{SN}, \qquad l, m \in \{1, \dots, H+1\}\end{aligned} \tag{5.64}$$

$\square$

5.4.3 Overall Lyapunov function

The overall Lyapunov candidate function $\mathcal{L}$ is defined as the summation of the value, costate, policy, and the state Lyapunov candidate functions. The quadratic form of $\Delta\mathcal{L}$ is obtained following Lemmas 5.3 and 5.4 as

$$\begin{aligned}\Delta\mathcal{L} &= \Delta\mathcal{L}_V + \Delta\mathcal{L}_\lambda + \Delta\mathcal{L}_\pi + \Delta\mathcal{L}_x \\ &\leq -W^T \mathbf{A} W + W^T \mathbf{B} M^{(i)} + M^{(i)T} \mathbf{B}^T W + M^{(i)T} \mathbf{C} M^{(i)}\end{aligned} \tag{5.65}$$

where $W = \left[\tilde{W}_V^{(i)T}, \tilde{W}_\lambda^{(i)T}, \tilde{W}_\pi^{(i)T}, X^T\right]^T$ and
$M^{(i)} = \left[M_V^{(i)T}, M_\lambda^{(i)T}, M_\pi^{(i)T}, M_x^{(i)T}\right]^T$ with

$$\mathbf{A} = \begin{bmatrix} A_V & 0 & 0 & 0 \\ 0 & A_\lambda + A_{\pi,3} & A_{\pi,2}^T & 0 \\ 0 & A_{\pi,2} & A_{\pi,1} - A_{x,2} & 0 \\ 0 & 0 & 0 & A_{x,1} \end{bmatrix} \tag{5.66}$$

$$\mathbf{B} = \begin{bmatrix} B_V & 0 & 0 & 0 \\ 0 & B_\lambda & B_{\pi,2} & 0 \\ 0 & 0 & B_{\pi,1} & B_x \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} C_V & 0 & 0 & 0 \\ 0 & C_\lambda & 0 & 0 \\ 0 & 0 & C_\pi & 0 \\ 0 & 0 & 0 & C_x \end{bmatrix} \tag{5.67}$$

The elements of A, B and C are given in Eqs. (5.54), (5.56), (5.58), and (5.64). We provide preliminary results in Lemmas 5.5 and 5.6 and Assumption 5.3 to assure the negative semi-definiteness of Eq. (5.65).

Lemma 5.5. *Let square matrices a_χ with $\chi \in \{V, \lambda, \pi\}$ whose block matrices are given as*

$$[a_\chi]_{lm} = \nabla_l e_\chi^{(i)T} K_{\chi,m}, \quad l, m \in \{1, \dots H+1\} \tag{5.68}$$

where $\nabla_l e_\chi^{(i)T}$ and $K_{\chi,m}$ are given in Eqs. (5.35) and (5.48), respectively. Then $a_\chi \succeq 0$ when $\tilde{W}_{\chi,l}^{(i)} = 0$ for all l, where $X \succeq 0$ means X is PSD.

Proof 5.7. *$\nabla_l e_\chi^{(i)}$ and $K_{\chi,l}$ become identical except for the backward propagation signals from the outer level ($\delta_{\chi,l}^{(i)}$ and $K_{\chi,l}^+$, respectively). The backward propagation signals are expressed with respect* *to $\nabla\sigma(p_{\chi,l})$ and $\hat{W}^{(i)}_{\chi,l+1}$ in the case of $\delta^{(i)}_{\chi,l}$, whereas $K^{+}_{\chi,l}$ is written with $\xi^{+}_{\chi,l}$ and $W^{*}_{\chi,l+1}$. When $\tilde{W}^{(i)}_{\chi,l} = 0$, two signals become identical. Hence $\nabla_l e^{(i)}_{\chi} = K_{\chi,l}$. Since $a_{\chi} = \nabla e^{(i)T}_{\chi} K_{\chi}$, where $\nabla e^{(i)}_{\chi} = [\nabla_1 e^{(i)}_{\chi}, \ldots \nabla_{H+1} e^{(i)}_{\chi}]$ and $K_{\chi} = [K_{\chi,1}, \ldots K_{\chi,H+1}]$, it is PSD.* $\square$

Remark 5.2. The proof of Lemma 5.5 guarantees the matrix a_{χ} to be PSD only if the weights are optimal. Since the positive semi-definiteness is a sufficient condition for the stability of the GDHP algorithm, the accurate initial estimation of the weights is a critical stage throughout the learning process. This motivates the pre-training which assists the initial learning stage for the deep RL problems [37].

Assumption 5.3. *System dynamics is assumed to be uniformly upper bounded by $\sup_{x_k \in \Omega} \|g(x_k)\|_2 \leq 1$. The optimal policy network weight is assumed to be $\|W^{*}_{\pi}\|_2 \leq 1$. In addition, the activation function satisfies the linear growth condition $\|\sigma(p)\|_2 \leq L_1 \|p\|_2$ and Lipschitz condition $\|\sigma(p_1) - \sigma(p_2)\|_2 \leq L_2 \|p_1 - p_2\|_2$ for all $p, p_1, p_2 \in \mathbb{R}$ with the constants $0 \leq L_1, L_2 \leq 1$.*

Remark 5.3. Uniform upper bound condition for dynamics g can be obtained by controlling the sample time Δt arbitrary small. When the state space dynamics Eq. (2.25) is generated by the discretization of the continuous time counterpart (i.e., $\dot{x} = \bar{f}(x) + \bar{g}(x)u$), $g = \bar{g}\Delta t$ holds. Policy network weight condition can be achieved by the normalization of state-space dynamics. Moreover, the linear growth and Lipschitz conditions are hold when the rectified linear unit (ReLU) (i.e., $f(x) = \max(x, 0)$) activation function is used with $L_1, L_2 = 1$.

Lemma 5.6. *Under Assumption 5.2, the condition $\tilde{W}^{(i)}_{\chi,l} = 0$, $\chi \in \{V, \lambda, \pi\}, \forall l$ in Lemma 5.5 and Assumption 5.3, then $\mathbf{A}$ defined in*

Eq. (5.66) becomes PSD when the following conditions for the learning rates are satisfied.

$$\begin{aligned} 0 &\le \eta_V \le 2\|a_V\|_2^{-1} \\ 0 &\le \eta_\lambda \le 2\|a_\lambda\|_2^{-1} \\ \left(1-\sqrt{\frac{1-||A_{x,2}||_2}{2}}\right)\|a_\pi\|_2^{-1} & \\ \le \eta_\pi &\le \left(1+\sqrt{\frac{1-||A_{x,2}||_2}{2}}\right)\|a_\pi\|_2^{-1} \end{aligned} \tag{5.69}$$

Proof 5.8. *We proceed the proof by showing the definiteness of three blocks of* $\boldsymbol{A}$*:* A_V*,* $\tilde{\boldsymbol{A}}$*, and* $A_{x,1}$*, where* $\tilde{\boldsymbol{A}}$ *is defined as*

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} A_\lambda + A_{\pi,3} & A_{\pi,2}^T \\ A_{\pi,2} & A_{\pi,1} - A_{x,2} \end{bmatrix} \tag{5.70}$$

In addition to the definitions of a_χ *in Eq. (5.68), let* $a_{\pi\lambda}$ *whose block matrices are defined as* $[a_{\pi\lambda}]_{lm}$ *be* $\nabla_l e_\pi^{(i)T} K_{\pi\lambda,m}$*. Then the following relationships hold for matrices* A_V*,* A_λ*,* $A_{\pi,1}$*,* $A_{\pi,2}$*, and* $A_{\pi,3}$ *as*

$$\begin{aligned} A_V &= \eta_V(a_V + a_V^T) - \eta_V^2 a_V^T a_V \\ A_\lambda &= \eta_\lambda(a_\lambda + a_\lambda^T) - \eta_\lambda^2 a_\lambda^T a_\lambda \\ A_{\pi,1} &= \eta_\pi(a_\pi + a_\pi^T) - \eta_\pi^2 a_\pi^T a_\pi \\ A_{\pi,2} &= \eta_\pi a_{\pi\lambda} - \eta_\pi^2 a_\pi^T a_{\pi\lambda} \\ A_{\pi,3} &= \eta_\pi^2 a_{\pi\lambda}^T a_{\pi\lambda} \end{aligned} \tag{5.71}$$

According to Lemma 5.5, a_V *is PSD when* $\tilde{W}_{V,l}^{(i)} = 0$ *for all* l*. We choose* $0 \le \eta_V \le 2\|a_V\|_2^{-1}$ *such that* $A_V \succeq 0$*.*

In the case of $\tilde{\boldsymbol{A}}$*, it is enough to find the condition that* $(A_{\pi,1} -$

$A_{x,2}) - A_{\pi,2}(A_\lambda + A_{\pi,3})^{-1}A_{\pi,2}^T \succeq 0$ by using Schur complement [164]. Similar to A_V, we can make $A_\lambda \succeq 0$ if $0 \le \eta_\lambda \le 2\|a_\lambda\|_2^{-1}$, and

$$\begin{aligned}
&(A_{\pi,1} - A_{x,2}) - A_{\pi,2}(A_\lambda + A_{\pi,3})^{-1}A_{\pi,2}^T \\
&\succeq (A_{\pi,1} - A_{x,2}) - A_{\pi,2}A_{\pi,3}^{-1}A_{\pi,2}^T \\
&= (A_{\pi,1} - A_{x,2}) - (I - \eta_\pi a_\pi^T)(I - \eta_\pi a_\pi^T)^T \\
&= 2\eta_\pi(a_\pi + a_\pi^T) - 2\eta_\pi^2 a_\pi^T a_\pi - A_{x,2} - I
\end{aligned} \tag{5.72}$$

According to the Assumption 5.3 and the definition of $K_{x,l}$ in Eq. (5.62),

$$\|K_{x,l}\|_2 \le \|g(X)\|_2 \|K_{\pi,l}^-\|_2 \|\sigma\big(\hat{p}_{\pi,l-1}^{(i)}\big)\|_2 \le 1 \tag{5.73}$$

Thus, $A_{x,2} = K_x^T K_x \preceq I$, and $\tilde{\boldsymbol{A}} \succeq 0$ under which η_π is given as

$$\begin{aligned}
&\left(1 - \sqrt{\frac{1 - ||A_{x,2}||_2}{2}}\right)\|a_\pi\|_2^{-1} \\
&\qquad \le \eta_\pi \le \left(1 + \sqrt{\frac{1 - ||A_{x,2}||_2}{2}}\right)\|a_\pi\|_2^{-1}
\end{aligned} \tag{5.74}$$

The final block always satisfies $A_{x,1} \succeq 0$ due to the Assumption 5.2 (i.e., $0 < \alpha_{x_n} < 0.5$), which concludes the proof. □

Theorem 5.3. *Under Assumptions 5.1 - 5.3 and the learning rates η_V, η_λ, and η_π determined by Lemma 5.6, the value, the costate, and the policy networks weights and the state are uniformly ultimately bounded (UUB). Moreover, the ultimate bound converges as $i \to \infty$.*

Proof 5.9. *With the learning rates η_V, η_λ, and η_π, the quadratic form of Eq. (5.65) satisfies the following inequality,*

$$\Delta\mathcal{L} \le -\boldsymbol{A}_m\|W\|_2^2 + 2\boldsymbol{B}_M\|M^{(i)}\|_2\|W\|_2 + \boldsymbol{C}_M\|M^{(i)}\|_2^2 \tag{5.75}$$

where $\boldsymbol{A}_m = \lambda_{min}(\boldsymbol{A})$, $\boldsymbol{B}_M = \sigma_{max}\big(B\big)$ *and* $\boldsymbol{C}_M = \lambda_{max}(\boldsymbol{C})$. *Therefore, if*

$$\|W\|_2 > \|M^{(i)}\|_2 \left[\frac{\boldsymbol{B}_M}{\boldsymbol{A}_m} + \sqrt{\left(\frac{\boldsymbol{B}_M}{\boldsymbol{A}_m}\right)^2 + \boldsymbol{C}_M} \right] \tag{5.76}$$

, then $\Delta\mathcal{L} < 0$, *which indicates that the value, costate, and policy networks weights and state are uniformly ultimately bounded (UUB). Moreover, since* $M^{(i)}$ *converges to* M *as* $i \to \infty$.

$$M = \begin{bmatrix} -\epsilon_V^{+*} + \epsilon_V^* \\ -\big(\nabla_X X^+\big)^T \epsilon_\lambda^{+*} + \epsilon_\lambda^* \\ -\epsilon_\pi^{+*} + \left(\frac{1}{2} R^{-1} g^T(X)\right)\epsilon_\lambda^* \\ -g(X)\epsilon_\pi^* \end{bmatrix} \tag{5.77}$$

, and obviously $\|M\|_2 < \|M^{(i)}\|_2$ *by triangular inequality, the ultimate bound of* $\|W\|_2$ *also shrinks to* $\|M\|_2 \left[\frac{\boldsymbol{B}_M}{\boldsymbol{A}_m} + \sqrt{\left(\frac{\boldsymbol{B}_M}{\boldsymbol{A}_m}\right)^2 + \boldsymbol{C}_M} \right]$.
$\square$

Remark 5.4. Note that M is only a function of the DNNs reconstruction errors ϵ_χ^*, which only depend on the structure of the DNNs, thus cannot be reduced by the GDHP algorithm. It can be concluded that the structural design of the DNNs is crucial for both the convergence of the network weights and the closed-loop stability.

5.5 Simulation results and discussions

In this section, the proposed algorithm is implemented on the control problem for the nonlinear distributed system which is presented in [165]. We show that utilizing the DNNs as a function approximator successfully solved the system with high-dimensional state, control, and the output spaces.

5.5.1 System description

Temperature control with a two-dimensional rectangular geometry is presented as a benchmark problem. The rectangular domain is $\Omega = \{(x, y) \mid 0 \leq x \leq x_{\text{max}},\ 0 \leq y \leq y_{\text{max}}\}$ where $x_{\text{max}} = 1.6$ and $y_{\text{max}} = 3.2$. The control boundaries are given as $\partial\Omega_1 = \{(x, y) \mid y = 0\}$ and $\partial\Omega_2 = \{(x, y) \mid x = 0\}$, which are located on the bottom and the left, respectively. The temperature $T(x, y, t)$ in domain Ω is controlled by the heat source located on the control boundaries $\partial\Omega_1$ and $\partial\Omega_2$, while the top and the right boundaries are insulated.

The problem concerns the specific domain of interest given by $\Omega_c = \{(x, y) \mid x_c \leq x \leq x_{\text{max}},\ y_c \leq y \leq y_{\text{max}}\}$ where $x_c = 1.2$ and $y_c = 2.4$. The objective is to control the output which is defined as the average temperature value of the sub-domain

$$z(t) = \frac{1}{|\Omega_c|} \int_{x_c}^{x_{\text{max}}} \int_{y_c}^{y_{\text{max}}} T(x, y, t) dx dy \tag{5.78}$$

to the reference point. The geometry is illustrated in Fig. 5.2.

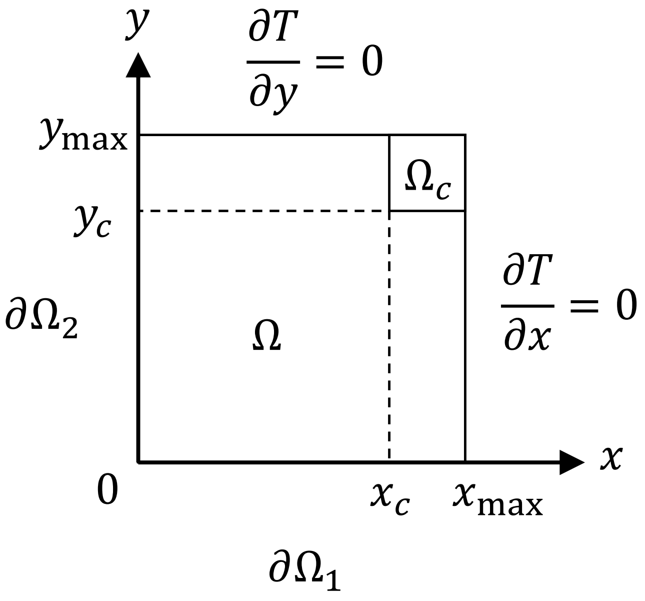


Figure 5.2: Description of the temperature control system domain.

The system in the domain $(x, y, t) \in \Omega \times [0, t_{\text{max}}]$ is governed by the heat equation given as

$$\begin{aligned} \frac{\partial T}{\partial t} &= \frac{\lambda}{c}\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) + S(T), \\ S(T) &= S_M \exp\left(-\frac{\beta_1}{\beta_2 + T}\right) \end{aligned} \tag{5.79}$$

where $\lambda = 0.5$ is the heat conduction coefficient and $c = 0.5$ is the heat capacity. It is assumed that there exists an internal heat source $S(T)$, with the constants $\beta_1 = 0.2$, $\beta_2 = 1.0$, and $S_M = 0.1$. The initial condition is given as $T(x, y, 0) = 0$. The boundary conditions on which the controllers are located are

$$\begin{aligned} T(x, 0, t) - \lambda \frac{\partial T}{\partial y} &= u_1(x, t), \quad x \in \partial\Omega_1 \\ T(0, y, t) - \lambda \frac{\partial T}{\partial x} &= u_2(y, t), \quad y \in \partial\Omega_2 \end{aligned} \tag{5.80}$$

where u_1 and u_2 refer to controls. The boundary conditions regarding the insulation are

$$\begin{aligned} \frac{\partial T}{\partial x}(x_{\text{max}}, y, t) &= 0 \\ \frac{\partial T}{\partial y}(x, y_{\text{max}}, t) &= 0 \end{aligned} \tag{5.81}$$

All the physical properties are dimensionless.

5.5.2 Algorithmic settings

The governing equation is discretized with the finite difference method for the spatial derivative to get the ODE. The domain is discretized by 16×16 grids which yields the system dimensions $S = 256$ and $A = 32$. The system was simulated with a sample time of 0.005. The initial policy is

$$u_1(x,t) = \begin{cases} u(t), & 0 \leq x \leq 0.4 \\ \big(1 - (x - 0.2)/1.2\big)u(t), & 0.4 \leq x \leq 1.6 \end{cases}$$

$$u_2(y,t) = \begin{cases} u(t), & 0 \leq x \leq 0.8 \\ \big(1 - (y - 0.4)/2.4\big)u(t), & 0.8 \leq x \leq 3.2 \end{cases}$$

$$u(t) = \begin{cases} 5t, & 0 \leq t \leq 0.5 \\ 1, & 0.5 \leq t \leq 2.0 \end{cases}$$

The initial controller is employed until episode 30, after which the DNNs policy network is implemented until the final episode, 500.

Five-layer DNNs were used for the value and costate networks and four-layer DNNs were used for policy network. The number of nodes in the value, costate, and policy networks were (S, 256, 128, 64, 1), (S, 256, 128, 128, S), and (S, 256, 128, A), respectively. ReLU activation functions were used and the last layer of policy function was set to be tanh function to constrain the policy value between $[-1, 1]$. Learning rates were set to be $\eta_V = 10^{-3}$, $\eta_\lambda = 10^{-3}$ and $\eta_\pi = 10^{-4}$, respectively.

5.5.3 Control result

With the algorithmic settings, the GDHP algorithm was applied to the system for 500 episodes. We observe the results in episodes 20, 80, 150 and 500, which are the initial policy, early, middle, and the last stages of the GDHP learning process, respectively.

State trajectories of the episodes is illustrated in Fig. 5.3. Each row stands for the the temporal evolution of the two-dimensional snapshot within the episode. According to the geometry description in Fig. 5.2, the north-east part is the target sub-domain Ω_c. The result shows that the sub-domain temperature is controlled to the reference point of 0.15 as the episode proceeds.

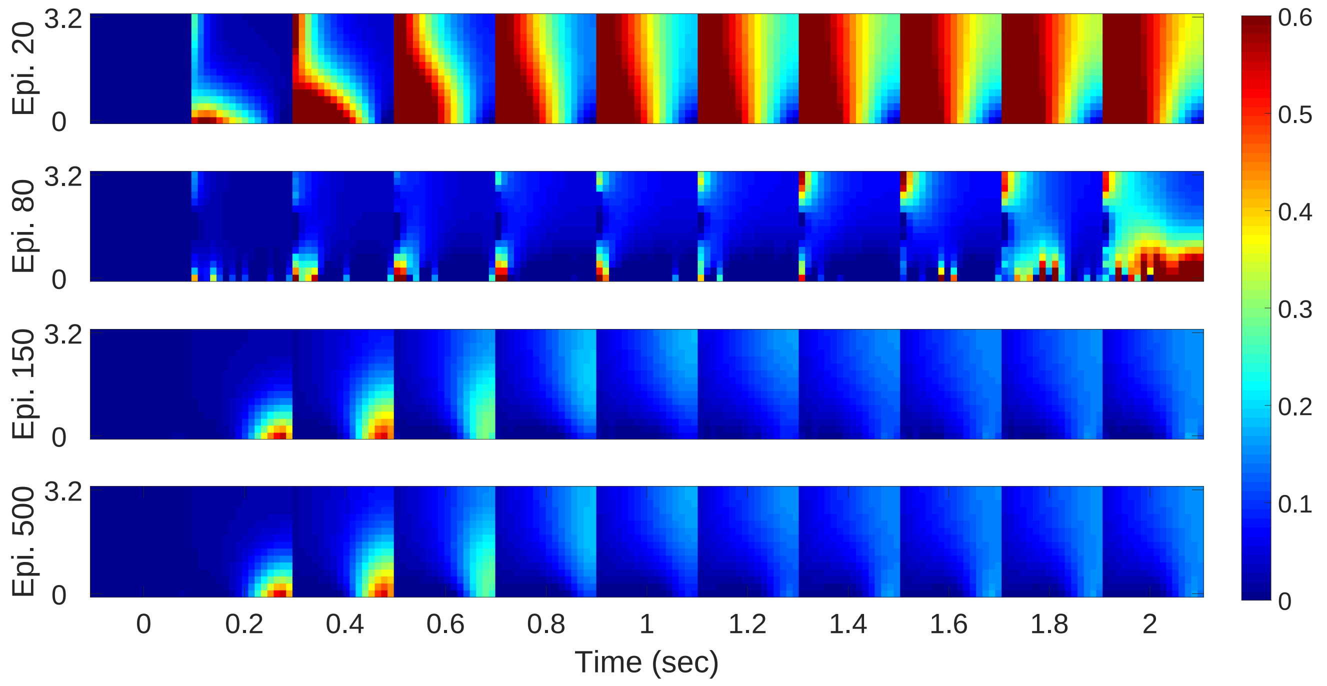


Figure 5.3: Two dimensional plot of state trajectories in episodes 20, 80, 150, and 500. The x-axis denotes the time snapshots and the y-axis is the episode number.

Figures 5.4 and 5.5 show the surface plot of control trajectories $u_1(x,t)$ and $u_2(y,t)$ in episodes 20, 80, 150, and 500, respectively. The control values of episode 80 and 150 are shown to be not converged, while they are stabilized in episode 500.

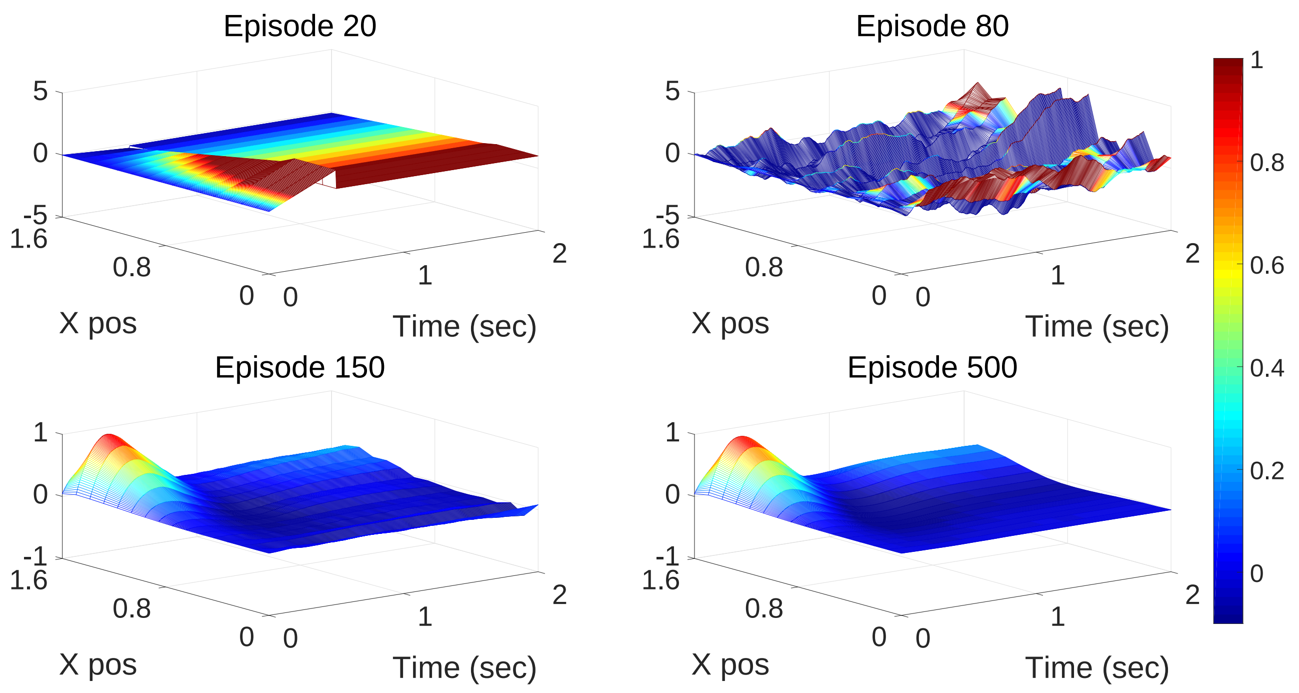


Figure 5.4: Surface plot of temporal trajectories of the control located on the bottom boundary $\partial\Omega_1$ in episodes 0, 80, 150, and 500.

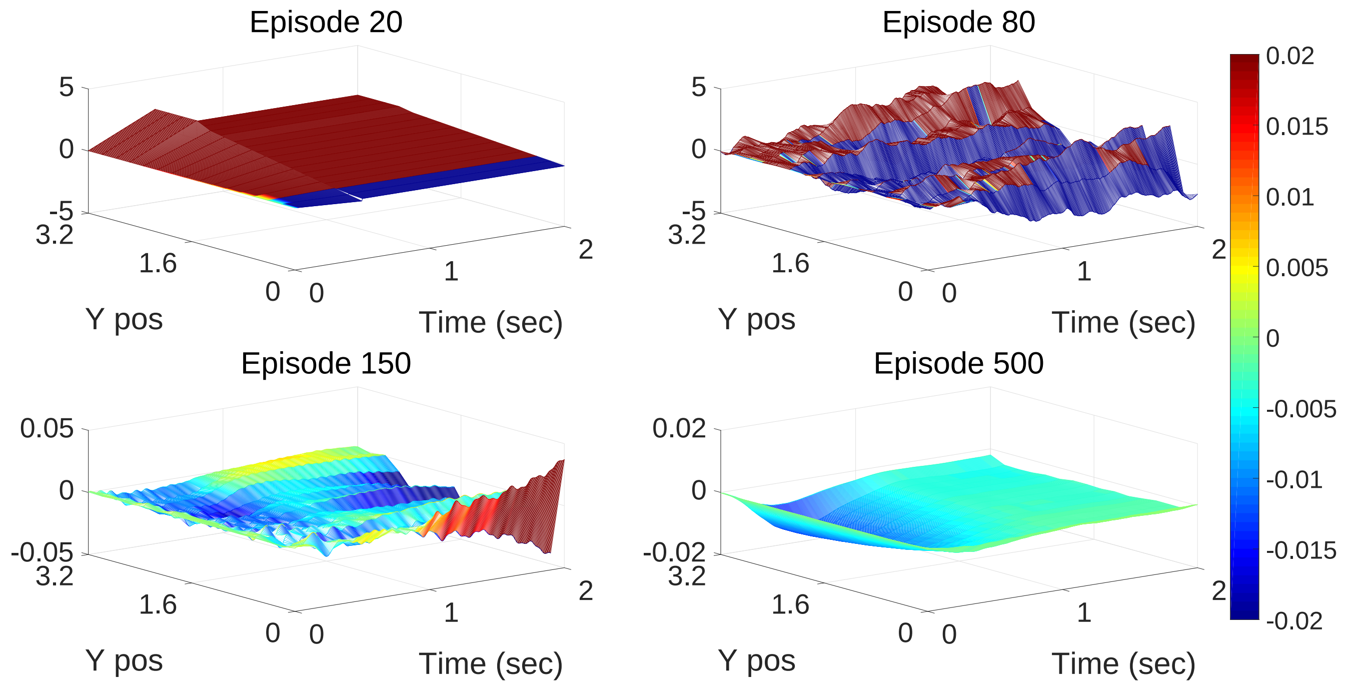


Figure 5.5: Surface plot of temporal trajectories of the control located on the left boundary $\partial\Omega_2$ in episodes 0, 80, 150, and 500.

The output function in Eq. (5.78) is expressed in Fig. 5.6 with the reference point. This figure clearly demonstrates that the control objective is satisfied with the proposed algorithm. In addition, the value network in Fig. 5.7 validates that the DNNs successfully approximates the high-dimensional functions.

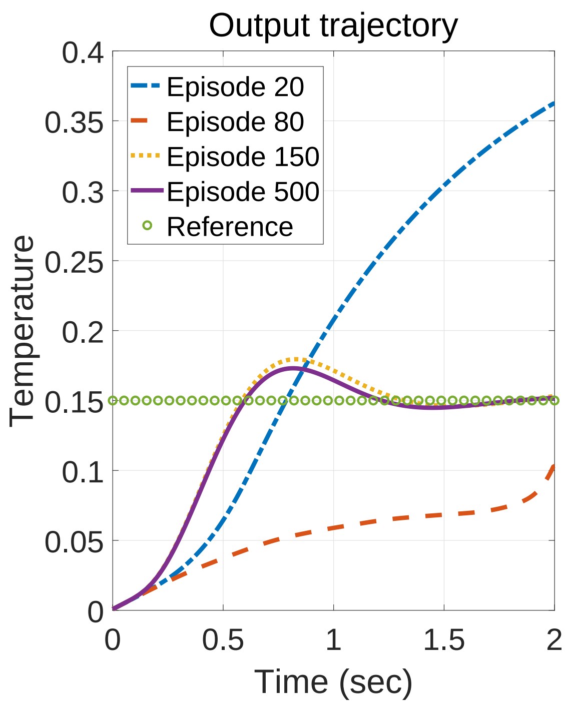


Figure 5.6: Output trajectories in episodes 0, 80, 150, and 500. Circle mark denotes the reference point.

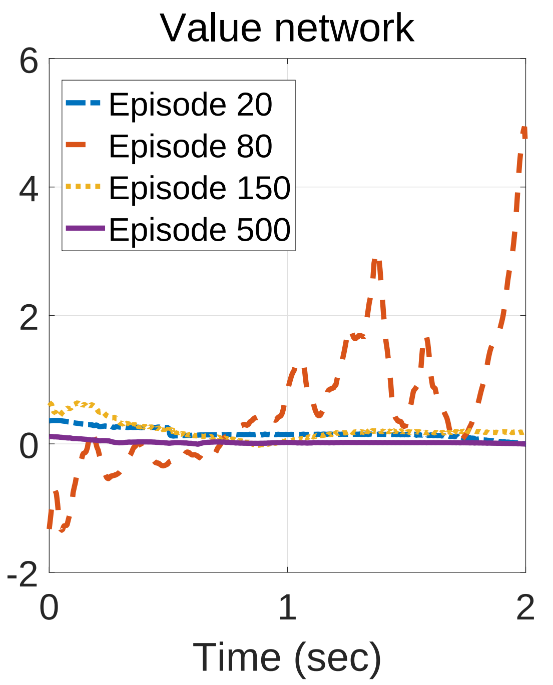


Figure 5.7: Value network in episodes 0, 80, 150, and 500.

Chapter 6

Primal-dual differential dynamic programming for constrained dynamic optimization of continuous system

6.1 Introduction

Dynamic optimization (DO) is an important formulation for engineering discipline, where a variety of numerical methods have been developed [9]. The methods can be classified into direct and indirect method by the static optimization transcription technique [166]. Transcription result of the direct method is a static nonlinear programming (NLP), which explicitly considers the system dynamics, thus can be understood as forward method. On the other hand, the indirect method yield the boundary value problem (BVP), based on the Pontryagin's maximum principle, and the transcription direction is backward [11].

This study focuses on the differential dynamic programming (DDP) method, the third class of numerical method which is a hybrid forward-backward method that considers both the optimality condition expressed by Hamilton-Jacobi-Bellman (HJB) equation and the system dynamics [167]. In DDP framework, the long-horizon DO problem is decomposed into a single-horizon NLP sub-problem. Therefore, the algorithmic complexity is just linear with respect to the horizon and shows a quadratic convergence property [168, 169]. The asymptotic convergence is equivalent result to that of the Newton optimization method, nonetheless, they do not necessarily yield the same search directions in the initial stage [170].

An important extension of the DDP method is the connection with the recently developed reinforcement learning (RL) methods [171, 172, 173, 174]. This is because the solution of the HJB equation is the value function, which can be utilized for the model-based framework in RL problems.

In the viewpoint of constrained DO, well-established NLP methods such as barrier method and penalty method can be directly implemented to the sub-problem. The terminal constraint problem was considered in [175] and the path constrained DDP problem was addressed in [176] and [177]. The general-purpose DDP framework was developed by [178] associated with various stability training methods including trust region method. Aforementioned studies use projected gradient methods thus limited to the constraint having control variable as arguments. The generalized class of constraint (i.e., state variable-only constraint) was solved by [179], [180], and [181] using the augmented Lagrangian method, however the Lagrangian is updated in first-order.

In this study, we tackle the constrained DO problem by the primal-dual value function formalism. The advantages of the proposed method are twofold: the general constraint can be considered and the second-order Newton's method for dual Lagrangian variable can be incorporated.

6.2 Primal-dual differential dynamic programming for constrained dynamic optimization

6.2.1 Augmented Lagrangian method

Among the constraint augmentation method, the augmented Lagrangian (AL) method and the interior-point method are the most frequently considered recently. For example, interior point optimization (IPOPT) provides an excellent framework to compute the central path, by adopting the backtracking line-filter search method to prevent the ill-condition in computing the barrier function [15]. Another important class is AL method. In dynamic optimization problem, trial-and-error based line search method cannot be directly adopted since the objective for each trial point cannot be evaluated immediately. This leads to the frequent constraint violation during the iteration, which causes ill-conditioning in computing the log barrier function at its derivatives. Another advantage of AL method is that the inequality constraints can be augmented without introducing slack variable, hence reducing computational complexity [180]. Moreover, it is known to have better parallel scalability potential and have good warm-starting capabilities [182]. From this point of view, augmented Lagrangian method has been mainly considered in DDP framework [176, 179, 178, 180].

Consider the static optimization problem defined by

$$\begin{aligned} \min_{x} \quad & L(x) \\ \textit{s.t.} \quad & g_k(x) \leq 0, \quad k = 1, \ldots, G \end{aligned} \tag{6.1}$$

The basic form of AL method uses augmented objective as the Lagrange function plus the quadratic penalty function, i.e., $L + \lambda^T g + \frac{1}{2}c\|g\|_2^2$. Here $c > 0$ denotes the penalty parameter. In the case of inequality constraints, the Rockafellar formulation introduces a slack variable $z > 0$ to transform to the following optimization problem in which only equality constraint exists [183].

$$\min_{x,z>0} L + \lambda^T(g + z) + \frac{1}{2}c\|g + z\|_2^2 \tag{6.2}$$

Since minimization of z can be explicitly performed, the minimizer is substituted to yield the following augmentation function:

$$\tilde{L} = L + \frac{1}{2c}\sum_{k=1}^{G}\Big(\max\big[\lambda_k + cg_k, 0\big]^2 - \lambda_k^2\Big) \tag{6.3}$$

The solution essentially shifts the k^{th} inequality constraint boundary from zero to $-\dfrac{\lambda_k}{c}$. For the active inequality constraint, the augmentation function equivalent to that of the equality constraint is used, whereas $-\dfrac{\lambda_k^2}{2c}$ for the inactive inequality constraint [184, 185].

Based on the result in Eq. (6.3), the augmentation function for the original problem Eqs. (2.31) - (2.34) is written as

$$\begin{aligned}\tilde{L}(x(t), u(t), \lambda(t), c(t), t) &= L(x(t), u(t), t)\\ &+ \frac{1}{2c(t)}\sum_{k=1}^{G}\Big(\max\big[\lambda_k(t) + c(t)g_k(x, u, t), 0\big]^2 - \lambda_k^2(t)\Big) \qquad (6.4)\\ \tilde{\phi}(x(t_f), \mu, c(t_f), t_f) &= \phi(x(t_f), t_f)\\ &+ \frac{1}{2c(t_f)}\sum_{k=1}^{H}\Big(\max\big[\mu_k + c(t_f)h_k(x), 0\big]^2 - \mu_k^2\Big) \qquad (6.5)\end{aligned}$$

where $c(t) > 0$ now becomes a time-varying penalty parameter. In the case of equality constaints, the max operator is omitted. The path cost augmentation function can be written in the alternative form:

$$\psi_k(\lambda_k, g_k, c) = \begin{cases} 0 & \text{if} \quad g_k \leq -\dfrac{\lambda_k}{c} \\ \lambda_k(t) g_k + \dfrac{1}{2} c(t) g_k^2 + \dfrac{1}{2c(t)} \lambda_k^2(t) & \text{otherwise} \end{cases} \tag{6.6}$$

where $\psi(\lambda, g, c) = \sum_{k=1}^{G} \psi_k(\lambda_k, g_k, c)$ and analogously, terminal augmentation function is $\psi(\mu, h, c) = \sum_{k=1}^{H} \psi_k(\mu_k, h_k, c)$.

The unconstrained objective provides the approximate optimizer by regarding the c as a hyperparameter, in which the exact optimizer is recovered as $c \to \infty$ [186]. The augmentation function is second order differentiable with respect to x, u, and λ thus can be implemented in DDP algorithm. The augmented Lagrangian method inherits the properties of both the Lagrangian method and the penalty method, and complements the issues of both methods. The pure penalty method requires the penalty parameter c to be infinity in order to obtain the feasibility, while using of the KKT condition of Lagrangian method relax the problem. Moreover, the KKT condition has numerical problem when the objective is nonconvex, which can be resolved by the penalty parameter c, which forces the Hessian of the augmented objective $\tilde{J}$ to be positive definite.

6.2.2 Primal-dual differential dynamic programming algorithm

DDP algorithm uses quadratic expansions of the functions around the previous trajectory $\big(\bar{x}(t), \bar{u}(t), \bar{\lambda}(t)\big)$. In accordance, the 'bar' notation is used for an arbitrary function $F(x)$ being evaluated at the nominal condition $\bar{x}$ as $\bar{F}(\bar{x})$.

By taking the expansion of the both sides of Eq. (2.36), we can obtain the relationships of not only the value function V itself, but also the first and second order derivatives of value function [174]. Using the total derivative rule $\partial(\cdot)/\partial t = d(\cdot)/dt - \partial(\cdot)/\partial x \cdot \bar{f}$ for functions $\bar{V}$, $\bar{V}_x$, and $\bar{V}_{xx}$, following equation is obtained:

$$\frac{\partial V(x,t)}{\partial t} \approx \frac{d}{dt}\Big(\bar{V} + \bar{V}_x^T \delta x + \frac{1}{2}\delta x^T \bar{V}_{xx} \delta x\Big) - \bar{V}_x^T \bar{f} - \delta x^T \bar{V}_{xx} \bar{f} - \mathcal{V} \tag{6.7}$$

$\mathcal{V}$ denotes the third order derivative:

$$\mathcal{V} = \frac{1}{2} \sum_{i=1}^{S} \delta x^T \bar{V}_{xxx}^{(i)} \bar{f}^{(i)} \delta x \tag{6.8}$$

where $\bar{V}_{xxx}^{(i)}$ is the Hessian of the i^{th} element of $\bar{V}_x$ and $\bar{f}^{(i)}$ denotes the i^{th} element of $\bar{f}$. The second order expansion of the right hand side of

Eq. (2.36) can be done using the expansions of $\bar{\bar{L}}$, $\bar{V}_x$, and f as

$$\begin{aligned}
&\bar{\bar{L}} + \bar{V}_x^T \bar{f} \\
&\approx \bar{\bar{L}} + \begin{bmatrix} \bar{\bar{L}}_x \\ \bar{\bar{L}}_u \\ \bar{\bar{L}}_\lambda \end{bmatrix}^T \begin{bmatrix} \delta x \\ \delta u \\ \delta \lambda \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \delta x \\ \delta u \\ \delta \lambda \end{bmatrix}^T \begin{bmatrix} \bar{\bar{L}}_{xx} & \bar{\bar{L}}_{xu} & \bar{\bar{L}}_{x\lambda} \\ \bar{\bar{L}}_{ux} & \bar{\bar{L}}_{uu} & \bar{\bar{L}}_{u\lambda} \\ \bar{\bar{L}}_{\lambda x} & \bar{\bar{L}}_{\lambda u} & \bar{\bar{L}}_{\lambda\lambda} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \\ \delta \lambda \end{bmatrix} \\
&+ \bar{V}_x^T \bar{f} + \bar{V}_x^T \bar{f}_x \delta x + \bar{V}_x^T \bar{f}_u \delta u \\
&+ \delta x^T \bar{V}_{xx} \bar{f} + \delta x^T \bar{V}_{xx} \bar{f}_x \delta x + \delta x^T \bar{V}_{xx} \bar{f}_u \delta u + \mathcal{V} \qquad (6.9)
\end{aligned}$$

Finally, Taylor expansions of both hand sides Eqs. (6.7) - (6.9) are equated as

$$\begin{aligned}
&\frac{d}{dt}\left(\bar{V} + \bar{V}_x^T \delta x + \frac{1}{2} \delta x^T \bar{V}_{xx} \delta x\right) \\
&= \min_{\delta u} \max_{\delta \lambda} \left[\bar{\bar{L}} + \begin{bmatrix} \bar{\mathcal{H}}_x \\ \bar{\mathcal{H}}_u \\ \bar{\mathcal{H}}_\lambda \end{bmatrix}^T \begin{bmatrix} \delta x \\ \delta u \\ \delta \lambda \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \delta x \\ \delta u \\ \delta \lambda \end{bmatrix}^T \begin{bmatrix} \bar{\mathcal{H}}_{xx} & \bar{\mathcal{H}}_{xu} & \bar{\mathcal{H}}_{x\lambda} \\ \bar{\mathcal{H}}_{ux} & \bar{\mathcal{H}}_{uu} & \bar{\mathcal{H}}_{u\lambda} \\ \bar{\mathcal{H}}_{\lambda x} & \bar{\mathcal{H}}_{\lambda u} & \bar{\mathcal{H}}_{\lambda\lambda} \end{bmatrix} \begin{bmatrix} \delta x \\ \delta u \\ \delta \lambda \end{bmatrix} \right] \\
&\qquad (6.10)
\end{aligned}$$

where the derivatives of Hamiltonian $\bar{\mathcal{H}}$ become the coefficients for the Jacobian and the Hessian in the min-max objective.

$$\bar{\mathcal{H}}_x = \bar{\tilde{L}}_x + \bar{f}_x^T \bar{V}_x \tag{6.11}$$

$$\bar{\mathcal{H}}_u = \bar{\tilde{L}}_u + \bar{f}_u^T \bar{V}_x \tag{6.12}$$

$$\bar{\mathcal{H}}_\lambda = \bar{\tilde{L}}_\lambda \tag{6.13}$$

$$\bar{\mathcal{H}}_{xx} = \bar{\tilde{L}}_{xx} + \bar{V}_{xx}\bar{f}_x + \bar{f}_x^T \bar{V}_{xx} + \sum_{i=1}^{S} \bar{V}_x^{(i)} \bar{f}_{xx}^{(i)} \tag{6.14}$$

$$\bar{\mathcal{H}}_{xu} = \bar{\tilde{L}}_{xu} + \bar{V}_{xx}\bar{f}_u + \sum_{i=1}^{S} \bar{V}_x^{(i)} \bar{f}_{xu}^{(i)} \tag{6.15}$$

$$\bar{\mathcal{H}}_{x\lambda} = \bar{\tilde{L}}_{x\lambda} \tag{6.16}$$

$$\bar{\mathcal{H}}_{uu} = \bar{\tilde{L}}_{uu} + \sum_{i=1}^{S} \bar{V}_x^{(i)} \bar{f}_{uu}^{(i)} \tag{6.17}$$

$$\bar{\mathcal{H}}_{u\lambda} = \bar{\tilde{L}}_{u\lambda} \tag{6.18}$$

$$\bar{\mathcal{H}}_{\lambda\lambda} = \bar{\tilde{L}}_{\lambda\lambda} \tag{6.19}$$

The right hand side of Eq. (6.10) takes quadratic form, for which the explicit minimizer δu^* can be obtained as

$$\begin{bmatrix} \bar{\mathcal{H}}_{uu} & \bar{\mathcal{H}}_{u\lambda} \\ \bar{\mathcal{H}}_{\lambda u} & \bar{\mathcal{H}}_{\lambda\lambda} \end{bmatrix} \begin{bmatrix} \delta u^*(t) \\ \delta\lambda^*(t) \end{bmatrix} = - \begin{bmatrix} \bar{\mathcal{H}}_u + \bar{\mathcal{H}}_{ux}\delta x \\ \bar{\mathcal{H}}_\lambda + \bar{\mathcal{H}}_{\lambda x}\delta x \end{bmatrix} \tag{6.20}$$

The result of the linear equation is the optimizer of the min-max objective, which is written as

$$\delta u^*(t) = l_u + K_{xu}\delta x, \qquad \delta\lambda^*(t) = l_\lambda + K_{x\lambda}\delta x \tag{6.21}$$

Finally, the backward ODEs for $\bar{V}$ and its first and second order derivatives are obtained by substituting δu^* and $\delta\lambda^*$ into Eq. (6.10) and comparing the coefficients as

$$-\frac{d\bar{V}}{dt} = \bar{\bar{L}} + l^T\bar{\mathcal{H}}_v + \frac{1}{2}l^T\bar{\mathcal{H}}_M l \tag{6.22}$$

$$-\frac{d\bar{V}_x}{dt} = \bar{\mathcal{H}}_x + K_x^T\bar{\mathcal{H}}_v + \bar{\mathcal{H}}_{vx}^T l + K_x^T\bar{\mathcal{H}}_M l \tag{6.23}$$

$$-\frac{d\bar{V}_{xx}}{dt} = \bar{\mathcal{H}}_{xx} + K_x^T\bar{\mathcal{H}}_{vx} + \bar{\mathcal{H}}_{vx}^T K_x + K_x^T\bar{\mathcal{H}}_M K_x \tag{6.24}$$

where

$$l = [l_u; l_v], \quad K_x = [K_{xu}; K_{xv}] \tag{6.25}$$

$$\bar{\mathcal{H}}_v = [\bar{\mathcal{H}}_u; \bar{\mathcal{H}}_\lambda], \quad \bar{\mathcal{H}}_{vx} = [\bar{\mathcal{H}}_{ux}; \bar{\mathcal{H}}_{\lambda x}], \quad \bar{\mathcal{H}}_M = \begin{bmatrix} \bar{\mathcal{H}}_{uu} & \bar{\mathcal{H}}_{u\lambda} \\ \bar{\mathcal{H}}_{\lambda u} & \bar{\mathcal{H}}_{\lambda\lambda} \end{bmatrix} \tag{6.26}$$

where ; denotes the vertical concatenation of two vectors. The backward ODEs are subject to the following terminal boundary conditions

$$\bar{V}(x, t_f) = \bar{\bar{\phi}}^*(\bar{x}(t_f), t_f) \tag{6.27}$$

$$\bar{V}_x(x, t_f) = \bar{\bar{\phi}}_x^*(\bar{x}(t_f), t_f) \tag{6.28}$$

$$\bar{V}_{xx}(x, t_f) = \bar{\bar{\phi}}_{xx}^*(\bar{x}(t_f), t_f) \tag{6.29}$$

where

$$\bar{\bar{\phi}}^*(\bar{x}(t_f), t_f) = \max_{\mu} \bar{\bar{\phi}}(\bar{x}(t_f), \mu, t_f) \tag{6.30}$$

6.2.3 Overall algorithm

The overall constrained DDP algorithm is consisted with two stages: backward sweep and forward sweep.

The backward sweep aims to compute the Hamiltonians and the Lagrangian variable update. The backward ODEs with respect to the value function V and its derivative (Eqs. (6.22)-(6.26)) are solved from their terminal boundary conditions Eqs. (6.27)-(6.29). During solving the ODEs, the Hamiltonian $\bar{\mathcal{H}}$ and its derivatives (Eqs. (6.11)-(6.19)), and the feedforward feedback gains l and K_x (Eq. (6.25)) are obtained. The second order update of Lagrangian variable is then computed. In the forward sweep, the control variable is updated. The backward-forward scheme is repeated until the convergence criteria is satisfied. The overall scheme is summarized in Algorithm 2.

6.3 Results and discussions

A constrained optimal control problem of Van der Pol oscillator is demonstrated with the proposed DDP algorithm. The state-space model and the initial condition of the oscillator can be written as

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= -x_1 + x_2(1.4 - 0.14x_2^2) + 4u \\ x_1(0) &= -5, \quad x_2(0) = -5 \end{aligned} \tag{6.31}$$

Algorithm 2 DDP

```
1: procedure CONSTRAINED DDP
   Initialize: Initial guess of control $\bar{u}$ and corresponding state trajectory $\bar{x}$. Initial guess of Lagrangian $\bar{\lambda}$, penalty parameter $c_0$
2:     while Not converged do
3:         Backward sweep
4:         for $k = t_f$ to $t_0$ do
5:             Compute Hamiltonian $\bar{\mathcal{H}}$ and its derivatives Eqs. (6.11)-(6.19)
6:             Compute feedforward and feedback gains $l$ and $K_x$ by Eq. (6.25)
7:             Solve backward ODEs of $V$ and its derivatives with Eqs. (6.22)-(6.26)
8:         Forward sweep
9:         for $k = t_0$ to $t_f$ do
10:            Perform the forward operation of the system dynamics to obtain $\delta x$
11:            Second order updates of control and Lagrangian variables $\delta u$ and $\delta\lambda$ with Eq. (6.21)
12:        Increase penalty parameter $c = \gamma c$, $\gamma > 1$
```

The objective is to control the position coordinate $x_1(t)$ to zero within the horizon $t_f = 4.5$

$$J(u) = \int_0^{4.5} (x_1^2(t) + u^2(t))dt \tag{6.32}$$

The system is subject to the path constraint

$$-1 \leq u(t) + \frac{x_1(t)}{6} \leq 0, \quad \forall t \in [0, 4.5] \tag{6.33}$$

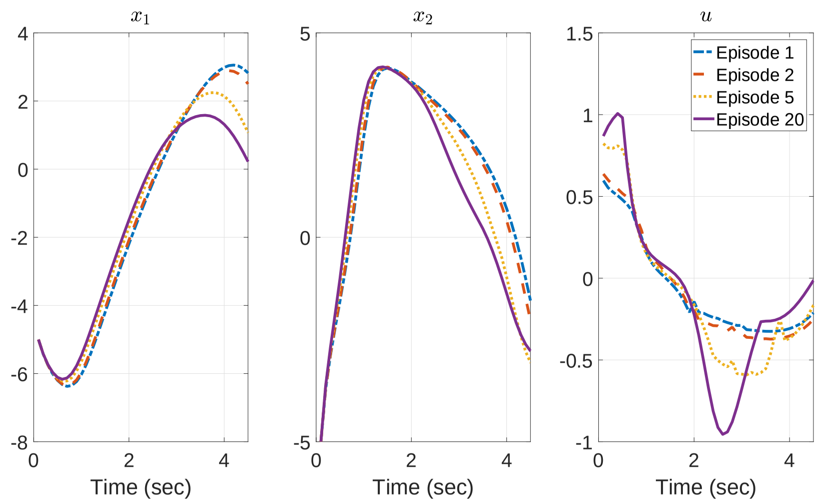


Figure 6.1: State and control trajectories under DDP algorithm in episode 1, 2, 5, and 20.

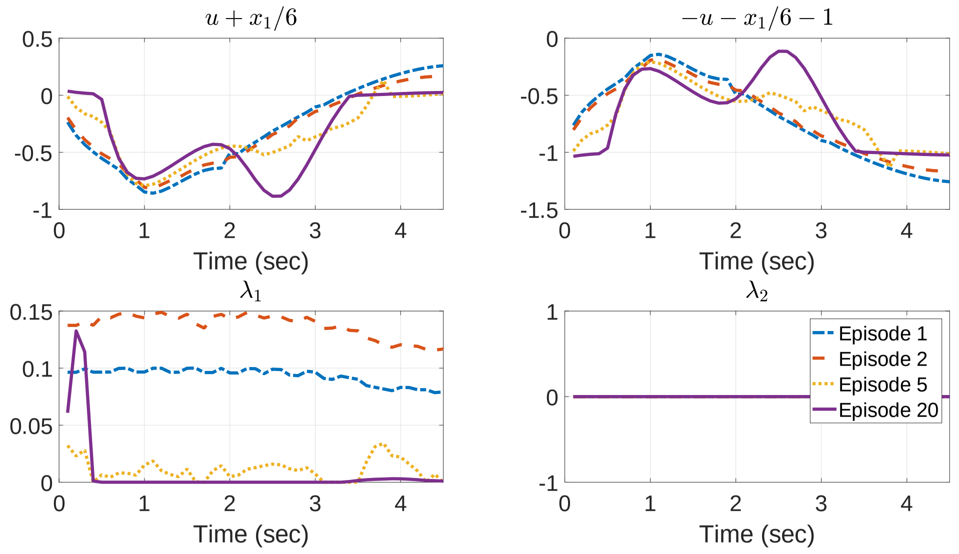


Figure 6.2: Two constraints and corresponding Lagrangian variables in episode 1, 2, 5, and 20.

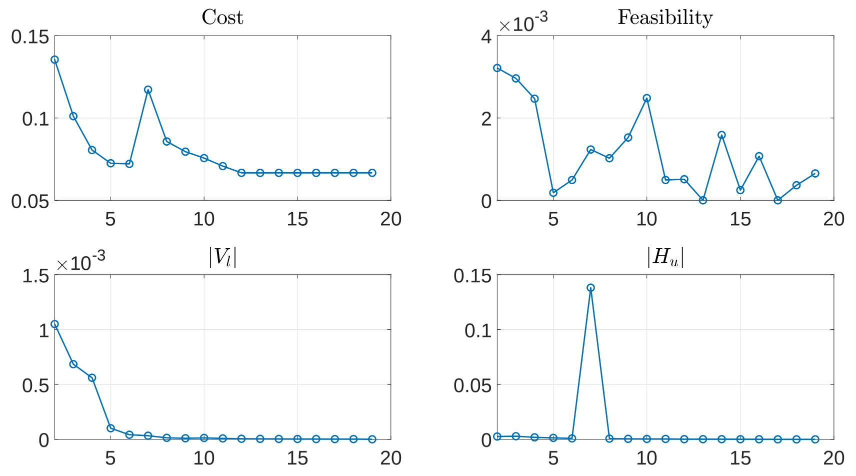


Figure 6.3: Episode cost, feasibility index, $|V_l|$, and $|H_u|$ plot.

After implementing the initial PID controller, proposed DDP algorithm is adopted for 20 episodes. Figure 6.1 shows the state and control trajectories under the policies computed in episodes 1, 2, 5, and 20. Compared to the initial PID controller in the first episode, the final policy yields a better control result. Two path constraints Eq. (6.33) and corresponding Lagrangian variables are illustrated in Fig. 6.2. It indicates that the path constraints are satisfied as episodes proceed, while the infeasibility occurs in the initial controller. In addition, proposed Lagrangian update rule satisfies the complementary slackness condition. We selected four monitoring statistics as episode cost, feasibility index, $|V_l|$, and $|H_u|$. As episode proceeds, the episode cost should decrease and the last three statistics should be converged within small tolerance values. Figure 6.3 show that the proposed algorithm provides proper convergence criteria.

Chapter 7

Concluding remarks

Sequential decision making problem is a crucial technology for plant-wide process optimization. While the dominant numerical method is the forward-in-time direct optimization, it is limited to the open-loop solution and has difficulty in considering the uncertainty. Dynamic programming method complements the limitations, nonetheless associated functional optimization suffers from the curse of dimensionality. The sample-based approach for approximating the dynamic programming, referred to as reinforcement learning (RL) can resolve the issue and investigated throughout this thesis. The method that accounts for the system model explicitly is in particular interest. The model-based RL is exploited to solve the three representative sequential decision making problems; scheduling, supervisory optimization, and regulatory control. The problems are formulated with partially observable Markov decision process, control-affine state space model, and general state space model, and associated model-based RL algorithms are point based value iteration (PBVI), globalized dual heuristic programming (GDHP), and differential dynamic programming (DDP), respectively.

The special features in each problems can be written as follows: First, for the scheduling problem, we developed the closed-loop feedback scheme which highlights the strength compared to the direct optimization method. In the second case, the regulatory control problem is tackled by the function approximation method which relaxes the functional optimization to the finite dimensional vector space optimization. Deep neural networks (DNNs) is utilized as the approximator, and the advantages as well as the convergence analysis is performed in the thesis. Finally, for the supervisory optimization problem, we developed the novel constraint RL framework that uses the primal-dual DDP method. Various illustrative examples are demonstrated to support the validity of the thesis statement.

7.1 Summary of the contributions

In this thesis, model-based RL algorithm is studied for solving the three representative sequential decision making problems; scheduling, supervisory optimization, and regulatory control. Point based value iteration (PBVI), globalized dual heuristic programming (GDHP), and differential dynamic programming (DDP) are the particular algorithms for POMDP, control-affine state space model, and general state space model.

The first part is about the infrastructure scheduling framework where maintenance, inspection, and sensor-installation are concerend. Inspection schedule is formulated by modeling the observation uncertainty of the system. Moreover, the sensor-installation is optimized by concerning the various costs of inspection methods and the installation cost. The complex heterogeneous schedule is successfully obtained by exploiting the feature of POMDP which is an effective framework for the discrete stochastic sequential decision process. Although the infinite horizon POMDP yields the belief-action mapping via alphavectors, it still depends on the root node of the policy tree. The fixed root node causes an inaccuracy to the value function approximation since PBVI algorithm relies on the heuristic belief collection. The receding horizon scheme is suggested to alleviate the dependency problem, based on the infinite horizon result and the offline and online methods for collecting the observation.

In the second part, the GDHP algorithm is formulated for the FHOC problem. To ensure good representation of the involved functions in high dimensional state space systems, we adopted DNNs. Various tricks to ensure stable learning, e.g., the replay buffer, the target network, pre-training, and higher-order discretization methods were utilized. The use of DNNs were shown to give improved robustness with respect to various stochastic uncertainties, such as those in the initial state and state transitions, especially in the context of a high dimensional dissipative PDE system. We also performed the convergence analysis of the GDHP algorithm with DNNs, by taking the three steps. First, the condition for which the costate iteration operator to be contraction was derived. Second, the moving target regression problem was considered by the three-steps decomposition. Finally, the UUB of DNNs weights and closed-loop stability were proven. Temperature control example shows that the proposed method enables us to obtain the optimal policy for the system with state dimension 256.

The last part of the thesis concerns the constrained dynamic optimization problem with primal-dual DDP framework. The game-theoretic saddle point problem is exploited to formulate the min-max operation for the primal-dual DDP. Based on the DDP formulation, the primal-dual augmented Lagrangian method is combined as a cost augmentation method. In addition, the trust region method and momentum based line search methods are utilized to enhance the efficiency of the Newton step.

7.2 Future works

The three problems are restricted to the simplified settings such as considering a single hierarchy, deterministic case, perfect model, compact domain optimization, etc. In fact, the real world problems have all kinds of difficulties in a single case. For instance, the plant-wide optimization should involve the multiple hierarchies of time and spatial scales, uncertainties from noise and imperfect model knowledge, decision variables in hybrid discrete and continuous domains. Hence, the future research direction should focus on the complex formulation, the integrated hierarchies, and also exploring the connection with other fields of computational sciences, which is also addressed in numerous review papers [5, 8, 16, 31, 57, 187].

The first possible extension is to the stochastic formulation, which is related to the theories of stochastic differential equation, stochastic filtering, and the stochastic optimal control [188]. Although basic numerical experiments about the robustness on using DNNs in the presence of the uncertainties of the initial state and state transition are performed in Chapter **4**, we should implement the stochastic theories as it has been done in the model predictive control [23, 189] and optimal control [64, 65, 172, 190, 191]. It is based on the solution methods for stochastic version of HJB or the approximation of the probability distribution such as Gaussian or the polynomial chaos expansion. Equipped with this theory, then the conventional control-loop structure that includes state estimation and/or on-line identification can be coupled with the approach.

The second direction is related to the variations in the system dynamics. Current implementations only consider the ODE or the discretized PDE for the system model with the decision variables contained in the compact set or finite set. However, many PSE problems are modelled as a differential algebraic equation (DAE) with the million dimension of mixed integer decision variables [12], and the direct optimization methods are successfully utilized for solving general problem settings [8, 16]. In order to develop DP based method as one of the candidate algorithm for generic optimization problem compared to other commercial softwares, it should be tested onto the same problem classes. Some possible modifications for accelerating the performance are from the ideas of direct optimization and DRL methods. The examples that enhance the direct optimization solver are the multiple shooting method and the line search filter method. Methodologies in DRL facilitates the efficient machine learning algorithms as well as well-designed DNNs structures suitable for the large-scale problems. Moreover, the new optimality conditions of DAE and mixed-integer programming for DP based solver should be explored [192].

The integration of the PSE hierarchies should be considered as a future research topic. Integrations of planning, scheduling, supervisory optimization, and regulatory control is an active research area [5, 193, 194]. While the integration requires the closed-loop solution to respond to the feedback signals from either higher or lower layers, the DP based solver can potentially advantageous to this situation. We suggest the hierarchical DP formulation that contains the different time-scales within a single problem, where the numerical ill-conditioning issue risen from the long-horizon can be tackled with the similar method that have been taken in multiple-shooting DP algorithm.

In the viewpoint of RL fields, as explained in Chapter **1**, there are two classes of algorithms which are model-based and model-free. Since either of the methods have disadvantages when used alone due to the model-plant mismatch and the data inefficiency, recent RL researches try to find the combination between the two methods. Some results learn model and policy simultaneously [195, 196], control the rolling horizon by evaluating the value function accuracy obtained from model with plant data [197, 198], or find the explicit method to compensate the model-plant mismatch to the estimated value function [199]. These researches can be jointly implemented with the offset-free tracking method in MPC, so that the mismatch could be reduced and to take the advantages of both model-based and model-free RL algorithms.

Bibliography

[1] S. Skogestad, "Plantwide control: The search for the self-optimizing control structure," *Journal of process control*, vol. 10, no. 5, pp. 487–507, 2000.

[2] D. E. Seborg, D. A. Mellichamp, T. F. Edgar, and F. J. Doyle III, *Process dynamics and control*. John Wiley & Sons, 2010.

[3] S. Engell and I. Harjunkoski, "Optimal operation: Scheduling, advanced control and their integration," *Computers & Chemical Engineering*, vol. 47, pp. 121–133, 2012.

[4] A. J. Isaksson, I. Harjunkoski, and G. Sand, "The impact of digitalization on the future of control and operations," *Computers & Chemical Engineering*, vol. 114, pp. 122–129, 2018.

[5] J. B. Rawlings and C. T. Maravelias, "Bringing new technologies and approaches to the operation and control of chemical process systems," *AIChE Journal*, vol. 65, no. 6, 2019.

[6] L. T. Biegler and I. E. Grossmann, "Retrospective on optimization," *Computers & Chemical Engineering*, vol. 28, no. 8, pp. 1169–1192, 2004.

[7] I. E. Grossmann and L. T. Biegler, "Part ii. future perspective on optimization," *Computers & Chemical Engineering*, vol. 28, no. 8, pp. 1193–1218, 2004.

[8] I. E. Grossmann and I. Harjunkoski, "Process systems engineering: Academic and industrial perspectives," *Computers & Chemical Engineering*, vol. 126, pp. 474–484, 2019.

[9] A. V. Rao, "A survey of numerical methods for optimal control," *Advances in the Astronautical Sciences*, vol. 135, no. 1, pp. 497–528, 2009.

[10] M. Bardi and I. Capuzzo-Dolcetta, *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*. Springer Science & Business Media, 2008.

[11] B. Chachuat, "Nonlinear and dynamic optimization: From theory to practice," report, 2007.

[12] L. T. Biegler, "An overview of simultaneous strategies for dynamic optimization," *Chemical Engineering and Processing: Process Intensification*, vol. 46, no. 11, pp. 1043–1053, 2007.

[13] D. P. Bertsekas, *Dynamic programming and optimal control*, vol. 1. Athena Scientific Belmont, MA, 2005.

[14] B. Baumrucker, J. Renfro, and L. T. Biegler, "Mpec problem formulations and solution strategies with chemical engineering applications," *Computers & Chemical Engineering*, vol. 32, no. 12, pp. 2903–2913, 2008.

[15] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[16] L. T. Biegler, "New nonlinear programming paradigms for the future of process optimization," *AIChE Journal*, vol. 63, no. 4, pp. 1178–1193, 2017.

[17] S. L. Janak, X. Lin, and C. A. Floudas, "A new robust optimization approach for scheduling under uncertainty: Ii. uncertainty with known probability distribution," *Computers & chemical engineering*, vol. 31, no. 3, pp. 171–195, 2007.

[18] Z. Li and M. Ierapetritou, "Process scheduling under uncertainty: Review and challenges," *Computers & Chemical Engineering*, vol. 32, no. 4-5, pp. 715–727, 2008.

[19] J. H. Lee, “From robust model predictive control to stochastic optimal control and approximate dynamic programming: A perspective gained from a personal journey,” *Computers & Chemical Engineering*, vol. 70, pp. 114–121, 2014.

[20] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, “The explicit linear quadratic regulator for constrained systems,” *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[21] G. M. Kopanos and E. N. Pistikopoulos, “Reactive scheduling by a multiparametric programming rolling horizon framework: a case of a network of combined heat and power units,” *Industrial & Engineering Chemistry Research*, vol. 53, no. 11, pp. 4366–4386, 2014.

[22] G. Pannocchia and J. B. Rawlings, “Disturbance models for offset-free model-predictive control,” *AIChE journal*, vol. 49, no. 2, pp. 426–437, 2003.

[23] A. Mesbah, “Stochastic model predictive control: An overview and perspectives for future research,” *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 30–44, 2016.

[24] D. Gupta, C. T. Maravelias, and J. M. Wassick, “From rescheduling to online scheduling,” *Chemical Engineering Research and Design*, vol. 116, pp. 83–97, 2016.

[25] J. Cui and S. Engell, “Medium-term planning of a multiproduct batch plant under evolving multi-period multi-uncertainty by means of a moving horizon strategy,” *Computers & Chemical Engineering*, vol. 34, no. 5, pp. 598–619, 2010.

[26] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[27] V. M. Zavala and L. T. Biegler, “The advanced-step nmpc controller: Optimality, stability and robustness,” *Automatica*, vol. 45, no. 1, pp. 86–93, 2009.

[28] B. Chachuat, B. Srinivasan, and D. Bonvin, “Adaptation strategies for real-time optimization,” *Computers & Chemical Engineering*, vol. 33, no. 10, pp. 1557–1567, 2009.

[29] J. H. Lee, “From robust model predictive control to stochastic optimal control and approximate dynamic programming: A perspective gained from a personal journey,” *Computers & Chemical Engineering*, vol. 70, pp. 114–121, 2014.

[30] S. J. Lee, G. Lee, J. C. Suh, and J. M. Lee, “Online burst detection and location of water distribution systems and its practical applications,” *Journal of Water Resources Planning and Management*, vol. 142, no. 1, p. 04015033, 2015.

[31] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko, “Reinforcement learning for control: Performance, stability, and deep approximators,” *Annual Reviews in Control*, 2018.

[32] J. Shin, T. A. Badgwell, K.-H. Liu, and J. H. Lee, “Reinforcement learning–overview of recent progress and implications for process control,” *Computers & Chemical Engineering*, vol. 127, pp. 282–294, 2019.

[33] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic programming*, vol. 5. Athena Scientific Belmont, MA, 1996.

[34] F. L. Lewis and D. Vrabie, “Reinforcement learning and adaptive dynamic programming for feedback control,” *IEEE Circuits and Systems Magazine*, vol. 9, no. 3, 2009.

[35] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*, vol. 703. John Wiley & Sons, 2007.

[36] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[37] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[38] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1433–1440, IEEE, 2016.

[39] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[40] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.

[41] S. Spielberg, A. Tulsyan, N. P. Lawrence, P. D. Loewen, and R. B. Gopaluni, "Towards self-driving processes: A deep reinforcement learning approach to control," *AIChE Journal*, 2019.

[42] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst, *Reinforcement learning and dynamic programming using function approximators*. CRC press, 2017.

[43] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human–level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[44] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," *arXiv preprint arXiv:1511.06581*, 2015.

[45] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[46] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 449–458, JMLR. org, 2017.

[47] M. P. Deisenroth, G. Neumann, J. Peters, *et al.*, "A survey on policy search for robotics," *Foundations and Trends® in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2013.

[48] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, pp. 1057–1063, 2000.

[49] S. M. Kakade, "A natural policy gradient," in *Advances in neural information processing systems*, pp. 1531–1538, 2002.

[50] J. Kober and J. R. Peters, "Policy search for motor primitives in robotics," in *Advances in neural information processing systems*, pp. 849–856, 2009.

[51] J. Peters, K. Mulling, and Y. Altun, "Relative entropy policy search," in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[52] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[53] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, 2016.

[54] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," *arXiv preprint arXiv:1611.01224*, 2016.

[55] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, 2015.

[56] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[57] V. Venkatasubramanian, "The promise of artificial intelligence in chemical engineering: Is it here, finally?," *AIChE Journal*, vol. 65, no. 2, pp. 466–478, 2019.

[58] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning*, pp. 1–9, 2013.

[59] B. Recht, "A tour of reinforcement learning: The view from continuous control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 253–279, 2019.

[60] W. Sun, N. Jiang, A. Krishnamurthy, A. Agarwal, and J. Langford, "Model-based rl in contextual decision processes: Pac bounds and exponential improvements over model-free approaches," in *Conference on Learning Theory*, pp. 2898–2933, 2019.

[61] D. V. Prokhorov and D. C. Wunsch, "Adaptive critic designs," *IEEE transactions on Neural Networks*, vol. 8, no. 5, pp. 997–1007, 1997.

[62] M. Abu-Khalaf and F. L. Lewis, "Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach," *Automatica*, vol. 41, no. 5, pp. 779–791, 2005.

[63] Y. Jiang and Z.-P. Jiang, "Robust adaptive dynamic programming and feedback stabilization of nonlinear systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 882–893, 2014.

[64] E. Theodorou, J. Buchli, and S. Schaal, “A generalized path integral control approach to reinforcement learning,” *journal of machine learning research*, vol. 11, no. Nov, pp. 3137–3181, 2010.

[65] E. Theodorou, Y. Tassa, and E. Todorov, “Stochastic differential dynamic programming,” in *Proceedings of the 2010 American Control Conference*, pp. 1125–1132, IEEE, 2010.

[66] N. S. Kaisare, J. M. Lee, and J. H. Lee, “Simulation based strategy for nonlinear optimal control: Application to a microbial cell reactor,” *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, vol. 13, no. 3-4, pp. 347–363, 2003.

[67] J. M. Lee and J. H. Lee, “Approximate dynamic programming-based approaches for input–output data-driven control of nonlinear processes,” *Automatica*, vol. 41, no. 7, pp. 1281–1288, 2005.

[68] J. M. Lee, N. S. Kaisare, and J. H. Lee, “Choice of approximator and design of penalty function for an approximate dynamic programming based control approach,” *Journal of Process Control*, vol. 16, no. 2, pp. 135–156, 2006.

[69] J. H. Lee and J. M. Lee, “Approximate dynamic programming based approach to process control and scheduling,” *Computers & chemical engineering*, vol. 30, no. 10-12, pp. 1603–1618, 2006.

[70] J. M. Lee and J. H. Lee, “An approximate dynamic programming based approach to dual adaptive control,” *Journal of process control*, vol. 19, no. 5, pp. 859–864, 2009.

[71] H. Nosair, Y. Yang, and J. M. Lee, “Min–max control using parametric approximate dynamic programming,” *Control Engineering Practice*, vol. 18, no. 2, pp. 190–197, 2010.

[72] Y. Yang and J. M. Lee, “Probabilistic modeling and dynamic optimization for performance improvement and risk management of

plant-wide operation," *Computers & Chemical Engineering*, vol. 34, no. 4, pp. 567–579, 2010.

[73] Y. Yang and J. M. Lee, "A switching robust model predictive control approach for nonlinear systems," *Journal of Process Control*, vol. 23, no. 6, pp. 852–860, 2013.

[74] E. Ikonen, I. Selek, and K. Najim, "Process control using finite markov chains with iterative clustering," *Computers & Chemical Engineering*, vol. 93, pp. 293–308, 2016.

[75] Y. Cui, L. Zhu, M. Fujisaki, H. Kanokogi, and T. Matsubara, "Factorial kernel dynamic policy programming for vinyl acetate monomer plant model control," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pp. 304–309, IEEE, 2018.

[76] B. Sun, M. He, Y. Wang, W. Gui, C. Yang, and Q. Zhu, "A data-driven optimal control approach for solution purification process," *Journal of Process Control*, vol. 68, pp. 171–185, 2018.

[77] Y. Ge, S. Li, and P. Chang, "An approximate dynamic programming method for the optimal control of alkai-surfactant-polymer flooding," *Journal of Process Control*, vol. 64, pp. 15–26, 2018.

[78] J. W. Kim, G. B. Choi, and J. M. Lee, "A pomdp framework for integrated scheduling of infrastructure maintenance and inspection," *Computers & Chemical Engineering*, vol. 112, pp. 239–252, 2018.

[79] J. W. Kim, B. J. Park, H. Yoo, J. H. Lee, and J. M. Lee, "Deep reinforcement learning based finite-horizon optimal tracking control for nonlinear system," *IFAC-PapersOnLine*, vol. 51, no. 25, pp. 257–262, 2018.

[80] R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable Markov processes over a finite horizon," *Operations Research*, vol. 21, no. 5, pp. 1071–1088, 1973.

[81] M. Hauskrecht, "Value-function approximations for partially observable Markov decision processes," *Journal of Artificial Intelligence Research*, vol. 13, pp. 33–94, 2000.

[82] B. Øksendal, *Stochastic Differential Equations: An Introduction with Applications*. Berlin, Heidelberg: Springer-Verlag, 3 ed., 1992.

[83] J. Pineau, G. Gordon, S. Thrun, *et al.*, "Point-based value iteration: An anytime algorithm for POMDPs," in *International Joint Conference on Artificial Intelligence*, vol. 3, pp. 1025–1032, 2003.

[84] J. W. Kim, G. B. Choi, J. C. Suh, and J. M. Lee, "Dynamic optimization of maintenance and improvement planning for water main system: Periodic replacement approach," *Korean Journal of Chemical Engineering*, vol. 33, no. 1, pp. 25–32, 2016.

[85] A. W. Herrmann, "ASCE 2013 report card for America's infrastructure," in *IABSE Symposium Report*, vol. 99, pp. 9–10, International Association for Bridge and Structural Engineering, 2013.

[86] F. Flammini, A. Gaglione, F. Ottello, A. Pappalardo, C. Pragliola, and A. Tedesco, "Towards wireless sensor networks for railway infrastructure monitoring," in *Electrical Systems for Aircraft, Railway and Ship Propulsion (ESARS), 2010*, pp. 1–6, IEEE, 2010.

[87] I. Jawhar, N. Mohamed, and K. Shuaib, "A framework for pipeline infrastructure monitoring using wireless sensor networks," in *Wireless Telecommunications Symposium, 2007. WTS 2007*, pp. 1–7, IEEE, 2007.

[88] G. B. Choi, J. W. Kim, J. C. Suh, K. H. Jang, and J. M. Lee, "A prioritization method for replacement of water mains using rank aggregation," *Korean Journal of Chemical Engineering*, pp. 1–7, 2017.

[89] M. Memarzadeh and M. Pozzi, "Integrated inspection scheduling and maintenance planning for infrastructure systems," *Computer-Aided*

Civil and Infrastructure Engineering, vol. 31, no. 6, pp. 403–415, 2016.

[90] S. Madanat, R. Mishalani, and W. H. W. Ibrahim, "Estimation of infrastructure transition probabilities from condition rating data," *Journal of Infrastructure Systems*, vol. 1, no. 2, pp. 120–125, 1995.

[91] K. Golabi, R. B. Kulkarni, and G. B. Way, "A statewide pavement management system," *Interfaces*, vol. 12, no. 6, pp. 5–21, 1982.

[92] F. Guignier and S. Madanat, "Optimization of infrastructure systems maintenance and improvement policies," *Journal of Infrastructure Systems*, vol. 5, no. 4, pp. 124–134, 1999.

[93] S. Madanat and M. Ben-Akiva, "Optimal inspection and repair policies for infrastructure facilities," *Transportation Science*, vol. 28, no. 1, pp. 55–62, 1994.

[94] H. Ellis, M. Jiang, and R. B. Corotis, "Inspection, maintenance, and repair with partial observability," *Journal of Infrastructure Systems*, vol. 1, no. 2, pp. 92–99, 1995.

[95] M. Jiang, R. B. Corotis, and J. H. Ellis, "Optimal life-cycle costing with partial observability," *Journal of Infrastructure Systems*, vol. 6, no. 2, pp. 56–66, 2000.

[96] E. Byon and Y. Ding, "Season-dependent condition-based maintenance for a wind turbine using a partially observed Markov decision process," *IEEE Transactions on Power Systems*, vol. 25, no. 4, pp. 1823–1834, 2010.

[97] Y. Kleiner, "Scheduling inspection and renewal of large infrastructure assets," *Journal of Infrastructure Systems*, vol. 7, no. 4, pp. 136–143, 2001.

[98] C.-A. Robelin and S. M. Madanat, "History-dependent bridge deck maintenance and replacement optimization with Markov decision

processes," *Journal of Infrastructure Systems*, vol. 13, no. 3, pp. 195–201, 2007.

[99] K. Papakonstantinou and M. Shinozuka, "Optimum inspection and maintenance policies for corroded structures using partially observable Markov decision processes and stochastic, physically based models," *Probabilistic Engineering Mechanics*, vol. 37, pp. 93–108, 2014.

[100] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Autonomous Agents and Multi-Agent Systems*, pp. 1–51, 2013.

[101] M. T. Spaan and N. Vlassis, "Perseus: Randomized point-based value iteration for POMDPs," *Journal of Artificial Intelligence Research*, vol. 24, pp. 195–220, 2005.

[102] T. Smith and R. Simmons, "Heuristic search value iteration for POMDPs," in *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pp. 520–527, AUAI Press, 2004.

[103] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *Robotics: Science and Systems*, vol. 2008, Zurich, Switzerland., 2008.

[104] G. Shani, P. Poupart, R. I. Brafman, and S. E. Shimony, "Efficient ADD operations for point-based algorithms.," in *International Conference on Automated Planning and Scheduling*, pp. 330–337, 2008.

[105] P. Poupart, K.-E. Kim, and D. Kim, "Closing the gap: Improved bounds on optimal POMDP sSolutions.," in *International Conference on Automated Planning and Scheduling*, pp. 194–201, 2011.

[106] M. Wang and R. Dearden, "Run-time improvement of point-based POMDP policies.," in *International Joint Conference on Artificial Intelligence*, pp. 2408–2414, 2013.

[107] G. Rens and A. Ferrein, “Belief-node condensation for online POMDP algorithms,” in *AFRICON, 2013*, pp. 1–5, IEEE, 2013.

[108] H.-S. Baik, H. S. Jeong, and D. M. Abraham, “Estimating transition probabilities in Markov chain-based deterioration models for management of wastewater systems,” *Journal of Water Resources Planning and Management*, vol. 132, no. 1, pp. 15–24, 2006.

[109] K. Papakonstantinou and M. Shinozuka, “Probabilistic model for steel corrosion in reinforced concrete structures of large dimensions considering crack effects,” *Engineering Structures*, vol. 57, pp. 306–326, 2013.

[110] G. S. Park and E. S. Park, “Improvement of the sensor system in magnetic flux leakage-type nondestructive testing (NDT),” *IEEE Transactions on Magnetics*, vol. 38, no. 2, pp. 1277–1280, 2002.

[111] C. Fan, F. Sun, and L. Yang, “Investigation on nondestructive evaluation of pipelines using infrared thermography,” in *Infrared and Millimeter Waves and 13th International Conference on Terahertz Electronics, 2005. IRMMW-THz 2005. The Joint 30th International Conference*, vol. 2, pp. 339–340, IEEE, 2005.

[112] J. A. Liggett and L.-C. Chen, “Inverse transient analysis in pipe networks,” *Journal of Hydraulic Engineering*, vol. 120, no. 8, pp. 934–955, 1994.

[113] P. J. Shull, *Nondestructive Evaluation: Theory, Techniques, and Applications*. CRC press, 2016.

[114] S. A. Andreou, D. H. Marks, and R. M. Clark, “A new methodology for modelling break failure patterns in deteriorating water distribution systems: Theory,” *Advances in Water Resources*, vol. 10, no. 1, pp. 2–10, 1987.

[115] A. Martins, J. P. Leitão, and C. Amado, “Comparative study of three stochastic models for prediction of pipe failures in water supply sys-

tems," *Journal of Infrastructure Systems*, vol. 19, no. 4, pp. 442–450, 2013.

[116] S. A. Andreou, D. H. Marks, and R. M. Clark, "A new methodology for modelling break failure patterns in deteriorating water distribution systems: Theory," *Advances in Water Resources*, vol. 10, no. 1, pp. 2–10, 1987.

[117] M. A. Cesare, C. Santamarina, C. Turkstra, and E. H. Vanmarcke, "Modeling bridge deterioration with Markov chains," *Journal of Transportation Engineering*, vol. 118, no. 6, pp. 820–833, 1992.

[118] S. Madanat and W. H. W. Ibrahim, "Poisson regression models of infrastructure transition probabilities," *Journal of Transportation Engineering*, vol. 121, no. 3, pp. 267–272, 1995.

[119] The Office of Waterworks Seoul Metropolitan Government, "Basic breakdown cost table for the waterworks construction," September 2010.

[120] J. W. Kim, B. J. Park, H. Yoo, J. H. Lee, and J. M. Lee, "A model-based deep reinforcement learning method applied to finite-horizon optimal control of nonlinear control-affine system," *Journal of Process Control*. Under review.

[121] D. Adhyaru, I. Kar, and M. Gopal, "Fixed final time optimal control approach for bounded robust controller design using Hamilton—Jacobi-–Bellman solution," *IET Control Theory and Applications*, vol. 3, no. 9, pp. 1183–1195, 2009.

[122] H. Zhang, L. Cui, X. Zhang, and Y. Luo, "Data–driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 2226–2236, 2011.

[123] D. Wang, D. Liu, and Q. Wei, "Finite–horizon neuro-optimal tracking control for a class of discrete–time nonlinear systems using adaptive

dynamic programming approach," *Neurocomputing*, vol. 78, no. 1, pp. 14–22, 2012.

[124] Q. Zhao, H. Xu, and S. Jagannathan, "Neural network–based finite–horizon optimal control of uncertain affine nonlinear discrete-time systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 3, pp. 486–499, 2015.

[125] C. Mu, D. Wang, and H. He, "Data–driven finite-horizon approximate optimal control for discrete-time nonlinear systems using iterative HDP approach," *IEEE Transactions on Cybernetics*, vol. 48, no. 10, pp. 2948–2961, 2018.

[126] C. Mu, Z. Ni, C. Sun, and H. He, "Data–driven tracking control with adaptive dynamic programming for a class of continuous-time nonlinear systems," *IEEE Transactions on Cybernetics*, vol. 47, no. 6, pp. 1460–1470, 2017.

[127] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[128] K. G. Vamvoudakis and F. L. Lewis, "Online actor—critic algorithm to solve the continuous-time infinite horizon optimal control problem," *Automatica*, vol. 46, no. 5, pp. 878–888, 2010.

[129] A. Heydari and S. N. Balakrishnan, "Finite–horizon control-constrained nonlinear optimal control using single network adaptive critics," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 1, pp. 145–157, 2013.

[130] C. Mu, D. Wang, and H. He, "Novel iterative neural dynamic programming for data-based approximate optimal control design," *Automatica*, vol. 81, pp. 240–252, 2017.

[131] T. Cheng, F. L. Lewis, and M. Abu-Khalaf, "Fixed–final–time–constrained optimal control of nonlinear systems using neural net-

work HJB approach," *IEEE Transactions on Neural Networks*, vol. 18, no. 6, pp. 1725–1737, 2007.

[132] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[133] Y. H. Cheng, B. Jiang, H. Li, and X. D. Han, "On-orbit Reconfiguration Using Adaptive Dynamic Programming for Multi-mission-constrained Spacecraft Attitude Control System," *International Journal of Control, Automation and Systems*, vol. 17, no. 4, pp. 822–835, 2019.

[134] J. H. Lee, J. Shin, and M. J. Realff, "Machine learning: Overview of the recent progresses and implications for the process systems engineering field," *Computers & Chemical Engineering*, vol. 114, pp. 111 – 121, 2018.

[135] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[136] L. Buşoniu, T. de Bruin, D. Tolić, J. Kober, and I. Palunko, "Reinforcement learning for control: Performance, stability, and deep approximators," *Annual Reviews in Control*, vol. 46, pp. 8–28, 2018.

[137] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[138] D. P. Bertsekas, "Feature-based aggregation and deep reinforcement learning: A survey and some new implementations," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 1, pp. 1–31, 2019.

[139] D. Liu, Q. Wei, D. Wang, X. Yang, and H. Li, *Adaptive dynamic programming with applications in optimal control.* Springer, 2017.

[140] D. Liu, D. Wang, D. Zhao, Q. Wei, and N. Jin, "Neural–network–based optimal control for a class of unknown discrete-time nonlinear systems using globalized dual heuristic programming," *IEEE*

Transactions on Automation Science and Engineering, vol. 9, no. 3, pp. 628–634, 2012.

[141] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, pp. 1928–1937, 2016.

[142] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[143] H. Van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning.," in *AAAI*, vol. 2, p. 5, Phoenix, AZ, 2016.

[144] Y. Jiang and Z. P. Jiang, "Robust adaptive dynamic programming and feedback stabilization of nonlinear systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 882–893, 2014.

[145] J. R. Dormand and P. J. Prince, "A family of embedded Runge-Kutta formulae," *Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 19–26, 1980.

[146] L. F. Shampine and M. W. Reichelt, "The MATLAB ODE suite," *SIAM Journal on Scientific Computing*, vol. 18, no. 1, pp. 1–22, 1997.

[147] I. S. Chin, K. S. Lee, and J. H. Lee, "A technique for integrated quality control, profile control, and constraint handling for batch processes," *Industrial & engineering chemistry research*, vol. 39, no. 3, pp. 693–705, 2000.

[148] A. Armaou and P. D. Christofides, "Dynamic optimization of dissipative PDE systems using nonlinear order reduction," *Chemical Engineering Science*, vol. 57, no. 24, pp. 5083–5114, 2002.

[149] B. Luo, H. N. Wu, and H. X. Li, “Adaptive optimal control of highly dissipative nonlinear spatially distributed processes with neuro-dynamic programming.,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 4, pp. 684–696, 2015.

[150] J. W. Kim, T. H. Oh, S. H. Song, D. W. Jeong, and J. M. Lee, “Convergence analysis of the model-based deep reinforcement learning for optimal control of nonlinear control-affine system,” *Automatica*. Under review.

[151] M. Abu-Khalaf and F. L. Lewis, “Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network HJB approach,” *Automatica*, vol. 41, no. 5, pp. 779–791, 2005.

[152] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, “Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 4, pp. 943–949, 2008.

[153] B. Lincoln and A. Rantzer, “Relaxing dynamic programming,” *IEEE Transactions on Automatic Control*, vol. 51, no. 8, pp. 1249–1260, 2006.

[154] X. Liu and S. Balakrishnan, “Convergence analysis of adaptive critic based optimal control,” in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*, vol. 3, pp. 1929–1933, IEEE, 2000.

[155] D. Adhyaru, I. Kar, and M. Gopal, “Fixed final time optimal control approach for bounded robust controller design using Hamilton-Jacobi-Bellman solution,” *IET control theory & applications*, vol. 3, no. 9, pp. 1183–1195, 2009.

[156] K. G. Vamvoudakis and F. L. Lewis, “Online actor–critic algorithm to solve the continuous-time infinite horizon optimal control problem,” *Automatica*, vol. 46, no. 5, pp. 878–888, 2010.

[157] A. Heydari and S. N. Balakrishnan, “Finite-horizon control-constrained nonlinear optimal control using single network adaptive critics,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 1, pp. 145–157, 2013.

[158] S. Bhasin, R. Kamalapurkar, M. Johnson, K. G. Vamvoudakis, F. L. Lewis, and W. E. Dixon, “A novel actor–critic–identifier architecture for approximate optimal control of uncertain nonlinear systems,” *Automatica*, vol. 49, no. 1, pp. 82–92, 2013.

[159] Q. Zhao, H. Xu, and S. Jagannathan, “Neural network-based finite-horizon optimal control of uncertain affine nonlinear discrete-time systems,” *IEEE transactions on neural networks and learning systems*, vol. 26, no. 3, pp. 486–499, 2015.

[160] H. Zhang, L. Cui, X. Zhang, and Y. Luo, “Data-driven robust approximate optimal tracking control for unknown general nonlinear systems using adaptive dynamic programming method,” *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 2226–2236, 2011.

[161] T. Poggio, K. Kawaguchi, Q. Liao, B. Miranda, L. Rosasco, X. Boix, J. Hidary, and H. Mhaskar, “Theory of Deep Learning III: explaining the non-overfitting puzzle,” *arXiv preprint arXiv:1801.00173*, 2017.

[162] C. Mu, D. Wang, and H. He, “Novel iterative neural dynamic programming for data-based approximate optimal control design,” *Automatica*, vol. 81, pp. 240–252, 2017.

[163] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, “Why and when can deep-but not shallow-networks avoid the curse of dimensionality: a review,” *International Journal of Automation and Computing*, vol. 14, no. 5, pp. 503–519, 2017.

[164] J. Gallier, “The schur complement and symmetric positive semidefinite (and definite) matrices,” *Penn Engineering*, 2010.

[165] L. Petzold, J. B. Rosen, P. E. Gill, L. O. Jay, and K. Park, *Numerical Optimal Control of Parabolic PDES Using DASOPT*, pp. 271–299. New York, NY: Springer New York, 1997.

[166] L. T. Biegler, "An overview of simultaneous strategies for dynamic optimization," *Chemical Engineering and Processing: Process Intensification*, vol. 46, no. 11, pp. 1043–1053, 2007.

[167] D. H. Jacobson and D. Q. Mayne, "Differential dynamic programming," 1970.

[168] D. Murray and S. Yakowitz, "Differential dynamic programming and newton's method for discrete optimal control problems," *Journal of Optimization Theory and Applications*, vol. 43, no. 3, pp. 395–414, 1984.

[169] L.-Z. Liao and C. A. Shoemaker, "Convergence in unconstrained discrete-time differential dynamic programming," *IEEE Transactions on Automatic Control*, vol. 36, no. 6, pp. 692–706, 1991.

[170] L.-z. Liao and C. A. Shoemaker, "Advantages of differential dynamic programming over newton's method for discrete-time optimal control problems," tech. rep., Cornell University, 1992.

[171] 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.*, pp. 300–306, IEEE, 2005.

[172] Y. Pan and E. Theodorou, "Probabilistic differential dynamic programming," in *Advances in Neural Information Processing Systems*, pp. 1907–1915, 2014.

[173] G. I. Boutselis, Y. Pan, G. De La Tore, and E. A. Theodorou, "Stochastic trajectory optimization for mechanical systems with parametric uncertainties," *arXiv preprint arXiv:1705.05506*, 2017.

[174] W. Sun, Y. Pan, J. Lim, E. A. Theodorou, and P. Tsiotras, “Min-max differential dynamic programming: Continuous and discrete time formulations,” *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 12, pp. 2568–2580, 2018.

[175] W. Sun, E. A. Theodorou, and P. Tsiotras, “Continuous-time differential dynamic programming with terminal constraints,” in *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 1–6, IEEE, 2014.

[176] T. Lin and J. Arora, “Differential dynamic programming technique for constrained optimal control,” *Computational Mechanics*, vol. 9, no. 1, pp. 27–40, 1991.

[177] Z. Xie, C. K. Liu, and K. Hauser, “Differential dynamic programming with nonlinear constraints,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 695–702, IEEE, 2017.

[178] G. Lantoine and R. P. Russell, “A hybrid differential dynamic programming algorithm for constrained optimal control problems. part 1: theory,” *Journal of Optimization Theory and Applications*, vol. 154, no. 2, pp. 382–417, 2012.

[179] D. J. Ruxton, “Differential dynamic programming applied to continuous optimal control problems with state variable inequality constraints,” *Dynamics and Control*, vol. 3, no. 2, pp. 175–185, 1993.

[180] E. Pellegrini and R. P. Russell, “Applications of the multiple-shooting differential dynamic programming algorithm with path and terminal constraints,” in *AAS/AIAA Astrodynamics Specialist Conference*, 2017.

[181] B. Plancher, Z. Manchester, and S. Kuindersma, “Constrained unscented dynamic programming,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5674–5680, IEEE, 2017.

[182] N.-Y. Chiang, R. Huang, and V. M. Zavala, "An augmented lagrangian filter method for real-time embedded optimization," *IEEE Transactions on Automatic Control*, vol. 62, no. 12, pp. 6110–6121, 2017.

[183] R. T. Rockafellar, "Augmented lagrangians and applications of the proximal point algorithm in convex programming," *Mathematics of operations research*, vol. 1, no. 2, pp. 97–116, 1976.

[184] E. G. Birgin and J. M. Martinez, *Practical augmented Lagrangian methods for constrained optimization*, vol. 10. SIAM, 2014.

[185] E. Pellegrini and R. P. Russell, "A multiple-shooting differential dynamic programming algorithm," in *AAS/AIAA Space Flight Mechanics Meeting*, vol. 2, 2017.

[186] A. R. Conn, N. I. Gould, and P. Toint, "A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds," *SIAM Journal on Numerical Analysis*, vol. 28, no. 2, pp. 545–572, 1991.

[187] J. H. Lee, J. Shin, and M. J. Realff, "Machine learning: Overview of the recent progresses and implications for the process systems engineering field," *Computers & Chemical Engineering*, vol. 114, pp. 111–121, 2018.

[188] B. Øksendal, "Stochastic differential equations," in *Stochastic differential equations*, pp. 65–84, Springer, 2003.

[189] G. Williams, A. Aldrich, and E. A. Theodorou, "Model predictive path integral control: From theory to parallel computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.

[190] S. Levine and V. Koltun, "Learning complex neural network policies with trajectory optimization," in *International Conference on Machine Learning*, pp. 829–837, 2014.

[191] E. Theodorou, "Nonlinear stochastic control and information theoretic dualities: Connections, interdependencies and thermodynamic interpretations," *Entropy*, vol. 17, no. 5, pp. 3352–3375, 2015.

[192] J. Sjöberg, *Optimal control and model reduction of nonlinear DAE models*. PhD thesis, Institutionen för systemteknik, 2008.

[193] C. T. Maravelias and C. Sung, "Integration of production planning and scheduling: Overview, challenges and opportunities," *Computers & Chemical Engineering*, vol. 33, no. 12, pp. 1919–1930, 2009.

[194] V. M. Charitopoulos, L. G. Papageorgiou, and V. Dua, "Closed-loop integration of planning, scheduling and multi-parametric nonlinear control," *Computers & Chemical Engineering*, vol. 122, pp. 172–192, 2019.

[195] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search (2015)," *arXiv preprint arXiv:1501.05611*, 2015.

[196] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, "Plan online, learn offline: Efficient learning and exploration via model-based control," *arXiv preprint arXiv:1811.01848*, 2018.

[197] W. Sun, J. A. Bagnell, and B. Boots, "Truncated horizon policy search: Combining reinforcement learning & imitation learning," *arXiv preprint arXiv:1805.11240*, 2018.

[198] V. Feinberg, A. Wan, I. Stoica, M. Jordan, J. Gonzalez, and S. Levine, "Model-based value expansion for efficient model-free reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, 2018.

[199] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine, "Combining model-based and model-free updates for trajectory-centric reinforcement learning," in *Proceedings of the 34th*

International Conference on Machine Learning-Volume 70, pp. 703–711, JMLR. org, 2017.

초 록

순차적 의사결정 문제는 공정 최적화의 핵심 분야 중 하나이다. 이 문제의 수치적 해법 중 가장 많이 사용되는 것은 순방향으로 작동하는 직접법 (direct optimization) 방법이지만, 몇가지 한계점을 지니고 있다. 최적해는 open-loop의 형태를 지니고 있으며, 불확정성이 존재할때 방법론의 수치적 복잡도가 증가한다는 것이다. 동적 계획법 (dynamic programming) 은 이러한 한계점을 근원적으로 해결할 수 있지만, 그동안 공정 최적화에 적극적으로 고려되지 않았던 이유는 동적 계획법의 결과로 얻어진 편미분 방정식 문제가 유한차원 벡터공간이 아닌 무한차원의 함수공간에서 다루어지기 때문이다. 소위 차원의 저주라고 불리는 이 문제를 해결하기 위한 한가지 방법으로서, 샘플을 이용한 근사적 해법에 초점을 둔 강화학습 방법론이 연구되어 왔다. 본 학위논문에서는 강화학습 방법론 중, 공정 최적화에 적합한 모델 기반 강화학습에 대해 연구하고, 이를 공정 최적화의 대표적인 세가지 순차적 의사결정 문제인 스케줄링, 상위단계 최적화, 하위단계 제어에 적용하는 것을 목표로 한다. 이 문제들은 각각 부분관측 마르코프 결정 과정 (partially observable Markov decision process), 제어-아핀 상태공간 모델 (control-affine state space model), 일반적 상태공간 모델 (general state space model)로 모델링된다. 또한 각 수치적 모델들을 해결하기 위해 point based value iteration (PBVI), globalized dual heuristic programming (GDHP), and differential dynamic programming (DDP)로 불리는 방법들을 도입하였다.

이 세가지 문제와 방법론에서 제시된 특징들을 다음과 같이 요약할 수 있다: 첫번째로, 스케줄링 문제에서 closed-loop 피드백 형태의 해를 제시할 수 있었다. 이는 기존 직접법에서 얻을 수 없었던

형태로서, 강화학습의 강점을 부각할 수 있는 측면이라 생각할 수 있다. 두번째로 고려한 하위단계 제어 문제에서, 동적 계획법의 무한차원 함수공간 최적화 문제를 함수 근사 방법을 통해 유한차원 벡터공간 최적화 문제로 완화할 수 있는 방법을 도입하였다. 특히, 심층 신경망을 이용하여 함수 근사를 하였고, 이때 발생하는 여러가지 장점과 수렴 해석 결과를 본 학위논문에 실었다. 마지막 문제는 상위 단계 동적 최적화 문제이다. 동적 최적화 문제에서 발생하는 제약 조건하에서 강화학습을 수행하기 위해, 원-쌍대 미분동적 계획법 (primal-dual DDP) 방법론을 새로 제안하였다. 앞서 설명한 세 가지 문제에 적용된 방법론을 검증하고, 동적 계획법이 직접법에 비견될 수 있는 방법론이라는 주장을 실증하기 위해 여러가지 공정 예제를 실었다.

주요어 : 강화 학습, 최적 제어, 동적 최적화, 스케줄링, 딥러닝, 부분관측 마르코프 의사결정, 미분 동적 계획법

학번 : 2014-21532