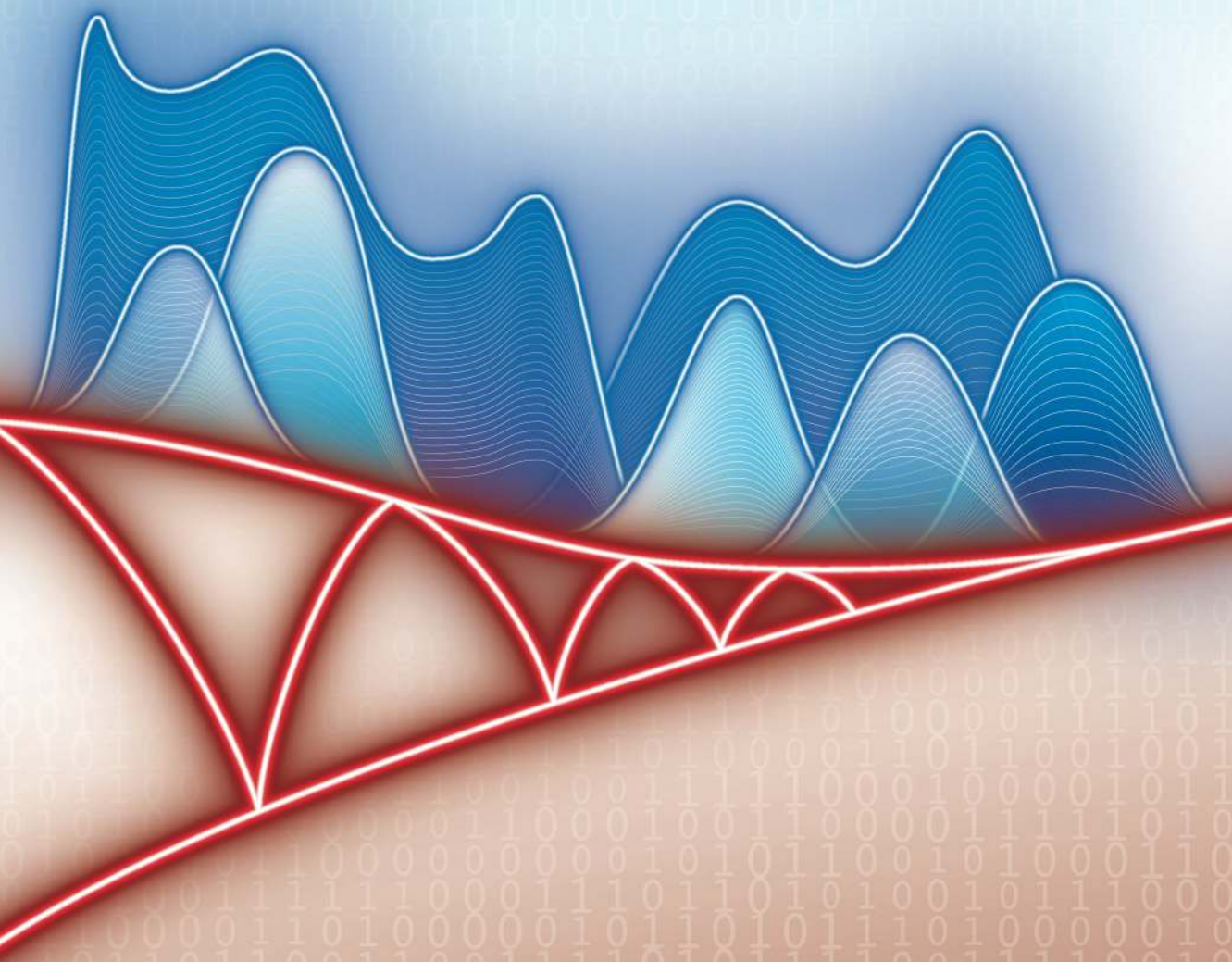
Policy Iteration with Limited Data and Partial Information



Roy van Zuijlen

Policy Iteration with Limited Data and Partial Information

Roy Albertus Cornelis van Zuijlen

The research is carried out as part of the ITEA4 20216 ASIMOV project. The ASIMOV activities are supported by the Netherlands Organisation for Applied Scientific Research TNO and the Dutch Ministry of Economic Affairs and Climate (project number: AI211006). The research leading to these results is partially funded by the German Federal Ministry of Education and Research (BMBF) within the project ASIMOV-D under grant agreement No. 01IS21022G [DLR], based on a decision of the German Bundestag.

The work described in this thesis was carried out at the Eindhoven University of Technology.

A catalogue record is available from the Eindhoven University of Technology Library.
ISBN: 978-90-386-6700-3

Typeset by the author using the pdf LaTeX documentation system.
Cover design: Roy van Zuijlen
Reproduction: Ipskamp Printing, Enschede, the Netherlands

Policy Iteration with Limited Data and Partial Information

Proefschrift

ter verkrijging van de graad van doctor aan de Technische Universiteit Eindhoven, op gezag van de rector magnificus prof.dr. S.K. Lenaerts, voor een commissie aangewezen door het College voor Promoties, in het openbaar te verdedigen op dinsdag 2 juni 2026 om 16:00 uur

door

Roy Albertus Cornelis van Zuijlen

geboren te Nijmegen

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	prof.dr.ir. M.J.G. van de Molengraft	
1e promotor:	dr. D.J. Guerreiro Tomé Antunes	
2e promotor:	prof.dr.ir. W.P.M.H. Heemels	
leden:	prof.dr. R.M. Jungers	(Université catholique de Louvain)
	prof.dr.ir. B. De Schutter	(Technische Universiteit Delft)
	prof.dr.ir. N. Meratnia	
	dr.ir. M. Schoukens	

Het onderzoek dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

Summary

Policy Iteration with Limited Data and Partial Information

Modern engineering systems, e.g. in robotics and manufacturing, often operate under partial observability: decisions must be made from indirect, noisy, and incomplete measurements rather than full knowledge of the system state. Achieving acceptable performance using direct output-feedback policies and systematically improving these policies is challenging: performance changes may stem from noise, estimation errors, or limited data, and experimental results may not generalize beyond the observed trajectories. This thesis studies how control policies can be improved and evaluated under partial information with limited data. The central challenge is to make reliable, data-driven decisions when observations are indirect. To address this challenge, methods are developed for policy iteration, enabling principled improvement under partial observability.

The first part of the thesis addresses policy iteration when full state information is unavailable and control decisions must be based solely on measured outputs. This situation arises naturally in engineering systems due to sensing limitations, noise, or high-dimensional state spaces. Classical approaches rely on belief-state representations, which are often computationally intractable. This part develops policy iteration methods that operate directly in the output space using memoryless or static output-feedback policies. The considered systems are partially observable Markov decision processes, covering both discrete and continuous spaces, including linear-quadratic systems with Gaussian noise. By carefully structuring the policy improvement and evaluation steps, it is shown that monotonic performance improvement can be achieved despite the non-Markovian nature of the output process. An optimal structure for alternating improvement and evaluation steps with the least amount of computations is identified, leading to substantial gains in computational efficiency. Furthermore, model-free variants are introduced that estimate performance measures directly from data, enabling learning without explicit system models. The resulting framework is applied to classical output-feedback

control problems, demonstrating that reinforcement learning-based policy iteration provides a viable alternative to traditional analytical and gradient-based methods. This connection highlights how modern learning algorithms can be used to design output-feedback controllers for systems commonly encountered in engineering.

The second part of the thesis focuses on policy evaluation for fully observable Markov decision processes, with an emphasis on data efficiency. Policy evaluation is a key component in the aforementioned policy iteration algorithms. Classical methods often require large data sets and fail to exploit prior knowledge that is frequently available in engineering applications. The thesis develops least-squares temporal-difference methods that explicitly connect data-driven value estimation to underlying probabilistic models of the system dynamics. By interpreting least-squares policy evaluation as a model-based procedure, prior information about system behaviour can be incorporated through Bayesian principles, leading to significantly improved convergence and reduced estimation error when data are limited. In addition, the role of eligibility traces in least-squares temporal-difference learning is analysed in depth. The thesis shows that, unlike in direct temporal-difference methods, extensions using expected or mixed eligibility traces do not fundamentally improve least-squares value-function estimates, and are equivalent to simpler formulations. Together, these results clarify the theoretical foundations of data-efficient policy evaluation and provide practical guidance for applying reinforcement learning methods in applications where experimental data are expensive to obtain and/or simulators or digital twins are available.

The third part of the thesis focuses on state estimation for dynamical systems with noisy, nonlinear, and possibly high-dimensional outputs, addressing scenarios where decisions based solely on measured outputs are insufficient. Accurate state estimates are essential for feedback control, measurement, and monitoring tasks, but exact Bayesian methods are often impractical due to the curse of dimensionality. This part develops Bayesian estimation techniques that reduce computational complexity while retaining probabilistic rigour. In the context of vision-based pose estimation, Bayesian filtering methods are combined with Gaussian process regression and deep learning to construct reliable likelihood models from image data. These approaches are validated in both simulated and industrial environments, illustrating their effectiveness in automated metrology and manufacturing applications. In addition, a frequency-domain perspective on Bayesian filtering is introduced for linear systems with nonlinear outputs and general noise distributions. By representing probability densities using Fourier basis functions, exact Bayesian estimation can be carried out in a countable coefficient space, leading to tractable approximation schemes with controlled complexity growth. The applicability of this approach is demonstrated in the context of electron microscopy, showing that accurate state estimation can be achieved without prohibitive computational cost.

Together, the three parts of this thesis develop methods for conducting policy

iteration with limited data availability and partial information, addressing both the evaluation and improvement of policies as well as the estimation of hidden system states.

Keywords: reinforcement learning, policy iteration, output-feedback control, data-efficient control, Bayesian state estimation

Samenvatting

Moderne technologische systemen, bijvoorbeeld in robotica en productie, opereren vaak onder gedeeltelijke observeerbaarheid: beslissingen moeten worden genomen op basis van indirecte, ruisgevoelige metingen in plaats van volledige kennis van de systeemtoestand. Het behalen van acceptabele prestaties met behulp van regelstrategiën waarbij gemeten uitgangen direct worden teruggekoppeld naar beslissingen en het systematisch verbeteren van deze regelstrategiën is daarom uitdagend: prestatieveranderingen kunnen voortkomen uit ruis, schattingsfouten of beperkte data, en experimentele resultaten generaliseren mogelijk niet buiten de geobserveerde trajecten. Dit proefschrift bestudeert hoe regelstrategieën kunnen worden verbeterd en geëvalueerd onder gedeeltelijke informatie en met beperkte data. De voornaamste uitdaging is het nemen van betrouwbare, data-gedreven beslissingen wanneer observaties indirect zijn. Om deze uitdaging aan te pakken, worden methoden ontwikkeld voor beleidsiteratie, die principiële verbetering van deze regelstrategiën onder gedeeltelijke observeerbaarheid mogelijk maken.

Het eerste deel van het proefschrift behandelt beleidsiteratie wanneer de volledige systeemtoestand niet beschikbaar is en beslissingen uitsluitend gebaseerd moeten worden op gemeten uitgangen. Deze situatie ontstaat van nature in technische systemen door sensorbeperkingen, ruis of hoog-dimensionale toestandsruimten. Klassieke benaderingen maken gebruik van kansverdeling representaties van de systeemtoestand, die vaak computationeel onhanteerbaar zijn. In dit deel worden beleidsiteratie-methoden ontwikkeld die direct in de meetruimte opereren met geheugenloze of statische regelstrategiën waarbij gemeten uitgangen direct worden teruggekoppeld. De beschouwde systemen zijn partieel waarneembare Markov-beslissingsprocessen, die zowel discrete als continue toestands- en actieruimten omvatten, waaronder lineair-kwadratische systemen met Gaussiaanse verstoringen. Door de stappen voor beleidsverbetering en -evaluatie zorgvuldig te structureren, wordt aangetoond dat monotone prestatieverbetering kan worden bereikt ondanks het niet-Markovische karakter van het gemeten proces. Een optimale structuur voor afwisselende verbeterings- en evaluatiestappen met een minimale hoeveelheid berekeningen wordt geïdentificeerd, wat leidt tot aanzienlijke winst in computationele efficiëntie. Daarnaast worden modelvrije varianten geïntroduceerd die

prestatie-indicaties direct uit data schatten, waardoor leren zonder expliciete systeemmodellen mogelijk wordt. Het resulterende raamwerk wordt toegepast op klassieke regelproblemen waarbij gemeten uitgangen worden teruggekoppeld, en laat zien dat reinforcement learning-gebaseerde beleidsiteratie een levensvatbaar alternatief vormt voor traditionele analytische en gradiëntgebaseerde methoden. Deze verbinding benadrukt hoe moderne leeralgoritmen kunnen worden gebruikt voor het ontwerpen van regelaars voor systemen met gedeeltelijke observabiliteit, die veel voorkomen in de techniek.

Het tweede deel van het proefschrift richt zich op beleidsevaluatie voor volledig observeerbare Markov-beslissingsprocessen, met nadruk op data-efficiëntie. Beleidsevaluatie is een essentieel onderdeel van de bovengenoemde beleidsiteratie-algoritmen. Klassieke methoden vereisen vaak grote datasets en slagen er niet in om voorkennis, die in technische toepassingen vaak beschikbaar is, te benutten. Het proefschrift ontwikkelt kleinste-kwadraten temporal-difference methoden die data-gedreven waardeschatting expliciet verbinden met onderliggende probabilistische modellen van de systeemdynamica. Door kleinste-kwadraten beleidsevaluatie te interpreteren als een modelgebaseerde procedure, kan voorkennis over systeemgedrag via Bayesiaanse principes worden geïntegreerd, wat leidt tot aanzienlijk verbeterde convergentie en verminderde schattingsfouten wanneer data beperkt is. Daarnaast wordt de rol van eligibility traces in kleinste-kwadraten temporal-difference learning diepgaand geanalyseerd. Het proefschrift toont aan dat, in tegenstelling tot directe temporal-difference methoden, uitbreidingen met verwachte of gemengde eligibility traces geen fundamentele verbetering opleveren voor kleinste-kwadraten waardefunctieschattingen en equivalent zijn aan eenvoudigere formuleringen. Gezamenlijk verduidelijken deze resultaten de theoretische grondslagen van data-efficiënte beleidsevaluatie en bieden zij praktische richtlijnen voor het toepassen van reinforcement learning-methoden in toepassingen waar het verkrijgen van experimentele data kostbaar is en/of simulatoren of digitale tweelingen beschikbaar zijn.

Het derde deel van het proefschrift richt zich op toestandschatting voor dynamische systemen met ruisgevoelige, niet-lineaire en mogelijk hoog-dimensionale meetruimtes, en behandelt scenario's waarin beslissingen op basis van uitsluitend metingen onvoldoende zijn. Nauwkeurige toestandschattingen zijn essentieel voor terugkoppeling, metingen en monitoringtaken, maar exacte Bayesiaanse methoden zijn vaak onpraktisch vanwege de vloek van dimensionaliteit. In dit deel worden Bayesiaanse schattingstechnieken ontwikkeld die de computationele complexiteit reduceren, terwijl de probabilistische nauwkeurigheid behouden blijft. In de context van visie-gebaseerde oriëntatieschatting worden Bayesiaanse filtermethoden gecombineerd met Gaussian process regressie en deep learning om betrouwbare likelihood-modellen uit beelddata te construeren. Deze benaderingen worden gevalideerd in zowel gesimuleerde als industriële omgevingen, wat hun effectiviteit aantoont in geautomatiseerde metrologie- en productieprocessen. Daarnaast wordt

een frequentiedomeinperspectief op Bayesiaans filteren geïntroduceerd voor lineaire systemen met niet-lineaire metingen en algemene ruisverdelingen. Door kansdichtheden te representeren met behulp van Fourier-basisfuncties, kan exacte Bayesiaanse schatting worden uitgevoerd in een telbare coëfficiëntenruimte, wat leidt tot hanteerbare benaderingsschema's met gecontroleerde groei in complexiteit. De toepasbaarheid van deze benadering wordt aangetoond in de context van elektronenmicroscopie, waarbij een nauwkeurige toestandschatting kan worden bereikt zonder prohibitieve computationele kosten.

Gezamenlijk ontwikkelen de drie delen van dit proefschrift methoden voor het uitvoeren van beleidsiteratie onder beperkte databeschikbaarheid en gedeeltelijke informatie, waarbij zowel de evaluatie en verbetering van regelstrategiën als de schatting van systeemtoestanden worden behandeld.

Contents

I

Opening

Chapter 1

Introduction

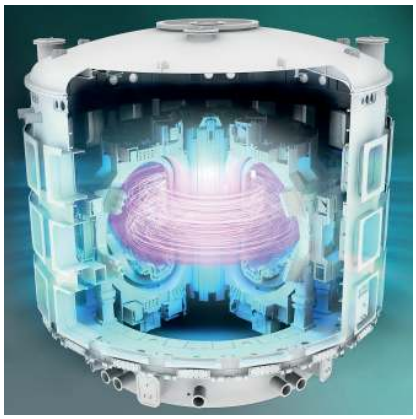
In many scientific and engineering problems, control decisions are needed to achieve desired objectives, such as reaching a target state or accomplishing a specified task. Control decisions can vary widely across applications, ranging from low-level actuator commands to higher-level choices such as mode switching or trajectory selection. In all these settings, a controller maps available information or measurements (e.g., sensor data) to control decisions. Such a mapping is commonly referred to as a control policy, which specifies how control decisions are selected based on the available information.

The design of policies is often guided by an optimality criterion. Optimality criteria provide a means to compare different policies by assigning a quantitative measure of their performance over time or over a task horizon. Such performance criteria may include tracking performance, energy consumption of the system, the time required to reach a desired position, and many others. In practice, these criteria often involve trade-offs, as improving one aspect of performance may degrade another. Moreover, the evaluation and design of control policies depend critically on what information about the system is available and how this information is exploited.

Figure 1.1 illustrates several examples of control applications, each characterized by distinct objectives and criteria for optimality. One example is the electron microscope, where a key control objective is to accurately and efficiently bring the image into focus by adjusting aberration correctors, to maximize image quality and system throughput. A second example arises in vision-based robotic manipulation and industrial metrology, where control decisions depend on estimates of an object's pose obtained from camera images. In such applications, control objectives may include accurate positioning or inspection, while performance depends on the quality of the underlying state estimates. A further example arises in nuclear

(a) Electron microscope [119].

(b) Coordinate measurement machine.



(c) Nuclear fusion reactor [120].

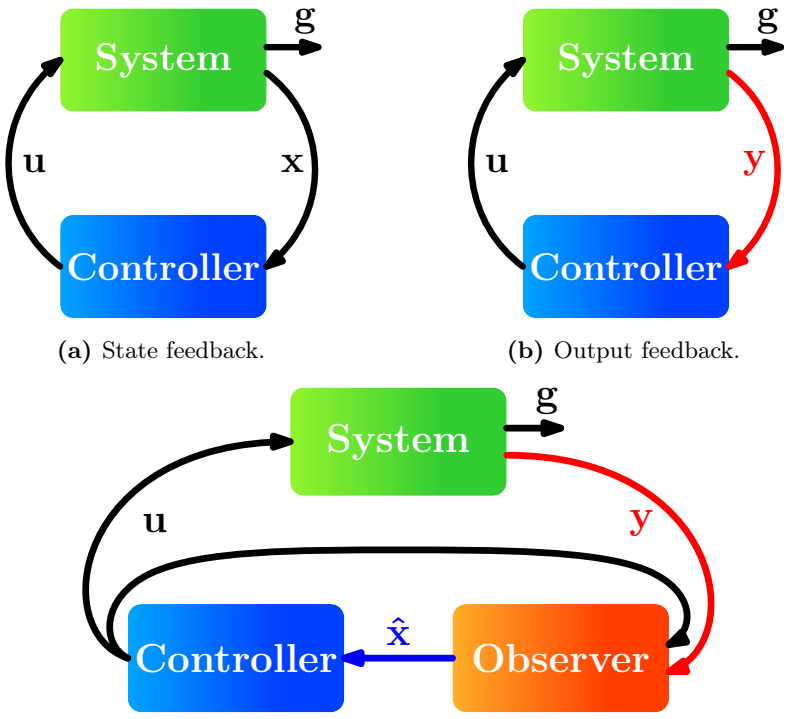
(d) AlphaZero [104].

Figure 1.1. Examples of control applications.

fusion research, where recent work has demonstrated the use of deep reinforcement learning to control a tokamak reactor. In this setting, the objective is to shape and maintain a high-temperature plasma within the vessel to generate energy, despite the highly nonlinear and unstable dynamics of the system [26]. Control policies have also been successfully applied to strategic decision-making problems such as board games like chess, Go, and shogi. In these cases, the objective is to maximize the probability of winning the game, subject to a large combinatorial state space and a complex set of rules. By combining reinforcement learning with Monte Carlo Tree Search, researchers have developed policies that outperform human experts in these domains [103, 105]. Despite the diversity of these applications, they can all

be formulated within common frameworks in which a control policy maps available information to decisions in order to optimize a specified performance criterion.

A schematic overview of three common architectures is depicted in Figure 1.2. In all three cases, a control policy generates an input (or decision) that is applied to the system, which evolves according to its internal dynamics and produces an output that is fed back to the controller, thereby forming a closed-loop interaction. In the first architecture, state feedback, the controller has direct access to the full system state without uncertainty and selects actions accordingly. In the second architecture, output feedback, the system provides only partial and/or noisy information about the underlying state, resulting in partial observability of the full system state. Control inputs and decisions are directly derived from measured outputs, such as sensor readings, or noisy versions of the full system state. A third intermediate architecture is observer-based state feedback, where



(a) State feedback.

(b) Output feedback.

(c) Observer-based state feedback.

Figure 1.2. Illustration of three common control architectures. The control input, system state, system output, and estimated state are denoted by u, x, y, and $\hat{x}$, respectively. g is a scalar that represents the performance.

an observer reconstructs an approximate state from past inputs and outputs, and the control policy operates on this estimate rather than on raw measurements. Across all three settings, the system dynamics may be subject to disturbances. Furthermore, each applied action incurs a scalar stage cost or reward that reflects immediate performance. These quantities are accumulated over time to define the performance criterion used to evaluate and compare policies. The fundamental difficulty in many applications lies in designing or learning policies that achieve strong performance despite limited state information, stochastic disturbances and measurement noise, and imperfect or no models.

While traditional control and decision-making approaches rely on models, a recent trend is to use reinforcement learning techniques. These techniques address control and decision-making problems without requiring a model, relying only on data from the system/process. In fact, reinforcement learning provides a general framework for designing control policies by optimizing cumulative performance criteria using models and/or data. Depending on the approach, this may involve constructing and exploiting an explicit model of the system or learning control policies directly from interaction data. From a systems and control perspective, reinforcement learning can be interpreted as a data-driven form of approximate dynamic programming for optimal control. However, properties fundamental in control theory, such as stability, robustness, and constraint satisfaction, are not inherently ensured by standard reinforcement learning algorithms. Viewing reinforcement learning through the lens of systems and control is therefore essential when deploying it in feedback control of dynamical systems.

From a theoretical standpoint, the foundations of reinforcement learning are rooted in dynamic programming and the framework of Markov decision processes. Comprehensive treatments of reinforcement learning algorithms and their properties can be found in standard textbooks, such as Sutton and Barto [115], which adopts a machine learning viewpoint, and Bertsekas [11], which approaches the subject from a systems and optimal control perspective. These works cover both model-free and model-based approaches, as well as approximate methods required for large-scale and continuous state and action spaces. Model-free methods gained popularity in recent years due to the increased availability of data and computational resources (see, e.g., [42, 77, 102]). Despite the strong theoretical foundation in literature, practical reinforcement learning in complex systems raises significant challenges related to data efficiency, partial observability, and uncertainty.

1.1 Policy iteration

A central class of algorithms within reinforcement learning and optimal control is that of policy iteration. These methods alternate between policy evaluation, in which the performance of a given policy is assessed, and policy improvement, in which the policy is updated based on this assessment, with the goal of converging to a globally or locally optimal policy. One of the earliest works on policy iteration was by Howard [49], which established convergence to an optimal policy for Markov decision processes under suitable conditions.

Figure 1.3 illustrates the general structure of the policy iteration framework, highlighting the interaction between policy evaluation and policy improvement. Starting from an initial policy, the policy evaluation step estimates the performance of the current policy, typically in terms of a value or action–value function that quantifies expected cumulative cost or reward. Based on this evaluation, the policy improvement step constructs a new policy that is greedy or approximately greedy with respect to the estimated performance measure. These two steps are repeated iteratively, forming a closed-loop procedure with the aim of progressively improving the performance of the policy, which is guaranteed in the exact case [10].

Depending on the setting, policy evaluation and improvement may be carried out exactly using a known system model, or approximately using data generated through interaction with the system. As a result, policy iteration encompasses a wide range of algorithms, including an alternative to model-based classical dynamic

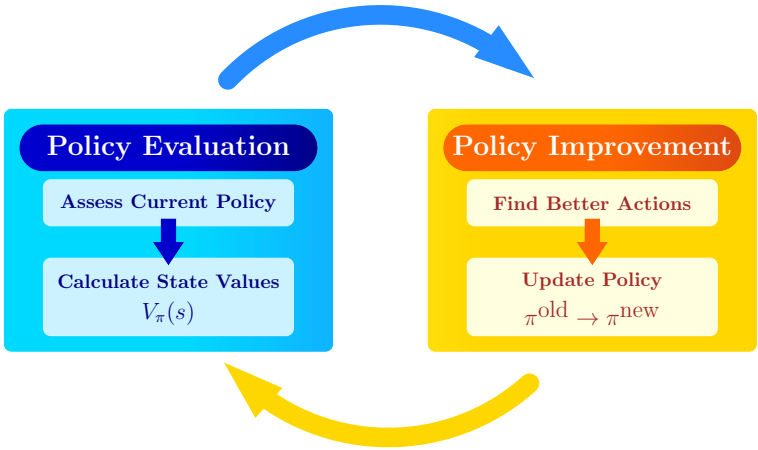


Figure 1.3. Schematic overview of the policy iteration framework, illustrating the alternating steps of policy evaluation and policy improvement that are repeated until convergence.

programming methods as well as approximate and reinforcement learning-based approaches for large-scale or partially known systems.

A class of algorithms that embodies the principles of policy iteration is the actor-critic framework. In these methods, the actor represents the policy, while the critic estimates the value function associated with the current policy. The critic evaluates the policy by approximating its expected performance, effectively performing the policy evaluation step of classical policy iteration. The actor then updates the policy parameters in a direction suggested by the critic, performing a policy improvement step. Actor-critic methods can therefore be interpreted in the context of Figure 1.3, where the critic corresponds to the policy evaluation step (left) and the actor to the policy improvement step (right). Whereas traditional policy iteration alternates between evaluation and improvement, actor-critic methods carry out both updates in parallel. Actor-critic methods form a central class of policy-gradient algorithms, combining an explicit policy representation with a learned value function to guide policy updates, and have been studied extensively in both classical and deep reinforcement learning settings [42, 57, 76, 116].

Despite its theoretical appeal, policy iteration has several practical limitations. First, it assumes full knowledge of the system model or, in the case of model-free variants, access to sufficient interaction data to accurately evaluate and improve the policy. In many real-world applications, collecting this data can be expensive, time-consuming, or even risky, limiting the feasibility of repeated policy evaluation and improvement steps. Second, policy iteration is inherently formulated for Markov decision processes, where the system state is fully observable and satisfies the Markov property. A process satisfies the Markov property if the current state contains all information from the past that is relevant for predicting future evolution [98, 115]. When the state is only partially observed, or the system dynamics are complex and non-Markovian (i.e. the Markov property does not hold), standard policy iteration may fail to converge or produce suboptimal policies. These limitations motivate the development of approximate, sample-efficient methods, and extensions for partially observable systems, which aim to retain the iterative improvement structure while addressing the practical constraints of real-world environments. One such perspective is provided by Bayesian statistics, which offer a unified way to reason about data efficiency and uncertainty. In the next section, we examine this framework in more detail.

1.2 Bayesian statistics as a unifying framework

Bayesian statistics provides a principled framework for learning from data and reasoning about uncertainty. At the core of Bayesian statistics lies the Bayes' rule, which describes how prior beliefs over unknown quantities are updated in light of observed data. The Bayes' rule is given as

$$\Pr(A|B) = \frac{\Pr(B|A)\Pr(A)}{\Pr(B)} \tag{1.1}$$

where $\Pr(A|B)$ is the conditional probability of event A given that B is true. $\Pr(B|A)$ is the likelihood of event B given that A is true, whereas $\Pr(A)$ and $\Pr(B)$ are the probabilities of observing events A and B, respectively.

In the context of reinforcement learning and control, this perspective makes it possible to include priors in a probabilistic formulation to enhance data efficiency for policy evaluation via maximum a posteriori estimation. Furthermore, Bayesian filtering provides a systematic mechanism for state estimation under partial observability. The probabilistic approaches of incorporating priors for policy evaluation and performing Bayesian filtering for state estimation are central to this thesis and form the foundation of the methods developed in the subsequent chapters.

1.2.1 Maximum a posteriori estimation

In maximum a posteriori estimation, the goal is to find an estimate for an unknown quantity that maximizes the posterior distribution of the unknown quantity given observed data. Let $\mathcal{D}$ denote an observed data set and θ an unknown quantity, such as model parameters or features of a value function, which one wants to estimate. The Bayes' rule yields the posterior distribution

$$\Pr(\theta|\mathcal{D}) = \frac{\Pr(\mathcal{D}|\theta)\Pr(\theta)}{\Pr(\mathcal{D})} \tag{1.2}$$

where $\Pr(\mathcal{D}|\theta)$ is the likelihood of the data under the model parametrised by θ, $\Pr(\theta)$ is a prior distribution capturing existing knowledge, and $\Pr(\mathcal{D})$ is the marginal likelihood (or evidence) that ensures the posterior is a valid probability distribution. The estimate that maximizes the posterior distribution $\Pr(\theta|\mathcal{D})$ is defined as

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} \Pr(\theta|\mathcal{D}) = \arg\max_{\theta} \Pr(\mathcal{D}|\theta)\Pr(\theta) \tag{1.3}$$

where the second argument follows from the fact that $\Pr(\mathcal{D})$ does not depend on θ. When $\Pr(\theta)$ represents a uniform distribution, the term can be disregarded from the optimization problem. In this case, the maximum a posteriori estimation boils down to maximum likelihood estimation.

This probabilistic framework also underpins active inference, a generalization of Bayesian estimation to sequential decision-making. In active inference, agents maintain a posterior over hidden states and select actions that minimize expected uncertainty and achieve desired outcomes, effectively combining maximum a posteriori-like state estimation with planning under uncertainty [85].

1.2.2 Bayes' filter

As already mentioned before, in many reinforcement learning and control problems, the system state cannot be observed directly. In these cases, one can infer the state from noisy sensor measurements. Bayesian filtering provides a principled framework for performing this inference sequentially over time by maintaining a probability distribution over the state conditioned on all data observed so far. Rather than estimating the state independently at each time step, the Bayes' filter propagates uncertainty forward through the system dynamics and refines it whenever new measurements become available, thereby supporting decision-making under partial observability.

The Bayes' filter consists of two steps that are applied recursively:

1. **Prediction:** In the prediction step, the current belief is propagated through the system dynamics to obtain a prior distribution for the next state. This step accounts for process disturbances and reflects how uncertainty in the current state evolves under the system dynamics and applied control.

2. **Correction:** When a new measurement y_k becomes available, the predicted distribution is refined through the correction step by applying Bayes' rule:

$$\Pr(x_k|y_{1:k}, u_{1:k-1}) = \frac{\Pr(y_k|x_k)\Pr(x_k|y_{1:k-1}, u_{1:k-1})}{\int \Pr(y_k|x_k)\Pr(x_k|y_{1:k-1}, u_{1:k-1})dx_k} \tag{1.4}$$

   Here $\Pr(y_k|x_k)$ is the measurement likelihood, and the denominator is a normalization constant ensuring that the updated belief integrates to one. This update is a direct application of Bayes' rule, with the predicted state distribution acting as the prior and the measurement model providing the likelihood.

Together, the prediction and correction steps constitute the recursive Bayes' filter, which forms the conceptual foundation for a wide range of state estimation methods, including the Kalman filter [56] and its nonlinear extensions, as well as the Fourier-based Bayesian estimation techniques developed later in this thesis.

1.3 Objectives and contributions

Policy iteration provides a powerful framework to find optimal policies. As stated already above, its practical applicability is often restricted by limited data availability and partial information about the system state. Indeed, in many real-world control problems, only indirect or noisy observations are available, system models are imperfect or unknown, and data collection is costly or risky. These challenges affect both policy evaluation and policy improvement. As a result, there is a need for policy iteration methods that remain reliable and data-efficient when applied to realistic control systems operating under uncertainty and incomplete information, such as partially observable Markov decision processes. Therefore, the main objective of this thesis is:

Provide new policy iteration methods for settings with limited data and partial information, while providing convergence guarantees and achieving data efficiency.

The following three sections outline the research contributions that support this main objective.

1.3.1 Contributions to policy iteration for output feedback systems

When control decisions are based on outputs, the resulting decision process is generally non-Markovian, which fundamentally challenges the assumptions underlying classical policy iteration and complicates the design of monotonic policy improvement schemes. This section introduces the contributions of the thesis that extend policy iteration to these output feedback settings by explicitly accounting for partial observability and non-Markovian dynamics. The proposed methods cover both episodic and infinite-horizon problems, and address systems with discrete as well as continuous state and action spaces, thereby broadening the applicability of policy iteration to a wide range of output-based control problems.

Policy iteration for episodic POMDPs

Partially observable Markov decision processes (POMDPs) provide a general framework for decision-making under partial information. Classical solution approaches rely on belief-state representations, which transform the POMDP into a fully observable Markov decision process at the cost of a high-dimensional and often continuous state space. As an alternative, memoryless and finite-memory policies

operate directly on the available outputs, offering a more scalable and practical solution for many episodic tasks. However, when policies are defined on outputs rather than states or beliefs, the resulting output process is generally non-Markovian. This non-Markovian structure breaks a key assumption underlying classical policy iteration, causing policy-improvement steps at different stages of an episode to become interdependent and invalidating standard monotonicity arguments. The result is no guaranteed convergence to a (local) optimal policy.

To address these challenges, new policy iteration methods are required that can operate on output-based policies while preserving theoretical performance guarantees and computational efficiency. The first contribution, covered in Chapter 2, introduces a class of policy iteration algorithms tailored to episodic POMDPs, which explicitly account for the non-Markovian nature of the output process and enable structured, monotonic policy improvement.

Contribution I. *In the thesis a family of monotonically improving policy iteration algorithms for episodic POMDPs is developed that operate directly on output-based policies.*

The proposed monotonically improving policy iteration algorithms alternate between single-stage output-based policy improvements and policy evaluations according to a prescribed periodic pattern. We show that this family admits optimal patterns that maximize a natural computational-efficiency index, and we identify the simplest pattern with minimal period. Building on this structure, we further develop a model-free variant that estimates values from data and learns memoryless policies directly.

Policy iteration for static output feedback

Many control applications, e.g., linear dynamical systems with quadratic cost criteria, involve continuous state and action spaces. In this context, static output feedback policies are of particular practical interest, as they avoid explicit state estimation and reduce computational complexity. However, extending policy iteration to static output feedback remains challenging, since the output process is again non-Markovian and the policy optimization problem is inherently non-convex, even in linear-quadratic systems.

Building on the policy iteration framework developed for episodic POMDPs, the second contribution considers extensions to continuous state and action spaces and application to linear–quadratic systems. By exploiting the structure of linear dynamics and quadratic costs, the resulting methods enable output-based policy updates that preserve monotonic performance improvement. In addition, the frame-

work is extended beyond finite-horizon formulations to infinite-horizon settings, and the scalar case is analysed to provide insight into stability and convergence. This contribution will be covered in Chapter 3.

Contribution II. *In the thesis the policy iteration method for episodic POMDPs is extended to continuous POMDPs and applied to linear-quadratic systems and infinite-horizon.*

For continuous finite-horizon POMDP problems, we prove that the method converges to locally optimal output feedback policies. When applied to linear-quadratic systems, the resulting updates reduce to Riccati-like recursions. The framework is further extended to infinite-horizon linear-quadratic systems, where convergence and average-cost improvement are established for scalar systems.

1.3.2 Contributions to data-efficient policy evaluation

Policy evaluation is a central component of policy iteration, as it provides the performance estimates that guide the policy improvement step. In settings with limited data, accurate policy evaluation becomes particularly challenging, since methods usually rely on large amounts of interaction data or exact knowledge of the system dynamics. Approximate and data-driven policy evaluation methods, such as temporal difference learning, offer practical alternatives, but their performance and convergence properties depend critically on the amount of available data and how efficiently this data is used. Least-squares temporal difference learning is a class of policy evaluation methods that estimate value functions by solving a least-squares problem based on observed transitions, leading to data-efficient estimates. The contributions of this thesis related to data-efficient policy evaluation will be introduced in the remainder of this section. For the sake of brevity, we focus here on the crucial step of policy evaluation. The ideas can be extended to policy iteration by performing the usual cycle of policy evaluation and policy improvement.

Including prior knowledge

In many control applications, some form of prior knowledge about the system is available before data-driven learning begins. This knowledge may originate from first-principles modelling, previous experiments, or engineering intuition, and is often sufficient to describe approximate system dynamics or structural properties of the value function. Classical data-driven policy evaluation methods, however, typically do not treat such information and ignore it altogether, relying solely on

interaction data to estimate value functions. As a result, valuable information is discarded, leading to inefficient use of data and slow convergence when data is scarce.

A Bayesian perspective provides a structured way to incorporate prior knowledge into policy evaluation by encoding it as a prior distribution and combining it with observed data through maximum a posteriori estimation. In the context of least-squares temporal difference learning, this approach enables prior model information to directly influence value function estimates, improving robustness and data efficiency while retaining the computational structure of classical methods. The third contribution addresses this topic, and will be covered in Chapter 4.

Contribution III. *In the thesis prior model knowledge is encapsulated in a suitable prior to perform maximum a posteriori least-squares temporal difference learning.*

A Bayesian extension of least-squares temporal difference learning that incorporates prior knowledge of the system dynamics without explicitly identifying a parametric model is introduced. By representing state-transition probabilities with Dirichlet distributions, prior information can be systematically fused with data through maximum a posteriori estimation, as introduced in Section 1.2.1. The resulting algorithm preserves the recursive structure of classical LSTD while significantly improving convergence speed and estimation accuracy when data is scarce.

Expected eligibility traces

Eligibility traces provide a mechanism to propagate information over multiple time steps in temporal difference learning, thereby interpolating between one-step methods and Monte Carlo evaluation. In TD(λ) and LSTD(λ), these traces are constructed from realized state or feature trajectories and therefore reflect only the histories observed in the data. While this approach is effective in many settings, it inherently depends on sampled trajectories and may fail to exploit the full structure of the underlying Markov decision process.

Recently, expected eligibility traces have been proposed as an alternative that accounts not only for realized trajectories, but also for all potential feature histories that could have occurred under the system dynamics and policy. By replacing sample-based traces with their expected counterparts, these methods aim to reduce variance and better capture long-term dependencies in value function estimation. The fourth contribution, covered in Chapter 5, investigates the consequences of extending least-squares temporal difference learning with expected and mixed

eligibility traces, with particular attention to their equivalence properties and implications for data-efficient policy evaluation.

Contribution IV. *In the thesis least-squares temporal difference learning is extended with expected eligibility traces.*

We show that, unlike in direct TD methods, augmenting LSTD with theoretical expected eligibility traces does not change the resulting value-function estimate and is in fact equivalent to standard LSTD(0). More generally, LSTD with mixed traces is proven to be equivalent to LSTD with a suitably modified trace-decay parameter, thereby unifying several algorithms. The chapter further studies empirical variants based on sample-average traces and demonstrates that their estimates converge to those of the corresponding LSTD formulations.

1.3.3 Contributions to state estimation

While output feedback policy iteration provides a powerful alternative to full state-based control, there are many applications in which output feedback control alone is insufficient, and reliable state estimation becomes essential. In such settings, control policies must be supported by state estimation mechanisms that infer the state of the system from measurements. Consequently, the estimated state can be used in combination with state feedback (as in Figure 1.2c). This approach is referred to as certainty equivalence control [11]. State estimation therefore plays a complementary and, in some cases, indispensable role in enabling effective control and learning under partial observability.

The contributions of this thesis to Bayesian state estimation will be introduced in this section. The contributions are related to both practical estimation problems arising in real-world sensing applications, as well as fundamental limitations related to the computational complexity of Bayesian filtering.

Pose estimation

Accurate pose estimation from visual data is a fundamental problem in many industrial and robotic applications, where reliable state information is required to enable automated decision-making and control. In vision-based systems, the object pose is not observed directly but must be inferred from camera images, which are high-dimensional, noisy, and often subject to varying conditions. This makes pose estimation a challenging state estimation problem under severe partial observability.

Bayesian filtering provides a principled framework to address these challenges by combining prior knowledge, system dynamics, and image-based measurements

into a coherent probabilistic estimate of the object pose. However, constructing informative likelihood functions from raw image data remains a key difficulty, particularly in industrial settings where robustness and computational efficiency are critical. The fifth contribution, covered in Chapter 6, investigates vision-based pose estimation methods that integrate Bayesian filtering with data-driven techniques to obtain reliable pose estimates from images in both simulated and real-world environments.

Contribution V. *In the thesis a framework is developed to estimate the pose of an object using camera images by applying Bayesian filtering techniques.*

We present two complementary vision-based pose-estimation algorithms for industrial automation, both embedded within a Bayesian filtering framework. The first combines Gaussian process regression with template matching and Bayesian optimization to construct informative image-based likelihood functions, while the second employs deep neural networks to map camera images directly to parametrised likelihood models. The methods are validated in both simulated and industrial environments, where they provide reliable and consistent pose estimates for automated in-feed to coordinate measuring machines.

Bayesian estimation using Fourier basis functions

Bayesian filtering provides an exact and principled framework for state estimation under uncertainty, but its applicability is often limited by the curse of dimensionality. In general, the Bayes' filter operates on probability density functions defined over continuous state spaces, and repeated prediction and update steps can lead to increasingly complex distributions that are difficult to represent and compute. This challenge is particularly pronounced in systems with non-linear measurement models and non-Gaussian disturbances, where classical parametric filters are no longer applicable.

The sixth contribution, covered in Chapter 7, explores an alternative representation of Bayesian estimation based on a frequency-domain interpretation of the Bayes' filter. By expressing probability densities in terms of Fourier series coefficients, the filtering operations can be reformulated as algebraic manipulations in a countable coefficient space. Under mild assumptions, this representation enables exact Bayesian estimation within a structured and tractable function space, and

naturally leads to approximate methods with controlled computational complexity.

Contribution VI. *In the thesis the Bayes' filter is reformulated in the frequency domain, enabling exact and approximate Bayesian filtering in a countable space of Fourier series coefficients.*

We develop a frequency-domain formulation of Bayesian filtering for linear systems with bounded inputs, non-Gaussian disturbances, and non-linear measurement models, showing that exact filtering can be carried out in a countable space of Fourier coefficients under mild assumptions. Truncating the resulting Fourier expansions yields a principled approximate filter whose complexity remains constant during prediction steps and grows only linearly during measurement updates.

1.4 Outline of the thesis

The contributions of this thesis are organized in three parts: Part II (Chapters 2 and 3) covering policy iteration for output feedback systems, Part III (Chapters 4 and 5) covering data-efficient policy evaluation methods, and Part IV (Chapters 6 and 7) covering state estimation for output feedback systems. In Chapter 2, a new family of monotonically improving policy-iteration algorithms is proposed that alternate between single-stage output-based policy improvements and policy evaluations for systems with finite state and action spaces. These ideas are extended to continuous state and action spaces in Chapter 3. In Chapter 4, a data-based method is proposed to incorporate prior model knowledge in policy evaluation as an extension to least-squares temporal difference learning. Next, in Chapter 5, the possibilities and consequences of extending least-squares temporal difference with expected eligibility traces are investigated. In Chapter 6, a method is proposed to estimate the pose of an object using an image via Bayesian filtering. Furthermore, in Chapter 7, a frequency-domain interpretation of the operations of the Bayes' filter is given. We show that, under mild assumptions, exact Bayesian estimation can be pursued in a countable space of Fourier series coefficients, rather than in the usual functional space of probability densities. Conclusions and recommendations for future work are given in Part V (Chapter 8).

A note for the reader. Chapters 2-7 are all based on submitted/published articles and consequently are self-contained and can be read independently. A reference to the corresponding research paper is included at the beginning of each chapter.

II

Policy Iteration for Output-Feedback Systems

Chapter 2

Memoryless Policy Iteration

Abstract - Memoryless and finite-memory policies offer a practical alternative for solving partially observable Markov decision processes (POMDPs), as they operate directly in the output space rather than in the high-dimensional belief space. However, extending classical methods such as policy iteration to this setting remains difficult; the output process is non-Markovian, making policy-improvement steps interdependent across stages. We introduce a new family of monotonically improving policy-iteration algorithms that alternate between single-stage output-based policy improvements and policy evaluations according to a prescribed periodic pattern. We show that this family admits optimal patterns that maximize a natural computational-efficiency index, and we identify the simplest pattern with minimal period. Building on this structure, we further develop a model-free variant that estimates values from data and learns memoryless policies directly. Across several POMDP examples, our method achieves significant computational speedups over policy-gradient baselines and recent specialized algorithms in both model-based and model-free settings.

This chapter is based on: R.A.C. van Zuijlen and D.J. Antunes, "Memoryless Policy Iteration for Episodic POMDPs", *(submitted)*.

2.1 Introduction

Finding an optimal policy for a partially observable Markov decision problem (POMDP) is PSPACE-complete [83]. In the episodic case, the optimal value function over the belief simplex is piecewise linear but its complexity grows exponentially with the horizon [109]. As a result, a large body of work has focused on suboptimal but practical solution methods; see the survey papers [111], [63]. A prominent class of such methods restricts the policy search to memoryless or finite-memory policies that depend on a short window of past output. However, once cast in the output space of observations, this system becomes non-Markovian, and computing an optimal memoryless policy is itself an NP-hard problem [68], [122].

Approximate model-based and model-free techniques have therefore been explored to compute memoryless policies, long regarded as an open problem [6]. Since finite-memory policies can be represented as memoryless policies through state augmentation [67] most approaches below, and the methods proposed in this chapter, apply to both settings.

In the context of model-based approaches, [20] propose a rolling-horizon method for computing memoryless policies using mixed-integer linear programming (MILP). Across a large benchmark of POMDPs, their MILP-based policy is within 20% of the optimal POMDP value in 60% of the problems. [79] prove that the state-action frequencies and the expected cumulative rewards are rational functions of the policy and use linear optimization subject to polynomial constraints to find a memoryless policy. [112] show that finite-state controllers, which include memoryless policies, can be expressed with options via the so called option-observation initiation sets method, which simplifies the search for memoryless policies. More recently, [107] study problems modelled with the so-called agent state (a generally non-Markovian construct that subsumes finite output windows) and show that periodic time-varying policies can outperform stationary ones.

In the context of model-free methods, [7] propose a reinforcement learning (RL) method that relies on estimating the parameters of a POMDP using spectral methods, an oracle to compute an optimal memoryless policy, and a smart exploration strategy. Related to this approach, [54] propose a sample-efficient RL method for episodic undercomplete POMDPs, where the number of observations exceeds the number of states; data is used to estimate a model and efficiently guide exploration. [113] present policy gradient based online RL algorithms for POMDPs which learn an agent information state representation using multi-scale stochastic gradient descent. [108] discuss how ideas from approximate information state for computing agent-based (and memoryless) policies can be used to improve Q-Learning and actor-critic algorithms for POMDPs. [87] proposes a Q-learning type algorithm relying on stochastic optimization. Systems with non-Markovian rewards are considered in [19], in which a reward shaping approach is proposed

to handle temporal logic specifications. Policy gradient methods, which rely on parametrizations of stochastic policies and optimize its parameters via gradient estimates, have also been proposed in the context of memoryless policies [52], [9]. Several recent papers propose deep RL approaches for POMDP [18], [51], [4], [99].

A central challenge in all these approaches is the lack of the Markov property in the output space, which prevents direct application of standard MDP planning and RL techniques such as, most notably, policy iteration. Only a few papers attempt to extend policy iteration to the memoryless setting. [66] propose a policy-iteration scheme for the average-reward case, but each iteration requires an exhaustive search over the policy space, which is computationally prohibitive except for very small problems or heavily restricted policy classes. Memoryless policies are special cases of finite-state controllers, for which policy iteration has been proposed [43, 44]. However, the algorithms in [43, 44] work explicitly in the belief space and they do not scale up well with respect to the size of the problem [75]. In the related setting of general non-Markovian systems, [52] propose an average-cost algorithm reminiscent of policy iteration, but their policy improvement step performs a gradient update over the probabilities of a randomized policy, based on cost estimates from policy evaluation. In contrast, we focus on deterministic memoryless policies and depart from gradient-based approaches, which are fundamentally different from policy iteration and, as we shall see, substantially less computationally efficient for the problems we target.

Stochastic memoryless policies have been reported to outperform deterministic ones in several non-Markovian contexts [8], [106], [70], [124], [58]. However, the empirical results in [20] question the generality of this claim, and [129] shows that deterministic finite-history policies can approximate the optimal deterministic POMDP policy arbitrarily well. In our view, the structure of the output space plays a crucial role in comparing deterministic and stochastic policies. Deterministic policies, on which we focus in this chapter, are particularly appealing in applications where predictability and safety are essential.

In this chapter, we introduce a new family of policy-iteration algorithms for computing deterministic memoryless policies in episodic POMDPs. Each algorithm in this family alternates between single-stage, output-based policy improvements and policy-evaluation steps following a prescribed periodic pattern that determines which stages are improved and when. The non-Markovian nature of the output process introduces two key departures from classical MDP policy iteration. First, policy evaluation must compute not only the state–action Q-values but also the state-distribution trajectory, since both quantities influence output-based decisions. Second, policy improvements cannot be applied simultaneously across all stages: improvements at one stage affect the value of subsequent stages, making stagewise improvements inherently interdependent.

We analyse how the choice of periodic pattern influences a natural index of

computational efficiency in the time-invariant setting, and identify the patterns that maximize this index. Among them, we advocate the simplest optimal pattern, namely the one with minimal period. Under mild conditions, we show that every algorithm in this family converges monotonically to a locally optimal memoryless policy. Building on this structure, we further introduce a model-free variant that learns memoryless policies directly from data. Experiments demonstrate substantial computational gains over policy-gradient baselines and recent specialized methods, in both model-based and model-free settings.

The remainder of the chapter is organized as follows. Section 2.2 formulates the problem. Section 2.3 proposes the general class of algorithms and presents the main results. Section 2.4 tackles model-free methods. Section 2.5 provides numerical examples. Section 2.6 discusses future work and provides final remarks.

Notation: The number of elements in set $\mathcal{A}$ is denoted by $|\mathcal{A}|$. $\|\cdot\|_\Lambda^2$ denotes the weighted squared norm: $\|x\|_\Lambda^2 = x^\top \Lambda x$. The indicator function $\mathbb{1}\{E\}$ equals 1 if the statement E is true and 0 otherwise.

2.2 Problem formulation

Consider a standard Partially Observable Markov Decision Process. The states and actions at time $t \in \mathcal{T} := \{0, 1, ..., T\}$, $T \in \mathbb{N}$, are denoted by $S_t \in \mathcal{S}$ and $A_t \in \mathcal{A}$, where $\mathcal{S}$ and $\mathcal{A}$ are finite sets. The dynamics are characterized by

$$p_t(s'|s, a) = \Pr\{S_{t+1} = s' \mid S_t = s, A_t = a\}. \tag{2.1}$$

The initial state is assumed to be distributed according to a known probability distribution $\mu_0(s) = \Pr\{S_0 = s\}$. Observations at time t are denoted by $O_t \in \mathcal{O}$, where $\mathcal{O}$ is a finite set. The observation model at time t is given by

$$q_t(o|s) = \Pr\{O_t = o \mid S_t = s\}. \tag{2.2}$$

A finite time horizon (episodic task) is considered. The reward at time t is denoted by R_t and the (stage-dependent) expected reward is a function of the current state and action,

$$r_t(s, a) = \mathbb{E}\left[R_t | S_t = s, A_t = a\right]. \tag{2.3}$$

For POMDPs the optimal policy is in general a function of the complete history of observations and actions $H_t := \{A_0, O_0, ..., A_{t-1}, O_t\}$, but it can also be written as a function of the belief state $b_t(s) = \Pr\{S_t = s|H_t\}$ [109]. History dependent policies scale poorly with time and belief state dependent policies scale poorly with the state dimension n, making both approaches computationally impractical.

Motivated by this, in this chapter, we focus instead on finding (suboptimal) time-varying policies $\{\pi_t : \mathcal{O} \to \mathcal{A}\}$ that only depend on the current observation

$$\pi_t(a|o) = \Pr\{A_t = a | O_t = o\}. \tag{2.4}$$

With some abuse of notation, we use $\pi_t : \mathcal{O} \to \mathcal{A}$, where $\pi_t(o)$ is an action, to denote deterministic observation-based policies and probability $\pi_t(a|o) \in [0, 1]$ for their equivalent degenerate probabilistic representation

$$\pi_t(a|o) := \mathbb{1}\{a = \pi_t(o)\}. \tag{2.5}$$

More specifically, we are interested in *deterministic* policies: $\{\pi_t : \mathcal{O} \to \mathcal{A}\}_{t=0}^{T-1}$ that maximize the expected episodic return

$$L^\pi = \mathbb{E}\left[\sum_{t=0}^{T-1} R_t + V_T(s_T)\right] \tag{2.6}$$

where $V_T(s_T)$ is the terminal cost. These policies scale well, are simple to implement, and give predictable behaviour. In the next section, we propose a model-based policy iteration scheme, that monotonically improves an initial deterministic observation-based policy to a (local) optimum.

2.3 Model-based policy iteration for memoryless policies

A general class of memoryless policy iteration algorithms for episodic POMDPs is presented in Section 2.3.1 where it is shown that they converge monotonically. A proposed computationally efficient instance of this class is presented in Section 2.3.2.

2.3.1 Memoryless policy iteration with periodic stage updates

Consider a memoryless policy $\pi_t(a|o)$, let $Q_T^\pi(s, a) = V_T(s)$ and let the state-action value at time t be

$$\begin{aligned} Q_t^\pi(s, a) &= r_t(s, a) \\ &\quad + \sum_{s'} p_t(s' \mid s, a)\Big(\sum_{o'} q_{t+1}(o' \mid s')\Big(\sum_{a'} \pi_{t+1}(a' \mid o') Q_{t+1}^\pi(s', a')\Big)\Big) \end{aligned} \tag{2.7}$$

where the inner summation boils down to $Q_{t+1}^\pi(s', \pi_{t+1}(o))$ for deterministic policies (2.5). These state-action values can be computed backward in time from $t = T - 1$ until $t = 0$. Moreover, let the observation-action value be

$$\bar{Q}_t^\pi(o, a) = \sum_s Q_t^\pi(s, a)\alpha_t(s \mid o) \tag{2.8}$$

where $\alpha_t(s|o) = \Pr\{S_t = s|O_t = o\}$ is the posterior distribution of the state conditioned on observation o. Using Bayes' rule:

$$\alpha_t(s|o) = \frac{\Pr\{O_t = o|S_t = s\}\mu_t(s)}{\gamma_t(o)} \tag{2.9}$$

with $\gamma_t(o) = \Pr\{O_t = o\} = \sum_s \Pr\{O_t = o|S_t = s\}\mu_t(s)$, and prior $\mu_t(s) = \Pr\{S_t = s\}$. The state distribution evolves under π as

$$\mu_{t+1}(s) = \sum_{s'} \Big(\sum_{o'} \Big(\sum_{a'} p_t(s|s', a')\pi_t(a'|o') \Big) q_t(o'|s') \Big) \mu_t(s') \tag{2.10}$$

given the initial state distribution $\mu_0(s)$, where the inner summation boils down to $p_t(s|s', \pi_t(o'))$ for deterministic policies (2.5).

The proposed class of policy iteration algorithm is parametrized by an arbitrary periodic sequence $\tau_\ell \in \mathcal{T}$, $\ell \in \{0, 1, \dots, \}$ with arbitrary period M.

Given a deterministic initial policy π^0 and an M-periodic schedule τ_ℓ, set $\ell = 0$.

1. **Policy evaluation:** Compute $\bar{Q}_t^{\pi^\ell}(o, a)$ for $t = \tau_\ell$.
2. **Policy improvement:** Compute

$$\pi_t^{\ell+1}(o) = \arg\max_a \bar{Q}_t^{\pi^\ell}(o, a) \text{ for } t = \tau_\ell \tag{2.11}$$

and set $\pi_t^{\ell+1}(o) = \pi_t^\ell(o)$ for $t \neq \tau_\ell$. Set $\ell \to \ell + 1$ and repeat 1 and 2 until the policy does not change over M consecutive policy improvement steps.

We call this algorithm for a fixed sequence $\{\tau_\ell\}$ memoryless policy iteration. Note that to compute $\bar{Q}_t^{\pi^\ell}(o, a)$ in step 1 for a new policy obtained in step 2 one needs to compute the state-action value and the state distribution at time t. This can be done with a forward pass of (2.10) from $t = 0$ until $t = \tau_\ell$ and a backward pass of (2.7) from $t = T - 1$ until $t = \tau_\ell$.

In fully observable MDPs $\big(\mathcal{O} = \mathcal{S},\ q_t(o|s) = \mathbb{1}\{o = s\}\big)$, one has

$$\alpha_t(s|o) = \mathbb{1}\{s = o\},$$
$$\bar{Q}_t^\pi(o, a) = Q_t^\pi(o, a),$$

and μ_t plays no role in policy improvement. Therefore, standard policy iteration updates all stages simultaneously. In contrast, under partial observability, a change in policy at stage $t = \tau_\ell$ propagates through μ_t, altering $\bar{Q}_{t'}^\pi(o, a)$ at other stages. Thus, the observation-action values of interest (namely that for stage $\tau_{\ell+1}$) need to be updated immediately after a stage policy improvement is carried out, otherwise the policy improvement step at stage $\tau_{\ell+1}$ does not necessarily lead to a cost improvement.

We assume that the periodic sequence τ_ℓ is onto, ensuring that every stage is eventually updated. Moreover, we assume that $\tau_{\ell+1} \neq \tau_\ell$ as two consecutive policy improvements at the same stage would be redundant. Under these conditions, analogously to classical policy iteration, we can guarantee monotonic improvement of the expected episodic return and convergence to a fixed policy. Unlike the fully observable MDP setting, however, the limiting policy need not be globally optimal due to the intrinsic non-Markovian structure of the output process. We say that a memoryless policy is locally optimal if no single-stage, observation-based deterministic deviation can further improve the expected initial-stage value. The formal statement is provided next and the proof is provided in Section 2.7.1.

Theorem 2.1. *Let π^ℓ be the policy produced by memoryless policy iteration under an arbitrary periodic onto sequence τ_ℓ satisfying $\tau_{\ell+1} \neq \tau_\ell$, and denote its expected episodic return by L^{π^ℓ} as in* (2.6)*. Then*

$$L^{\pi^{\ell+1}} \geq L^{\pi^\ell} \text{ for all } \ell \in \{0, 1, 2, \dots\}$$

and the algorithm terminates at a locally optimal memoryless policy. ▲

There are many possibilities to choose the periodic sequence τ_ℓ, such as forward sweeps:

$$\tau = \{0, 1, ..., T-1, 0, 1, ..., T-1, ...\}$$

or, more in style of Dynamic Programming, backward sweeps:

$$\tau = \{T-1, T-2, ..., 1, 0, T-1, T-2, ..., 1, 0, ...\}.$$

Each sequence leads, in general, to a distinct *local* optimum. The backward sweep leads to a policy-iteration algorithm similar to that for MDPs. In fact, as one sweeps backwards only the state-action value needs to be updated and the state distribution update can be omitted as it is trivially obtained for MDPs. However, in POMDPs this is not necessarily the most efficient ordering. Optimized patterns are discussed next.

2.3.2 Efficient memoryless policy iteration

A key observation for reducing the computational effort of memoryless policy iteration under a given sequence τ_ℓ is the following. Suppose a policy improvement is performed at stage τ_0, and the next update occurs at $\tau_1 > \tau_0$. Then only the state distributions $\mu_{\tau_0+1}, \dots, \mu_{\tau_1}$ must be recomputed, since the change at τ_0 only affects the forward propagation of μ_t. In contrast, the state–action values $Q_t^{\pi^1}(s, a)$ for $t \geq \tau_0$ need not be recalculated until a later improvement is executed, because they can be obtained by backward recursion from (2.7). Consequently, computing

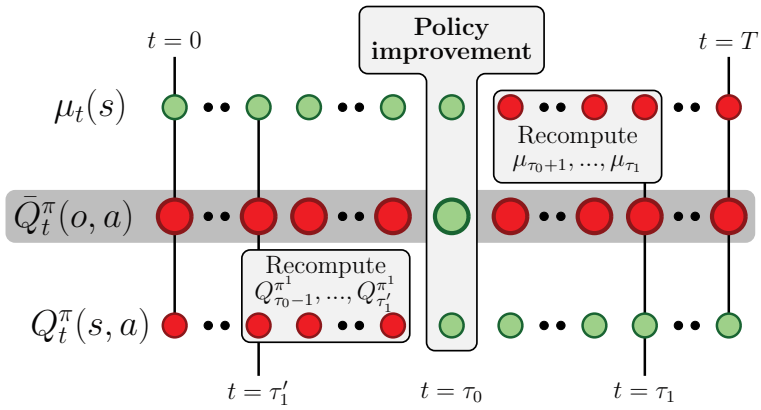


Figure 2.1. If the policy is improved at $t = \tau_0$, $\bar{Q}_t^{\pi^1}(o, a)$ is outdated for $t < \tau_0$ since $Q_t^{\pi^1}(s, a)$ changes, and for $t > \tau_0$ since $\mu_t(s)$ changes. If the next improvement occurs at τ_1, where $\tau_1 > \tau_0$, only $\mu_{\tau_0+1}, ..., \mu_{\tau_1}$ have to be evaluated. Moreover, if the next improvement occurs at τ_1', where $\tau_1' < \tau_0$, only $Q^{\pi^1}_{\tau_0-1}, ..., Q^{\pi^1}_{\tau_1'}$ have to be evaluated. Changes in quantities are denoted by ●, whereas unchanged quantities are denoted by ●.

$\bar{Q}^{\pi^1}_{\tau_1}(o, a)$ and performing the improvement at stage τ_1 only requires updating $\mu_{\tau_0+1}, \ldots, \mu_{\tau_1}$.

A symmetric argument holds when the next improvement occurs at $\tau_1' < \tau_0$. In that case, only the values $Q_t^{\pi^1}$ for $t = \tau_0 - 1, \ldots, \tau_1'$ must be recomputed backward, while the distributions μ_t for $t \leq \tau_0$ remain unaffected, as they depend solely on forward propagation from the initial state. These findings are summarized and illustrated in Figure 2.1.

Building on this insight, we define an index that quantifies the number of policy evaluation operations per period required by a sequence $\{\tau_\ell\}$ in the time-invariant case (p_t, q_t, r_t do not depend on t)

$$C = \frac{1}{M} \sum_{\ell=0}^{M-1} w_\mu \max(\tau_{\ell+1} - \tau_\ell, 0) + w_Q \max(0, \tau_\ell - \tau_{\ell+1}) \tag{2.12}$$

where

- w_μ - computational cost of one stage state distribution update (2.10),
- w_Q - computational cost of one stage state-action update (2.7).

Although w_μ and w_Q are often comparable (both involve summation over states and observations), distinguishing them enables a more general efficiency analysis.

The index C represents the total number of update operations required to perform M policy improvements, hence smaller C leads to faster learning, as improvements increase performance while evaluations are auxiliary. Note that this performance index assumes that the computational cost of each policy-evaluation step, as well as the expected value improvement of each policy-improvement step, is identical across all stages. The conclusions that follow therefore apply only to problems that (approximately) satisfy these assumptions. They do not extend to settings in which certain stages yield systematically larger value gains, or where policy-evaluation costs vary significantly across stages (e.g., time-varying problems).

The next result identifies the most computationally efficient periodic sequences.

Theorem 2.2. *Among all periodic onto update sequences satisfying $\tau_{\ell+1} \neq \tau_\ell$, those that minimize the computational cost index* (2.12) *are exactly the sequences for which $|\tau_{\ell+1} - \tau_\ell| = 1, \quad$ for every $\ell \in \mathbb{N}$, for any choice of weights w_μ and w_Q with $w_\mu + w_Q > 0$. Within this set of minimizing sequences, the following sequence is the unique one (up to cyclic shifts) with minimal period:*

$$\tau = (0, 1, \dots, T-2, T-1, T-2, \dots, 1, 0, 1, \dots).$$

In other words, the optimal periodic schedule with minimal period consists of a forward sweep $0, 1, \dots, T-1$ followed by a backward sweep $T-1, T-2, \dots, 0$, repeated indefinitely. ▲

Proof: Consider one period of the periodic sequence as $\tau_0, ..., \tau_{M-1}, \tau_M = \tau_0$, with the cost defined as in (2.12). Let $S_+ := \sum_{\ell=0}^{M-1} \max(\tau_{\ell+1} - \tau_\ell, 0)$ and $S_- := \sum_{\ell=0}^{M-1} \max(0, \tau_\ell - \tau_{\ell+1})$. By periodicity $\sum_{\ell=0}^{M-1} (\tau_{\ell+1} - \tau_\ell) = 0$, hence $S_+ = S_-$. Moreover, let $V := \sum_{\ell=0}^{M-1} |\tau_{\ell+1} - \tau_\ell| = S_+ + S_- = 2S_+$, then

$$C = \frac{w_\mu S_+ + w_Q S_-}{M} = \frac{w_\mu + w_Q}{2} \cdot \frac{V}{M}. \tag{2.13}$$

Thus, since $w_\mu + w_Q$, minimizing C is equivalent to minimizing the average variation V/M. Sequences that minimize the average variation are characterized by $|\tau_{\ell+1} - \tau_\ell| = 1$, leading to $V/M = 1$. In fact, if there exists an ℓ such that $|\tau_{\ell+1} - \tau_\ell| > 1$, then $V/M > 1$. Since the periodic sequence must be onto, the unique minimizer with minimal period M, is the monotone forward and backward sweep (and its cyclic shifts).

■

The specified optimal sequence with minimal period alternates between forward and backward sweeps. After each improvement step, only a single instance of either $\mu_t(s)$ or $Q_t^\pi(s, a)$ has to be recomputed. This sequence leads to two policy improvements per stage over a period. An optimal sequence with a larger period

would have a different number of policy improvements for different stages, and thus favour some stages. We therefore pick the one with the smallest period. This most efficient version of the general class of memoryless policy improvement algorithms provided in the previous section is summarized in Algorithm 1 and illustrated in Figure 2.2.

Consider again the forward and backward sweep at the end of Section 2.3.1. The cost C for those two sequences with period $M = T$ is

$$C = \frac{w_\mu + w_Q}{2} \cdot \frac{2(T-1)}{T} = \frac{(T-1)(w_\mu + w_Q)}{T} \geq C^* \tag{2.14}$$

for $T \geq 2$. Any sequence that has jumps, such that $|\tau_{\ell+1} - \tau_\ell| \geq 1$, has a higher average variation than 1, and thus a larger cost.

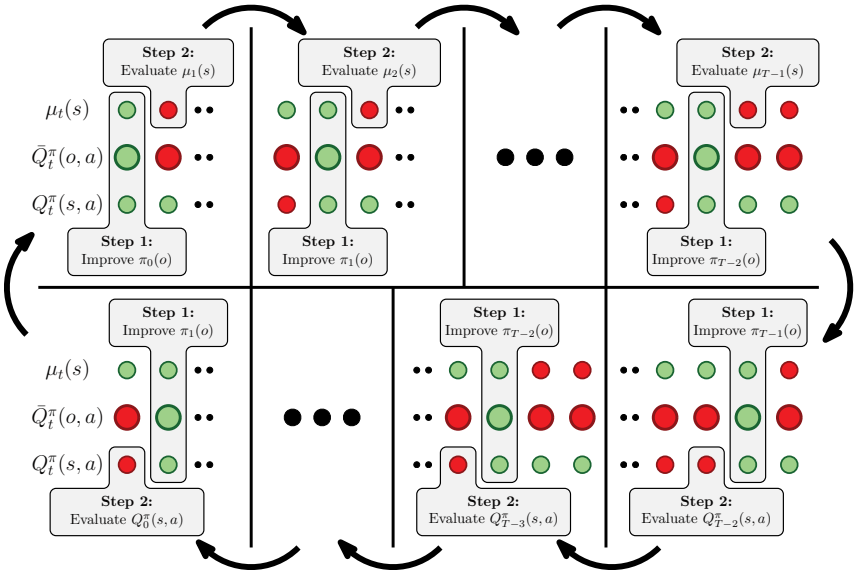


Figure 2.2. Illustration of Algorithm 1, which alternates between a forward sweep (upper row), where after each improvement step $\mu_t(s)$ is recomputed, and a backward sweep (bottom row), where after each improvement step $Q_t^\pi(s,a)$ is recomputed.

Algorithm 1 Efficient memoryless policy iteration for POMDPs

Require: Initial policy π^0, μ_0, V_T, T

Use initial policy π^0 to compute $Q_t^{\pi^0}(s,a)$ with (2.7) for $t \in \{0,\dots,T-1\}$
$\ell \leftarrow 0$
`policystable` $\leftarrow$ `false`
while `policystable` = `false` **do**
  Forward sweep
  for $t = 0$ to $T-2$ **do**
    Compute $\bar{Q}_t^{\pi}(o,a)$ with (2.8) and μ_t
    $\pi_t^{\ell+1}(o) \leftarrow \arg\max_a \bar{Q}_t^{\pi}(o,a)$
    Compute μ_{t+1} with (2.10) using $\pi_t^{\ell+1}$ and μ_t
    $\pi^{\ell+1} \leftarrow \{\pi_0^{\ell},...,\pi_{t-1}^{\ell},\pi_t^{\ell+1},\pi_{t+1}^{\ell},...,\pi_{T-1}^{\ell}\}$
    $\ell \leftarrow \ell + 1$
  end for
  Backward sweep
  for $t = T-1$ to 1 **do**
    Compute $\bar{Q}_t^{\pi}(o,a)$ with (2.8) and μ_t
    $\pi_t^{\ell+1}(o) \leftarrow \arg\max_a \bar{Q}_t^{\pi}(o,a)$
    Compute $Q_{t-1}^{\pi^{\ell+1}}(s,a)$ with (2.7) using $\pi_t^{\ell+1}$ and $Q_t^{\pi^{\ell}}(s,a)$
    $\pi^{\ell+1} \leftarrow \{\pi_0^{\ell},...,\pi_{t-1}^{\ell},\pi_t^{\ell+1},\pi_{t+1}^{\ell},...,\pi_{T-1}^{\ell}\}$
    $\ell \leftarrow \ell + 1$
  end for
  if $L^{\pi^{\ell}} = L^{\pi^{\ell-M}}$ **then**
    `policystable` $\leftarrow$ `true`
  end if
end while

2.4 Model-free policy iteration

Inspired by the findings of the previous section, we propose in this section two algorithms that learn a memoryless policy from data under two different assumptions: the first algorithm assumes that data contains the state, whereas the second algorithm assumes that it only contains observations. As a consequence, the quantities that can be directly estimated in the state-informed method are $\alpha_t(s|o)$, $Q_t^\pi(s,a)$, and $q_t(o|s)$. For the observation-only method, it is not possible to directly trace the stationary distribution $\mu_t(s)$, and thus estimate $\alpha_t(s|o)$ from data. The only quantity that can be estimated is $\bar{Q}_t^\pi(o,a)$. Table 2.1 summarizes the assumptions, estimated quantities and consequences.

The state-informed setting is realistic in scenarios where training is performed in simulation or a high-fidelity digital twin, while deployment is restricted to partial observations. This is common in robotics and autonomous systems, where full simulator state (e.g., position, velocity, or contact forces) is available during learning but not at execution time. In such cases, the proposed method naturally fits a sim-to-real workflow. The output-informed setting appears in settings where the training data is the same as deployment data and contains only the output.

The state-informed method is discussed in Section 2.4.1, whereas the observation-only method is presented in Section 2.4.2.

2.4.1 State-informed model-free policy iteration

In line with the model-based memoryless policy iteration scheme in Section 2.3.1, we propose a model-free variant of this algorithm, again parametrized by an arbitrary periodic sequence $\tau_\ell \in \mathcal{T}$, $\ell \in \{0,1,\dots,\}$ with arbitrary period M. For now, we consider a data set that contains state information, such that $Q_t^\pi(s,a)$ and $\alpha_t(s|o)$ can be estimated directly.

While to obtain $\alpha_t(s|o)$ from $\mu_t(s)$ for a given policy π^ℓ it is desirable to have on-policy data, to obtain $Q_t^{\pi^\ell}(s,a)$ we need off-policy data. Thus both quantities

Table 2.1. Summary of state-informed and observation-only model-free policy iteration.

Setting	Available Data	What can be estimated	Consequence
State-informed	States, observations, actions, rewards	$\alpha_t(s\|o)$, $Q_t^\pi(s,a)$, $q_t(o\|s)$	Requires access to the state
Observation-only	Observations, actions, rewards	$\bar{Q}_t^\pi(o,a)$	Need to recollect data after each improvement

are estimated based on off-policy data and restricting the relevant on-policy data for computing π^ℓ. We consider data sets $\mathcal{D}^\ell$, where the initial states are distributed according to the state distribution $\mu_0(s)$, and a behaviour policy $b_t(a|o)$ (e.g. an epsilon-soft version of policy π^ℓ) is applied afterwards. The data obtained with the behaviour policy is assumed to contain visits to all the state and control pairs a sufficient amount of times to obtain close predictions of $Q_t^{\pi^\ell}(s,a)$. Let these data sets be denoted by $\mathcal{D}^\ell := \{\mathcal{D}_1^\ell, \mathcal{D}_2^\ell, ..., \mathcal{D}_T^\ell\}$. For each time t, there are n_s data pairs:

$$\mathcal{D}_t^\ell := \left\{(S_{t,i}, O_{t,i}, A_{t,i}, R_{t,i}, S'_{t,i}, O'_{t,i}) | i = 1, 2, ..., n_s\right\}$$

where $S'_{t,i} = S_{t+1,i}$, $O'_{t,i} = O'_{t+1,i}$. This data set is sufficient for estimating $Q_t^{\pi^\ell}(s,a)$ and $\alpha_t(s|o)$. Before details on the estimation of these quantities are given, we define several variables that indicate the number of data samples in a data set:

$$N_t(s) = \sum_{i=1}^{n_s} \mathbb{1}\{S_{t,i} = s\}, \qquad N_t(s,o) = \sum_{i=1}^{n_s} \mathbb{1}\{S_{t,i} = s, O_{t,i} = o\},$$
$$N_t(o) = \sum_{i=1}^{n_s} \mathbb{1}\{O_{t,i} = o\}, \qquad N_t(s,a) = \sum_{i=1}^{n_s} \mathbb{1}\{S_{t,i} = s, A_{t,i} = a\}.$$

With these definitions, the observation model $q_t(o|s)$ can be estimated as $\hat{q}_t(o|s) = \frac{N_t(s,o)}{N_t(s)}$. Using bootstrapping techniques, $Q_t^\pi(s,a)$ can be estimated recursively, similar to (2.7), as

$$\hat{Q}_t^\pi(s,a) = \frac{1}{N_t(s,a)} \sum_{i=1}^{n_s} \left(\mathbb{1}\{S_{t,i} = s, A_{t,i} = a\} \left(R_{t,i} + \hat{V}_{t+1}(S'_{t,i})\right)\right) \quad (2.15)$$

where $\hat{V}_t(s) = \sum_o \hat{q}_t(o|s)\hat{Q}_t^\pi(s, \pi_t(o))$.

In addition, $\alpha_t(s|o)$ can be estimated directly using importance sampling in a recursive manner, closely related to (2.10). For the first stage, $\alpha_0(s|o)$ can be estimated as

$$\hat{\alpha}_0(s|o) = \frac{N_0(s,o)}{N_0(o)} \quad (2.16)$$

under the assumption that the data in the initial stage is distributed according to $\mu_0(s)$.

The estimated corrected state observation counter $\tilde{N}_t(s,o)$ can be computed recursively as

$$\tilde{N}_{t+1}(s',o') = \frac{\sum_{i=1}^{n_s} \mathbb{1}\{A_{t,i} = \pi_t(O_{t,i}), S'_{t,i} = s', O'_{t,i} = o'\}\tilde{N}_t(S_{t,i}, O_{t,i})}{\sum_{i=1}^{n_s} \mathbb{1}\{A_{t,i} = \pi_t(O_{t,i})\}} \quad (2.17)$$

where $\tilde{N}_0(s,o) = N_0(s,o)$. Based on these quantities, the posterior distribution $\alpha_t(s|o)$ can be estimated as

$$\hat{\alpha}_t(s|o) = \frac{\tilde{N}_t(s,o)}{\sum_s \tilde{N}_t(s,o)}. \tag{2.18}$$

Combining the estimates of $Q_t^\pi(s,a)$ and $\alpha_t(s|o)$, the observation-action value can be computed according to

$$\bar{Q}_t^\pi(o,a) = \sum_s \hat{Q}_t^\pi(s,a)\hat{\alpha}_t(s|o). \tag{2.19}$$

These estimates allow us to apply the following state-informed model-free policy iteration scheme, which parallels the model-based method as proposed in Section 2.3.1.

Given an initial deterministic policy π^0, and an M-periodic schedule $\{\tau_\ell\}$, set $\ell = 0$.

0. **Collect data $\mathcal{D}^\ell$:** Collect state, output and input trajectories using an ϵ soft version of policy π^ℓ.

1. **Policy evaluation:** Estimate $Q_t^{\pi^\ell}(s,a)$ and $\alpha_t(s|o)$ to compute $\bar{Q}_t^\pi(o,a)$ for $t = \tau_\ell$.

2. **Policy improvement:** Compute

$$\pi_t^{\ell+1}(o) = \arg\max_a \bar{Q}_t^\pi(o,a) \text{ for } t = \tau_\ell$$

and set $\pi_t^{\ell+1}(o) = \pi_t^\ell(o)$ for $t \neq \tau_\ell$. Set $\ell \to \ell + 1$ and repeat 0, 1 and 2 until the policy does not change after M consecutive policy improvement steps.

Recall from Section 2.3.1 that there are several sequences by which the policy improvement steps can be executed in a policy iteration method, all leading to (possibly different) *local* optima. An algorithm with the same efficient sequence of the model-based case alternating between forward and backward sweeps, is summarized in Algorithm 2. In the forward sweep, only $\alpha_t(s|o)$ is estimated using importance sampling, while $Q_t^\pi(s,a)$ remains unchanged. In the backward sweep $Q_t^\pi(s,a)$ is estimated while $\alpha_t(s|o)$ remains unchanged.

Algorithm 2 State-informed model-free policy iteration for POMDPs

Require: $\mathcal{D}^\ell$, π^0, N_{iter}

Use π^0 to compute $\hat{Q}_t^{\pi^0}(s,a)$ with (2.15) for $t \in \{0,\dots,T-1\}$
Compute $\hat{\alpha}_0(s|o)$ with (2.16)
Compute observation model $\hat{q}_t(o|s)$
$\ell \leftarrow 0$
for $i = 1$ to N_{iter} **do**
    Forward sweep
    for $t = 0$ to $T-2$ **do**
        Compute $\bar{Q}_t^\pi(o,a)$ with (2.19), using $\hat{Q}_t^\pi(s,a)$ and $\hat{\alpha}_t(s|o)$
        $\pi_t^{\ell+1}(o) \leftarrow \arg\max_a \bar{Q}_t^\pi(o,a)$
        Compute $\hat{\alpha}_{t+1}(s|o)$ with (2.18) using $\pi_t^{\ell+1}$ and $\hat{\alpha}_t(s|o)$
        $\pi^{\ell+1} \leftarrow \{\pi_0^\ell, ..., \pi_{t-1}^\ell, \pi_t^{\ell+1}, \pi_{t+1}^\ell, ..., \pi_{T-1}^\ell\}$
        $\ell \leftarrow \ell + 1$
    end for
    Backward sweep
    for $t = T-1$ to 1 **do**
        Compute $\bar{Q}_t^\pi(o,a)$ with (2.19), using $\hat{Q}_t^\pi(s,a)$ and $\hat{\alpha}_t(s|o)$
        $\pi_t^{\ell+1}(o) \leftarrow \arg\max_a \bar{Q}_t^\pi(o,a)$
        Compute $\hat{Q}_{t-1}^{\pi^{\ell+1}}(s,a)$ with (2.15) using $\pi_t^{\ell+1}$ and $\hat{Q}_t^{\pi^\ell}(s,a)$
        $\pi^{\ell+1} \leftarrow \{\pi_0^\ell, ..., \pi_{t-1}^\ell, \pi_t^{\ell+1}, \pi_{t+1}^\ell, ..., \pi_{T-1}^\ell\}$
        $\ell \leftarrow \ell + 1$
    end for
end for

2.4.2 Observation-only policy iteration

In the case only observations (and not state) information can be collected, the posterior distribution and the state-action value cannot be separated from each other. This however is a key feature to obtain an efficient policy iteration scheme. To still rely on the principles of monotonic convergence, developed in Section 2.3, the best one can do is to make only improvements at a single stage. The data sets used for observation-only model-free policy iteration are defined as

$$\bar{\mathcal{D}}_t^\ell := \left\{(o_{t,i}, a_{t,i}, r_{t,i}, o'_{t,i}) | i = 1, 2, ..., n_s\right\}.$$

This data set is generated by applying exploring actions at the stage considered for improvement, and on-policy actions of policy π^ℓ elsewhere. After the single-stage improvement, a complete new data set needs to be generated. In this method, the

observation-action values are estimated directly using Monte Carlo returns:

$$\bar{Q}_t^\pi(o,a) = \frac{1}{N_t(o,a)} \sum_{i=1}^{n_s} \left(\mathbb{1}\{O_{t,i} = o, A_{t,i} = a\} \sum_{k=t}^{T} R_{k,i} \right) \tag{2.20}$$

where $N_t(o,a) = \sum_{i=1}^{n_s} \mathbb{1}\{O_{t,i} = o, A_{t,i} = a\}$. The observation-only model-free policy iteration algorithm is omitted for the sake of brevity.

2.5 Numerical examples

This section empirically evaluates the proposed memoryless policy iteration algorithms. We focus on three topics aligned with Sections 2.3 and 2.4: (i) monotonic convergence behaviour, (ii) computational efficiency compared to existing methods, and (iii) performance in model-free settings. In Section 2.5.1, the proposed model-based algorithm of Section 2.3 is compared with model-based policy gradient and exhaustive search baselines. We also provide a runtime comparison with recently introduced methods. In Section 2.5.2, the proposed model-free methods are analysed. The implementation is publicly available at `https://github.com/royvanzuijlen/Memoryless-POMDPs`.

2.5.1 Model-based policy iteration

Random POMDPs are considered, which is common practice in the literature for these problems and allows us to easily scale the problem (see e.g. [69]). The first example is a small-size environment, where $|\mathcal{S}| = 20$, $|\mathcal{A}| = 2$, $|\mathcal{O}| = 4$, and $T = 6$. The second example is a mid-size environment, where $|\mathcal{S}| = 40$, $|\mathcal{A}| = 10$, $|\mathcal{O}| = 20$, and $T = 20$. A larger environment with $T = 50$, $|\mathcal{S}| = 500$, and $|\mathcal{O}| = |\mathcal{A}| = 100$ is considered later. The expected rewards $r_t(s,a)$ are also randomized.

For both examples, we compare the proposed model-based algorithm with a model-based policy gradient method. For the policy gradient method, a stochastic soft-max policy is used, defined as

$$\pi_t(a|o,\theta) = \frac{e^{\theta_{o,a,t}}}{\sum_b e^{\theta_{o,b,t}}} \tag{2.21}$$

parametrized by $\theta_{o,a,t}$, $o \in \mathcal{O}$, $a \in \mathcal{A}$, $t \in \mathcal{T}$. The objective function which is maximized is L^π. Through the dependency of L^π on parameters θ, we are able to compute the gradient $\nabla_\theta L^\pi(\theta)$. To compute the gradient we need to sweep over stages as all values of θ affect L^π. A backtracking line search method is used to ensure monotonic convergence. For the line search we used the Armijo rule which entails evaluating L^π a few times at each gradient step. Given the relatively low problem dimensions, computing the global optimal deterministic memoryless

policy is still possible via exhaustive search. The number of possible policies is $|\mathcal{A}|^{|\mathcal{O}|\cdot T}$. In fact, for the small-size environment, the number of possible policies is $2^{4\cdot 6} = 16.777.216$. For the mid-size environment however, this number is 10^{400}, which makes the global optimal policy impossible to compute.

In Figure 2.3 the speed of convergence of the proposed model-based policy iteration method is compared with the policy gradient method and the exhaustive search method. For this particular small-size example, all methods converge to the same solution, which happens to be the global optimal policy. The model-based policy iteration method converges to the optimal policy in 11 policy stage improvements. That implies that within one forward and backward sweep (as proposed in Section 2.3.2), no further improvements can be found. We count a stage improvement as a change of the action selected at a particular time t. The model-based policy gradient method improves T stages simultaneously with each gradient step. Thus, we have considered that each gradient step amounts to T stage improvement steps. Note that both the gradient method and our memoryless policy iteration need a sweep over states and stages to improve all the stages and thus such a plot is sensible. A comparison using computation times is provided in the sequel.

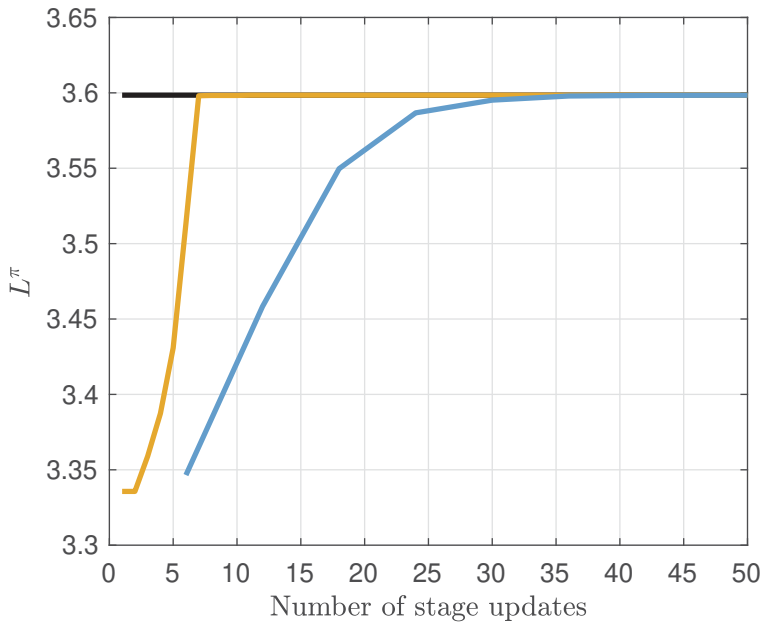


Figure 2.3. Random small-size POMDP, with $|\mathcal{S}| = 20$, $|\mathcal{A}| = 2$, $|\mathcal{O}| = 4$, and $T = 6$. Illustrated are the exhaustive search solution (—), the proposed model-based policy iteration method (—), and model-based policy gradient (—).

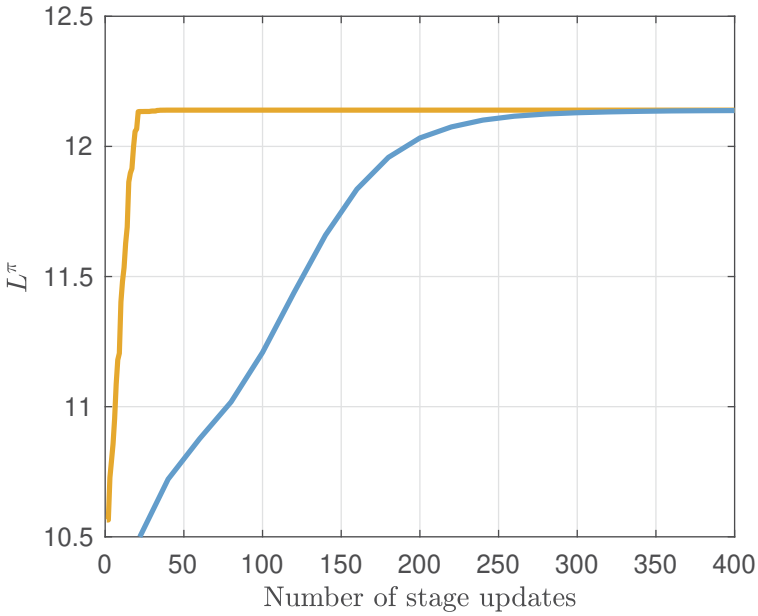


Figure 2.4. Random mid-size POMDP, with $|\mathcal{S}| = 40$, $|\mathcal{A}| = 10$, $|\mathcal{O}| = 20$, and $T = 20$. Illustrated are the proposed model-based policy iteration method (—), and model-based policy gradient (—).

From Figure 2.3 it is clear that the policy gradient method requires more stage improvements than the model-based policy iteration method until convergence.

Figure 2.4 shows the evolution of the expected episodic return versus the number of stage improvements for a mid-size POMDP. When the size of the problem increases, the difference between the policy iteration and the policy gradient method becomes even more apparent. Again, both methods converge to the same locally optimal deterministic memoryless policy. The policy iteration method however found this deterministic policy within 45 unique stage improvements, whereas the policy gradient method, after 400 unique stage improvements (or 20 gradient steps) and it is only (very) close to this policy.

We now compare the computation times of the two methods considered so far and also of the methods proposed in [20], [79]. For this we consider a POMDP, where $T = 5$ and $|\mathcal{S}| = 20$. The number of observations coincides with the number actions and varied to compare computation times. All computations are done on the same computer. For the MILP method, presented in [20], the Gurobi solver was used. The geometric method presented in [79] was solved using a standard sequential quadratic programming solver. All gradient-based methods were terminated when

the improvements were below a certain threshold. The exhaustive search method and the MILP method are only computed for $|\mathcal{O}| = |\mathcal{A}| \in \{2, 3\}$; larger values are too expensive to compute. The geometric method could be used without any problems up until $|\mathcal{O}| = |\mathcal{A}| = 16$.

The computation times are plotted in Figure 2.5. The policy iteration method significantly outperforms all the other methods. Recall that both the geometric method, as well as the policy gradient method, are not guaranteed to output a deterministic policy, whereas the policy iteration method has that guarantee. The policy iteration method usually was able to find the local optimal policy within 2 or 3 forward + backward sweeps. To illustrate the optimality of the converged policy, consider the results in Table 2.2. The table clearly indicates that the policy found with the proposed policy iteration method is at least as good as the other methods for these examples.

When the problem size increases, the policy iteration method remains superior to the policy gradient one. Consider e.g. a POMDP where $T = 20$, $|\mathcal{S}| = 200$, and

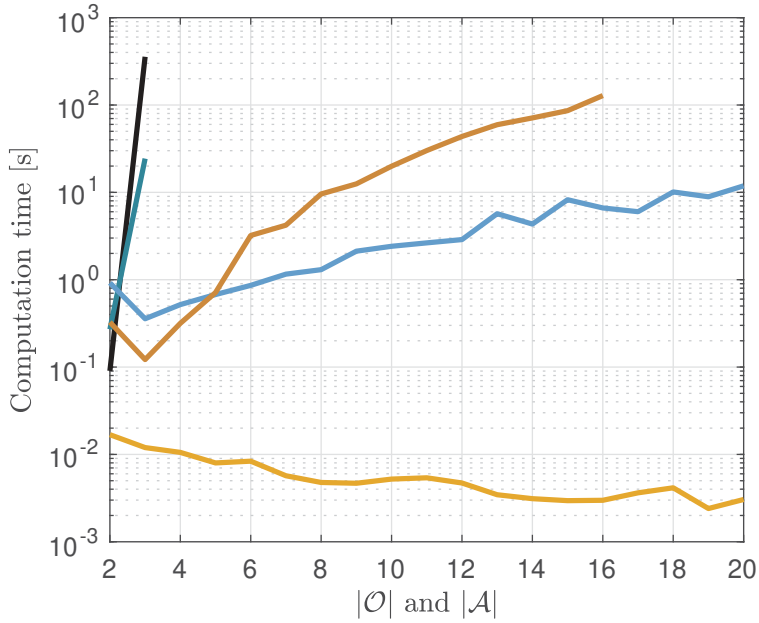


Figure 2.5. Comparison of computation time for the exhaustive search method (—), the developed model-based policy iteration method (—), a model-based policy gradient method (—), the MILP method of [20] (—), and the geometric method of [79] (—). $|\mathcal{O}|$ and $|\mathcal{A}|$ coincide, and are varied along the x-axis. Note the logarithmic scale on the y-axis.

Table 2.2. L^π after convergence to a (local) optimal policy.

$\lvert\mathcal{O}\rvert$ and $\lvert\mathcal{A}\rvert$	**Policy iteration**	**Policy gradient**	**Exhaustive search**	**[20]**	**[79]**
2	3.2125	3.2125	3.2125	3.2125	3.2125
3	3.2140	3.2140	3.2140	3.1284	3.2133
4	3.3868	3.3868	-	-	3.3865
5	3.4920	3.4920	-	-	3.4866
6	3.6060	3.6059	-	-	3.6054
7	3.5749	3.5749	-	-	3.5745
8	3.6338	3.6338	-	-	3.6338
9	3.6392	3.6392	-	-	3.6364
10	3.6378	3.6378	-	-	3.6360

$|\mathcal{O}| = |\mathcal{A}| = 50$. The policy iteration method was able to solve it in 0.7 seconds and achieve a local optimum where $L^\pi = 11.5880$. The policy gradient method on the other hand took 516 seconds to achieve the same local optimum. Even for a large POMDP, where $T = 50$, $|\mathcal{S}| = 500$, and $|\mathcal{O}| = |\mathcal{A}| = 100$, the policy iteration method found a local optimum within 15.06 seconds.

2.5.2 Model-free policy iteration

Consider again a random POMDP, where $|\mathcal{S}| = 5$, $|\mathcal{A}| = 5$, $|\mathcal{O}| = 5$, and $T = 10$. Data is collected in batches of 5000 episodes per iteration. The two proposed methods in Section 2.4 are compared with Least-Squares Policy Iteration (LSPI) [61] and policy gradient (REINFORCE algorithm [125]). Except for the state-informed memoryless policy iteration method, all the methods used for comparison and the observation-based memoryless policy iteration only have access to observations. For the state-informed model-free policy iteration, a single forward and backward sweep is executed per iteration, whereafter new data is collected using an epsilon-soft policy, where $\epsilon = 0.5$. The LSPI method estimates the Q-function batch-wise using all data, whereas the policy gradient method makes a gradient step after each episode. The proposed model-free methods and LSPI give deterministic policies by definition, whereas for policy gradient the soft-max parametrization is used, leading to a stochastic policy.

The results are given in Figure 2.6, where the performance measure L^π is averaged over 5 Monte Carlo simulations. The state-informed model-free policy iteration method leads to a memoryless, observation-based policy superior to the other methods. This result empirically supports the efficiency trade-offs discussed in Section 2.4: separating estimation of $Q_t^\pi(s, a)$ and $\alpha_t(s|o)$ enables efficient multi-stage improvement from a single data set. The observation-only model-free policy iteration method takes approximately 10 iterations to come close to give comparable

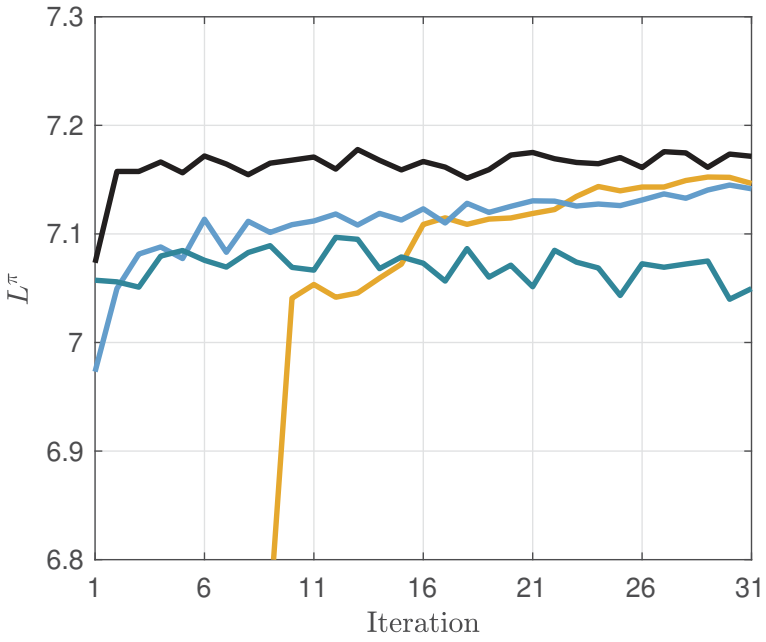


Figure 2.6. Comparison of state-informed model-free policy iteration (—), observation-only model-free policy iteration (—), LSPI (—), and policy gradient (—). After each iteration, the exact expected episodic return function of the learned policy is computed. The results are averaged over 5 Monte Carlo simulations, with 5000 episodes per iteration.

results, since then all stages have been improved at least once. For further iterations, it is rather consistent in finding an improved policy at each iteration. LSPI struggles to find a stable optimal policy in this partially observable setting, while policy gradient yields a comparably performing policy, albeit stochastic.

For the state-informed model-free policy iteration method, the data sets are generated with an epsilon-soft policy, where $\epsilon = 0.5$. By varying ϵ, the data set can be completely on-policy ($\epsilon = 0$), fully exploratory ($\epsilon = 1$), or anything in between. To estimate $Q_t^\pi(s, a)$, it is preferred to have a fully exploratory policy, to cover all states and actions and get an accurate estimate for each pair. To estimate $\alpha_t(s|o)$, however, an on-policy data set is preferred, with more data samples distributed according the state distribution of the old policy. In Figure 2.7, the effect of ϵ on the convergence is shown. Neither the extreme of fully exploratory policy ($\epsilon = 1$), nor the almost fully on-policy ($\epsilon = 0.01$), is preferred. The best results are obtained with intermediate values of ϵ.

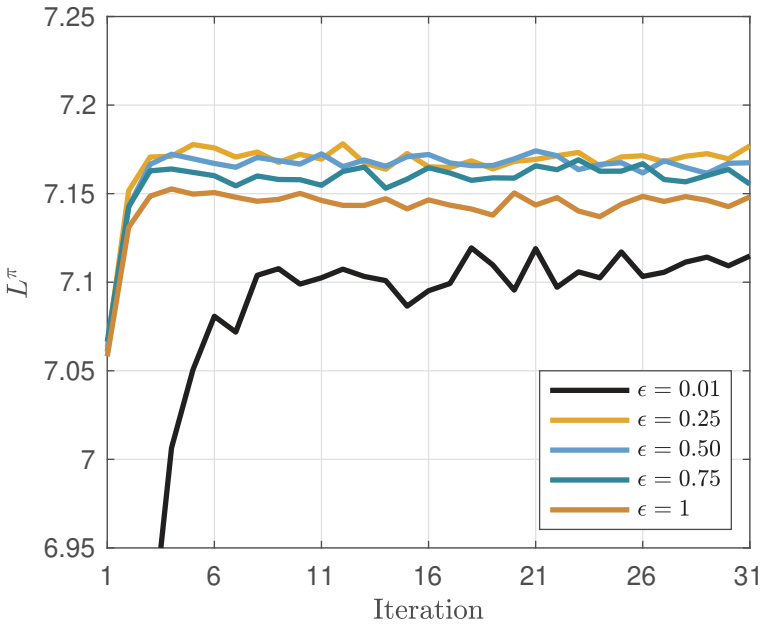


Figure 2.7. Effect of ϵ in epsilon-soft behaviour policy. The results are averaged over 20 Monte Carlo simulations, with 5.000 episodes per iteration.

2.6 Conclusions and future work

This chapter tackles the challenge of computing deterministic memoryless policies for episodic POMDPs from the perspective of policy iteration. We introduced a general class of memoryless policy iteration algorithms that perform single-stage output-based improvements according to a prescribed periodic schedule, and proved that any onto schedule yields monotonic improvement and convergence to a locally optimal memoryless policy. This establishes, for the episodic setting, a principled analogue of classical policy iteration that operates directly in the observation space despite the absence of the Markov property. Beyond monotonic convergence, we characterized the computational implications of different schedules and identified a unique (up to cyclic shift) schedule with minimal period that optimally balances forward and backward propagation of the state distribution and the state-action function.

The resulting algorithm retains the structure and interpretability of policy iteration while achieving substantial computational advantages over existing approaches, including policy gradient methods and recent specialized solvers for memoryless POMDP policies. Empirically, the proposed method consistently reaches local

optima using much less computations.

Based on the findings on the model-based policy iteration method, two model-free policy iteration methods are proposed. The first one is state-informed model-free policy iteration, where, during learning, the state of the POMDP is part of the data set. This leads to a rather efficient policy iteration scheme, where a single off-policy data set can improve the policy at all stages. The second method is observation-only model-free policy iteration, where there is no access to the underlying state; only observations are in the data set. In this method, after each single-stage policy improvement step, a new on-policy data set has to be gathered. The proposed model-free policy iteration methods are compared with LSPI and policy gradient. The state-informed method significantly outperforms the other methods with the same data set size. The observation-only method achieves eventually a better performing policy than conventional methods, although the rate of convergence is rather slow and thus a large amount of data is required before a satisfying performance level is achieved.

Directions for future research include extensions to infinite-horizon and continuous state, action and observation spaces. Furthermore, extensions to approximate policy iteration methods will allow handling much larger problems.

2.7 Appendix

2.7.1 Proof of Theorem 2.1

We first note that L^π can be written as:

$$\begin{aligned} L^\pi &= \mathbb{E}\left[\sum_{t=0}^{T-1} R_t + V_T(s_T)\right] \\ &= \sum_o \gamma_0(o) \sum_a \pi_0(a|o) \bar{Q}_0^\pi(o,a) \\ &= \sum_s \mu_0(s) \sum_o q_0(o|s) \sum_a \pi_0(a|o) Q_0^\pi(s,a). \end{aligned} \tag{2.22}$$

If $\tau_{\ell+1} = 0$, it is obvious that

$$\begin{aligned} L^{\pi^{\ell+1}} &= \sum_o \gamma_0(o) \sum_a \pi_0^{\ell+1}(a|o) \bar{Q}_0^{\pi^{\ell+1}}(o,a) \\ &\geq \sum_o \gamma_0(o) \sum_a \pi_0^{\ell}(a|o) \bar{Q}_0^{\pi^{\ell}}(o,a) \\ &= L^{\pi^\ell} \end{aligned} \tag{2.23}$$

since $\gamma_0(o)$ is independent of the policy applied at $t = 0$ and $\bar{Q}_0^{\pi^{\ell+1}}(o,a) = \bar{Q}_0^{\pi^\ell}(o,a)$ only depends on the policy applied at $t > 0$.

If the policy is improved at $\tau_{\ell+1} \neq 0$, we can unroll (2.22) up until $k = \tau_{\ell+1}$, such that we obtain:

$$\begin{aligned}
L^{\pi^{\ell+1}} &= \sum_{i=0}^{k-1} \sum_{s} \mu_i(s) \sum_{o} q_i(o|s) \sum_{a} \pi_i^{\ell+1}(a|o) r_i(s,a) \\
&\quad + \sum_{s'} \mu_k(s') \sum_{o'} q_k(o'|s') \sum_{a'} \pi_k^{\ell+1}(a'|o') \bar{Q}_k^{\pi^{\ell+1}}(s',a') \\
&= \sum_{i=0}^{k-1} \sum_{s} \mu_i(s) \sum_{o} q_i(o|s) \sum_{a} \pi_i^{\ell+1}(a|o) r_i(s,a) \\
&\quad + \sum_{o} \gamma_k(o) \sum_{a} \pi_k^{\ell+1}(a|o) \bar{Q}_k^{\pi^{\ell+1}}(o,a) \\
&= \sum_{i=0}^{k-1} \sum_{s} \mu_i(s) \sum_{o} q_i(o|s) \sum_{a} \pi_i^{\ell}(a|o) r_i(s,a) \\
&\quad + \sum_{o} \gamma_k(o) \sum_{a} \pi_k^{\ell+1}(a|o) \bar{Q}_k^{\pi^{\ell+1}}(o,a) \\
&\geq \sum_{i=0}^{k-1} \sum_{s} \mu_i(s) \sum_{o} q_i(o|s) \sum_{a} \pi_i^{\ell}(a|o) r_i(s,a) \\
&\quad + \sum_{o} \gamma_k(o) \sum_{a} \pi_k^{\ell}(a|o) \bar{Q}_k^{\pi^{\ell}}(o,a) \\
&= L^{\pi^{\ell}}.
\end{aligned} \tag{2.24}$$

This follows from the fact that the policy remains unchanged for $t < \tau_{\ell+1}$: $\pi_i^{\ell+1} = \pi_i^{\ell}$ for $i \in \{0, ..., \tau_{\ell+1} - 1\}$. Furthermore, $\bar{Q}_k^{\pi^{\ell+1}}(o,a) = \bar{Q}_k^{\pi^{\ell}}(o,a)$, since these values are independent on the policy at $k = \tau_{\ell+1}$. The inequality in (2.24) follows directly from the optimization problem. The last equality in (2.24) only holds when the values of $\bar{Q}_k^{\pi^{\ell}}(o,a)$ are aligned with (old) policy π^{ℓ}, which the class of algorithms ensures each improvement step. Therefore each improvement step in the class of algorithms monotonically increases the expected episodic return.

To continue the proof of convergence to a local optimal policy, let the expected episodic return after the j'th improvement be denoted by $L^{\pi^{(j)}}$. Then,

$$L^{\pi^{(0)}} \leq L^{\pi^{(1)}} \leq ... \leq L^{\pi^*}. \tag{2.25}$$

This sequence is bounded from above by the global optimal policy π^*. Let the period of $\{\tau_\ell\}$ be given as M. Since the number of policies is finite and the cost is upper and lower bounded and decreases monotonically, the cost must converge to a limit value. In particular, there exists m such that

$$L^{\pi^{(m)}} = L^{\pi^{(m+1)}} = ... = L^{\pi^{(m+M)}} \tag{2.26}$$

for the given M. Since the sequence $\{\tau_\ell\}$ is onto, no single stage can be improved further, and thus a local optimal policy has been found. ■

Chapter 3

Policy Iteration for Continuous POMDPs

Abstract - Finding optimal output-based policies for a continuous partially observable Markov decision process (POMDP) is challenging due to the non-Markovian nature of the output and the intractably large belief space. In this chapter, we extend our recent memoryless policy iteration framework to continuous state and control spaces, providing a deterministic algorithm that iteratively produces policies with monotonic cost improvement. Such a framework relies on a forward iteration, to propagate the state probability distribution, and a backward iteration, to propagate the value function. Similar to existing methods in literature, such as policy gradient, we show convergence to locally optimal policies. We apply the method to the static output feedback LQG problem, showing that the resulting forward and backward iterations reduce to two dual Riccati-like equations, retaining the monotone cost improvement properties of the finite-horizon setting. For infinite-horizon average-cost problems, we prove converge of the Riccati-like equations under mild assumptions to the solution of Algebraic Riccati-like equations when the state dimension is one. For general n-dimensional systems, convergence is observed empirically, although formal guarantees warrant further research. Numerical examples further demonstrate that our approach converges monotonically to a locally optimal policy, offering a practical alternative for continuous partially observable control.

This chapter is based on: R.A.C. van Zuijlen, D.J. Antunes, "Memoryless Policy Iteration for Continuous POMDPs: Application to Static Output Feedback LQG", *(submitted)*.

3.1 Introduction

Finding optimal policies for partially observable Markov decision processes (POMDPs) is, in general, computationally intractable. For finite state, action and observation spaces, the problem is PSPACE-complete [83]. In the finite-horizon case, the optimal value function over the belief simplex is piecewise linear, but its complexity grows exponentially in the horizon [109]. For continuous POMDPs the belief space becomes infinite dimensional [90], and the value function must be treated as a functional over probability measures, which significantly complicates both analysis and computation. As a result, exact solution methods are typically infeasible, and existing approaches rely heavily on approximation or structural restrictions [111]. Approximate methods include point-based value iteration [90], Gaussian processes [23], symbolic dynamic programming [132], and lazy cross-entropy search over policy trees [47].

In light of this intractability, a common approach is to restrict attention to memoryless or finite memory policies, which map current observations (or a finite window of past observations) directly to actions without maintaining an explicit belief state. Although such policies are generally suboptimal, they offer substantial computational advantages. However, when formulated purely in the observation space, the resulting control problem becomes non-Markovian, and computing an optimal memoryless policy is itself NP-hard [68].

In recent work, we introduced a tractable policy-search approach for finite space episodic POMDPs that computes suboptimal memoryless controllers via policy iteration. We have shown that this approach is computationally more efficient than policy gradient in several numerical examples (Chapter 2). Nevertheless, for continuous POMDPs, principled and scalable policy iteration schemes for memoryless policies remain an open problem. We address such an extension in the present chapter, motivated by the fact that policy iteration is typically computationally more efficient than policy gradient.

One setting in which memoryless policies arise in continuous POMDPs is linear–quadratic–Gaussian (LQG) systems with static output feedback. Unlike the fully observable case, restricting the control law to depend only on the current output leads to a fundamentally non-convex optimization problem, which is NP-hard in general [14]. This problem has been studied for several decades, particularly in continuous time; see, for example [65, 117] and the references therein. In discrete time and considering a finite-horizon, [32] proposed a method that alternates between *complete* policy evaluation and *complete* policy improvement, but lacks guarantees of convergence to a local optimum. Moreover, [72] also considered the finite-horizon setting, but restricted to time-invariant control policies. For the infinite-horizon case, existing methods are primarily gradient-based. Even in the single-input single-output setting, where the controller gain is a scalar value, the

set of stabilizing controllers can be disconnected [50]. The paper [130] iteratively solves two Lyapunov equations and performs gain updates with step sizes chosen to preserve closed-loop stability, though convergence guarantees are not provided. In [29, 78] analytical gradient expressions are derived and policy-gradient methods are used to converge to local optima. Moreover, [28] employs approximate dynamic programming but restricts attention to deterministic systems.

In this chapter, we propose a memoryless policy iteration scheme for continuous POMDPs with a finite-horizon, extending the previous chapter on finite state, action, and observation spaces (Chapter 2). Under mild assumptions, we prove that the proposed method converges to a locally optimal policy. We then specialize the approach to LQG systems, for which the required computations are tractable. In contrast to existing methods for static output feedback control design, the resulting updates do not rely on gradients. Furthermore, we extend the scheme to infinite-horizon LQG problems. When the state dimension is one, we establish convergence under mild assumptions of the Riccati-like equations to the solution of an Algebraic Riccati equation. For general n-dimensional systems convergence is observed empirically; providing formal guarantees in this case remains an open problem.

The remainder of the chapter is organized as follows. The problem is formulated in Section 3.2. Section 3.3 proposes the main algorithm and the main results for the finite-horizon setting. The application to static output feedback LQG systems is given Section 3.4 Section 3.5 discusses the extension to the infinite-horizon setting. Section 3.6 presents numerical examples. Section 3.7 discusses future work and provides final remarks.

Notation: $\rho(A) = \max(|\lambda_i|)$, where λ_i are the eigenvalues of matrix A. $\mathrm{Tr}(\cdot)$ denotes the trace of a matrix.

3.2 Problem formulation

Consider a process

$$x_{k+1} = f_k(x_k, u_k, w_k) \tag{3.1a}$$

$$y_k = h_k(x_k, v_k) \tag{3.1b}$$

where $x_k \in \mathcal{X}_k$ is the state, $u_k \in \mathcal{U}_k$ is the control input, and $y_k \in \mathcal{Y}_k$ is the process output. The process disturbance input and the measurement noise sequences are denoted by $w_k \in \mathcal{W}_k$ and $v_k \in \mathcal{V}_k$ for $k \in \mathbb{N}_0 := \mathbb{N} \cup \{0\}$ respectively, and are mutually independent sequences. For each k, the sets $\mathcal{X}_k$, $\mathcal{U}_k$, $\mathcal{Y}_k$, $\mathcal{W}_k$, and $\mathcal{V}_k$ are nonempty Borel spaces. Furthermore, f_k and h_k are Borel-measurable functions. The state process is Markovian conditioned on controls, and observations satisfy conditional independence: $\mathrm{Prob}[y_k|x_{0:k}, u_{0:k-1}] = \mathrm{Prob}[y_k|x_k]$. Throughout the

first part of the chapter we consider finite-horizon $N \in \mathbb{N}$ and define the stage index set as $\mathcal{K} = \{0, 1, ..., N-1\}$. The state is assumed to have an initial distribution p_0:

$$p_0(A) := \text{Prob}\left[x_0 \in A\right], \quad A \subseteq \mathcal{X}_0. \tag{3.2}$$

Consider the following cost

$$J = \mathbb{E}\left[\sum_{k=0}^{N-1} g_k(x_k, y_k, u_k) + g_N(x_N, y_N)\right] \tag{3.3}$$

for some horizon N. The goal of this chapter is to find a policy $\pi := \{\mu_0, \ldots, \mu_{N-1}\}$ that minimizes J, where μ_k maps outputs y_k into controls:

$$u_k = \mu_k(y_k), \quad k \in \mathcal{K}. \tag{3.4}$$

We restrict attention to admissible memoryless (output feedback) policies for which each mapping $\mu_k : \mathcal{Y}_k \to \mathcal{U}_k$ is Borel-measurable and such that all expectations appearing in the cost function are finite. The cost of policy π is therefore defined as

$$J_\pi = \mathbb{E}\left[\sum_{k=0}^{N-1} g_k(x_k, y_k, \mu_k(y_k)) + g_N(x_N, y_N)\right]. \tag{3.5}$$

We assume that the stage costs g_k are lower semianalytic functions and integrable with respect to the distributions induced by any admissible policy.

We will also consider infinite-horizon problems where cost (3.5) is replaced by

$$J_{\text{avg}} = \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} \mathbb{E}\left[g_k(x_k, y_k, \mu(y_k))\right]. \tag{3.6}$$

In this setting, we restrict the policy search to time-invariant policies, i.e., $\pi := \{\mu, \mu, ..., \mu\}$.

3.3 Policy iteration for general continuous spaces

As in standard policy iteration, the proposed algorithm produces a sequence of policies $\pi^{(0)}$, $\pi^{(1)}$, $\pi^{(2)}$, ..., that decrease the cost defined as (3.5). To introduce such a policy, we need some preliminaries.

For any admissible policy π, the closed-loop system induces a joint probability distribution over (x_k, y_k) at every stage k. The closed-loop transition kernel of the process under a policy $\pi^{(j)}$ is denoted by $F_k^{(j)}(\xi, A) = \text{Prob}[x_{k+1} \in A | x_k = \xi]$.

The observation kernel $H_k(\xi, A) = \text{Prob}[y_k \in A | x_k = \xi]$ is assumed to admit a probability density $q_y(\xi, y)$ with respect to Lebesgue measure on $\mathcal{Y}_k$.

Let $p_k^{(j)} : \mathcal{X}_k \to \mathbb{R}_{\geq 0}$ denote the probability measure of x_k under policy $\mu_k^{(j)}$. It can be computed recursively from $k = 0$ until $N - 1$ as

$$p_{k+1}^{(j)}(A) = \int_{\mathcal{X}_k} F_k^{(j)}(\xi, A) p_k^{(j)}(d\xi), \quad k \in \mathcal{K}. \tag{3.7}$$

The corresponding observation distribution $\gamma_k^{(j)}(A) = \text{Prob}[y_k \in A]$ is defined by

$$\gamma_k^{(j)}(A) = \int_{\mathcal{X}_k} \left(\int_A q_y(\xi, y) dy \right) p_k^{(j)}(d\xi). \tag{3.8}$$

Moreover, let $\zeta_k^{\mathsf{y}} : \mathcal{X}_k \to \mathbb{R}_{\geq 0}$ denote the state probability measure conditioned on measurement $y_k = \mathsf{y}$, i.e., $\zeta_k^{\mathsf{y}}(A) = \text{Prob}[x_k \in A | y_k = \mathsf{y}]$. By Bayes' rule it can be computed as follows

$$\zeta_k^{\mathsf{y}}(A) = \alpha \int_A q_y(\xi, y) p_k(d\xi) \tag{3.9}$$

where α is a normalization constant. We assume that $\int_A q_y(\xi, y) p_k(d\xi) > 0$ for all $y \in \mathcal{Y}_k$, ensuring that the posterior ζ_k^{y} is well defined.

For a given policy $\pi^{(j)}$, define the state-action value recursively as

$$\begin{aligned} &Q_k^{\pi^{(j)}}(x, u) = \\ &\mathbb{E}[g_k(x_k, y_k, u_k) + Q_{k+1}^{\pi^{(j)}}(x_{k+1}, \mu_{k+1}^{(j)}(y_{k+1})) | x_k = x, u_k = u] \end{aligned} \tag{3.10}$$

where the conditional expectation is taken with respect to w_k and v_{k+1}. Then the observation-action value can be computed as

$$\begin{aligned} \bar{Q}_k^{\pi^{(j)}}(y, u) &= \mathbb{E}[Q_k^{\pi^{(j)}}(x_k, u_k) | y_k = y, u_k = u] \\ &= \int_{\mathcal{X}_k} Q_k^{\pi^{(j)}}(\xi, u) \zeta_k^{\mathsf{y}}(d\xi) \end{aligned} \tag{3.11}$$

where ζ_k^{y} is defined in (3.9). The integral in (3.11) is well defined by measurability and integrability of $Q_k^{\pi^{(j)}}$ and the posterior distribution ζ_k^{y}.

The proposed policy iteration algorithm is an extension of our previous memoryless policy iteration algorithm (Algorithm 1 in Chapter 2) to continuous state, control and observation spaces. The algorithm is parametrized by a periodic sequence $\kappa_j \in \mathcal{K}$, $j \in \{0, 1, 2, ...\}$ with arbitrary period Λ. As standard policy iteration, the proposed algorithm has two steps: policy evaluation and policy improvement. We assume throughout that for every k and y, the minimization of $\bar{Q}_k^{\pi^{(j)}}(y_k, u_k)$ over $u_k \in \mathcal{U}_k$ admits at least one solution, for instance because $\mathcal{U}_k$ is compact and $\bar{Q}_k^{\pi^{(j)}}(y, u)$ is continuous in u.

Given an initial policy $\pi^{(0)}$ and a Λ-periodic sequence κ_j, set $j = 0$.

1. **Policy evaluation:** Compute $\bar{Q}_k^{\pi^{(j)}}(y_k, u_k)$ for $k = \kappa_j$.

2. **Policy improvement:** Compute

$$\mu_k^{(j+1)}(y_k) \in \arg\min_{u_k} \bar{Q}_k^{\pi^{(j)}}(y_k, u_k) \tag{3.12}$$

and set $\mu_k^{(j+1)}(y_k) = \mu_k^{(j)}(y_k)$ for $k \neq \kappa_j$. Set $j \to j+1$ and repeat 1 and 2 until the policy converges.

We assume that the periodic sequence κ_i is onto, ensuring that every stage is eventually updated, and that $\kappa_{i+1} \neq \kappa_i$, as two consecutive policy improvements at the same stage would be redundant. Under these assumptions, the proposed policy iteration algorithm converges to a locally optimal memoryless policy, where

$$\mu_k^*(y) \in \arg\min_{u} \bar{Q}_k^{\pi^*}(y, u), \quad \forall k \in \mathcal{K}. \tag{3.13}$$

Theorem 3.1. *Let $\pi^{(j)}$ be the policy produced by memoryless policy iteration under an arbitrary periodic onto sequence κ_j satisfying $\kappa_{i+1} \neq \kappa_i$, and denote its expected episodic cost by $J_{\pi^{(j)}}$ as in* (3.5)*. Then*

$$J_{\pi^{(j+1)}} \leq J_{\pi^{(j)}} \text{ for all } j \in \{0, 1, 2, ...\} \tag{3.14}$$

and the policy converges to a local optimum. ▲

The proof can be found in Section 3.8.1, and relies on standard interchange of integrals and expectations, which is justified by the assumed integrability of the cost functions and boundedness of the induced value functions.

In Chapter 2 it is argued that from a computational perspective the sequence

$$\kappa = \{0, 1, ..., N-2, N-1, N-2, ..., 1, 0, 1, ...\} \tag{3.15}$$

is the most efficient. The same arguments as in Chapter 2 hold for continuous spaces and therefore we consider sequence (3.15) hereafter.

3.4 Application to static output feedback

In this section we specialize the general memoryless policy iteration framework of Section 3.3 to LQG systems with static output feedback. In particular, we derive closed-form expressions for the state-action value function $Q_k^\pi(x, u)$ in (3.10) and the corresponding observation-action value $\bar{Q}_k^\pi(y, u)$ in (3.11) for a class of policies that is closed under the operations of the proposed algorithm, and which allow the policy improvement step to be carried out analytically.

Consider the following system

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{3.16a}$$
$$y_k = Cx_k + v_k \tag{3.16b}$$

where $x_k \in \mathbb{R}^{n_x}$, $u_k \in \mathbb{R}^{n_u}$, and $y_k \in \mathbb{R}^{n_y}$. The disturbances w_k, v_k are zero mean and independent with covariance $\mathbb{E}[w_k w_k^\intercal] = W$, $\mathbb{E}[v_k v_k^\intercal] = V$, respectively. We assume the initial state is Gaussian with an arbitrary mean and covariance $x_0 \sim \mathcal{N}(\bar{x}_0, \Theta_0)$. Moreover, we consider the following cost

$$\mathbb{E}[\sum_{k=0}^{N-1} x_k^\intercal Q x_k + u_k^\intercal R u_k + x_N^\intercal Q_N x_N] \tag{3.17}$$

where Q, R, Q_N are positive semi-definite. We wish to find a policy $u_k = \mu_k(y_k)$ that minimizes this cost, based on the algorithm described in Section 3.3.

We now show that if we consider a policy

$$u_k = K_k y_k + \beta_k \tag{3.18}$$

and try to improve such a policy at a given stage $\ell \in \mathcal{K}$ by considering

$$u_k = \begin{cases} K_k y_k + \beta_k, \text{ if } k \neq \ell \\ \tilde{u}_\ell, \text{ if } k = \ell \end{cases} \tag{3.19}$$

we obtain a policy taking the same form as (3.18), but with different parameters at stage ℓ, that is K_ℓ and β_ℓ.

Plugging in the policy (3.19) into (3.16) we obtain

$$x_{k+1} = (A + BK_kC)x_k + B\beta_k + w_k + BK_k v_k, \quad k \neq \ell, \tag{3.20}$$
$$x_{k+1} = Ax_k + B\tilde{u}_k + w_k, \quad k = \ell. \tag{3.21}$$

By taking the expected value on both sides we obtain also a recursion for $\bar{x}_k := \mathbb{E}[x_k]$:

$$\bar{x}_{k+1} = \mathcal{T}_{\bar{x}}(\bar{x}_k, K_k, \beta_k) := (A + BK_kC)\bar{x}_k + B\beta_k, \quad k \neq \ell, \tag{3.22}$$
$$\bar{x}_{k+1} = A\bar{x}_k + B\mathbb{E}[\tilde{u}_k], \quad k = \ell. \tag{3.23}$$

By defining the covariance as

$$\Theta_k = \mathbb{E}[(x_k - \bar{x}_k)(x_k - \bar{x}_k)^\intercal] \tag{3.24}$$

we obtain

$$\begin{aligned} \Theta_{k+1} &= \mathcal{T}_\Theta(\Theta_k, K_k) \\ &:= (A + BK_kC)\Theta(A + BK_kC)^\intercal + W + BK_kVK_k^\intercal B^\intercal \end{aligned} \tag{3.25}$$

for $k < \ell$.

We can show by induction that the cost (3.17) after and including time $k = \ell$ can be written as

$$\mathbb{E}[\sum_{k=\ell}^{N-1} x_k^\intercal Q x_k + u_k^\intercal R u_k + x_N^\intercal Q_N x_N | x_\ell] = x_\ell^\intercal P_\ell x_\ell + 2 S_\ell x_\ell + c_\ell. \tag{3.26}$$

In fact this is true for $\ell = N$ with $P_N = Q_N$, $S_N = 0$, $c_N = 0$ and assuming it is true for a given $\ell + 1$, we obtain

$$\begin{aligned}
&\mathbb{E}[\sum_{k=\ell}^{N-1} x_k^\intercal Q x_k + u_k^\intercal R u_k + x_N^\intercal Q_N x_N | x_\ell] \\
&\quad = \mathbb{E}[x_\ell^\intercal Q x_\ell + u_\ell^\intercal R u_\ell + x_{\ell+1}^\intercal P_{\ell+1} x_{\ell+1} + 2 S_{\ell+1} x_{\ell+1} + c_{\ell+1} | x_\ell] \\
&\quad = \mathbb{E}[x_\ell^\intercal Q x_\ell + (K_\ell C x_\ell + K_\ell v_\ell + \beta_\ell)^\intercal R (K_\ell C x_\ell + K_\ell v_\ell + \beta_\ell) | x_\ell] \\
&\qquad\quad + \mathbb{E}[2 S_{\ell+1}((A + B K_\ell C) x_\ell + B \beta_\ell + w_\ell + B K_\ell v_\ell) | x_\ell] \\
&\qquad\quad + \mathbb{E}[((A + B K_\ell C) x_\ell + B \beta_\ell)^\intercal P_{\ell+1} ((A + B K_\ell C) x_\ell + B \beta_\ell) | x_\ell] \\
&\qquad\quad + \operatorname{Tr}(P(W + B K_k V (B K_k)^\intercal)) \\
&\quad = x_\ell^\intercal P_\ell x_\ell + 2 S_\ell x_\ell + c_\ell
\end{aligned} \tag{3.27}$$

where

$$\begin{aligned}
P_\ell &= \mathcal{T}_P(P_{\ell+1}, K_\ell) \\
&:= Q + C^\intercal K_\ell^\intercal R K_\ell C + (A + B K_\ell C)^\intercal P_{\ell+1} (A + B K_\ell C), \\
S_\ell &= \mathcal{T}_S(P_{\ell+1}, S_{\ell+1}, K_\ell, \beta_\ell) \\
&:= \beta_\ell^\intercal R K_\ell C + S_{\ell+1}(A + B K_\ell C) + \beta_\ell^\intercal B^\intercal P_{\ell+1} (A + B K_\ell C), \\
c_\ell &= \mathcal{T}_c(P_{\ell+1}, S_{\ell+1}, c_{\ell+1}, K_\ell, \beta_\ell) \\
&:= c_{\ell+1} + \beta_\ell^\intercal (R + B^\intercal P_{\ell+1} B) \beta_\ell + \operatorname{Tr}(R K_\ell V K_\ell^\intercal) \\
&\qquad + 2 S_{\ell+1} B \beta_\ell + \operatorname{Tr}(P_{\ell+1}(W + B K_\ell V (B K_\ell)^\intercal)).
\end{aligned} \tag{3.28}$$

In the terminology of Section 3.3, this cost expression can be used to compute the state-action value function $Q_\ell^\pi(x_\ell, u_\ell)$ for the policy (3.19). The goal is to study the influence on the cost from $k = \ell$ onwards on $\tilde{u}_\ell$ so that optimal decisions can

be made. Actually, we can compute $\bar{Q}_\ell^\pi(y_\ell, u_\ell)$ directly:

$$\begin{aligned}
&\mathbb{E}[\sum_{k=\ell}^{N-1} x_k^\intercal Q x_k + u_k^\intercal R u_k + x_N^\intercal Q_N x_N | y_\ell] \\
&\quad = u_\ell^\intercal R u_\ell + \mathbb{E}[x_\ell^\intercal Q x_\ell | y_\ell] \\
&\qquad + \mathbb{E}[(Ax_\ell + Bu_\ell + w_\ell)^\intercal P_{\ell+1}(Ax_\ell + Bu_\ell + w_\ell) | y_\ell] \\
&\qquad + \mathbb{E}[2S_{\ell+1}(Ax_\ell + Bu_\ell + w_\ell) | y_\ell] + q_{\ell+1} \\
&\quad = u_\ell^\intercal (R + B^\intercal P_{\ell+1} B) u_\ell + \mathbb{E}[x_\ell^\intercal (Q + A^\intercal P_{\ell+1} A) x_\ell | y_\ell] \\
&\qquad + 2u_\ell^\intercal B^\intercal P_{\ell+1} A \mathbb{E}[x_\ell | y_\ell] + 2S_{\ell+1} A \mathbb{E}[x_\ell | y_\ell] + 2S_{\ell+1} B u_\ell + c_{\ell+1}.
\end{aligned} \tag{3.29}$$

The right-hand side of (3.29) is precisely the observation-action value $\bar{Q}_\ell^\pi(y_\ell, u_\ell)$ defined in (3.11), specialized to the LQG setting and expressed in terms of the conditional mean $\mathbb{E}[x_\ell | y_\ell]$. The minimizer of this expression is

$$u_\ell = G_\ell \mathbb{E}[x_\ell | y_\ell] + Z_\ell, \tag{3.30}$$

with

$$G_\ell = -(R + B^\intercal P_{\ell+1} B)^{-1} B^\intercal P_{\ell+1} A, \tag{3.31}$$

$$Z_\ell = -(R + B^\intercal P_{\ell+1} B)^{-1} B^\intercal S_{\ell+1}^\intercal. \tag{3.32}$$

Note that

$$\mathbb{E}[x_k | y_k] = \bar{x}_k + L_k(y_k - C\bar{x}_k) \tag{3.33}$$

with $L_k = \Theta_k C^\intercal (C\Theta_k C^\intercal + V)^{-1}$. Thus we can write

$$u_\ell = K_\ell y_\ell + \beta_\ell, \tag{3.34}$$

where

$$K_\ell = \mathcal{F}_K(P_{\ell+1}, \Theta_\ell) := G_\ell L_\ell, \tag{3.35}$$

$$\beta_\ell = \mathcal{F}_\beta(P_{\ell+1}, S_{\ell+1}, \Theta_\ell, \bar{x}_\ell) := G_\ell(I - L_\ell C)\bar{x}_\ell + Z_\ell. \tag{3.36}$$

Note that, as expected, the parameters of the optimized policy, namely β_ℓ, depend on the parameters of the control policy from the initial time until time ℓ, namely on the parameters β_k, $k \in \{0, 1, \ldots, \ell - 1\}$ (through the dependency of $\bar{x}_\ell$ and Θ_ℓ on these parameters). We conclude that the optimized control law also takes the form (3.18).

Based on all this we can recognize the elements that determine the expected value of a state given a measurement ($\bar{x}_k$ and Θ_k). These have to be computed in

a forward manner. And the elements that determine the Q-value: P_k, S_k and c_k, which have to be computed in a backward manner. These quantities correspond exactly to the two components of policy iteration in Section 3.3: the forward sweep computes the statistics required to evaluate the conditional expectation in (3.11), while the backward sweep evaluates the quadratic representation of Q_k^π in (3.10). The method in this section is summarized in Algorithm 3.

Algorithm 3 Policy iteration

Require: $K_k^{(0)}$, $\beta_k^{(0)}$, $k \in \mathcal{K}$, $\bar{x}_0$, Θ_0

Compute P_k, S_k, and c_k for $k \in \mathcal{K}$
$j \leftarrow 0$
`policystable` $\leftarrow$ `false`
while `policystable` = `false` **do**
  Forward sweep
  for $k = 0$ to $N-2$ **do**
    $K_k^{(j+1)} \leftarrow \mathcal{F}_K(P_{k+1}, \Theta_k)$
    $\beta_k^{(j+1)} \leftarrow \mathcal{F}_\beta(P_{k+1}, S_{k+1}, \Theta_k, \bar{x}_k)$
    $\Theta_{k+1} \leftarrow \mathcal{T}_\Theta(\Theta_k, K_k^{(j+1)})$
    $\bar{x}_{k+1} \leftarrow \mathcal{T}_{\bar{x}}(\bar{x}_k, K_k^{(j+1)}, \beta_k^{(j+1)})$
  end for
  Backward sweep
  for $k = N-1$ to 1 **do**
    $K_k^{(j+1)} \leftarrow \mathcal{F}_K(P_{k+1}, \Theta_k)$
    $\beta_k^{(j+1)} \leftarrow \mathcal{F}_\beta(P_{k+1}, S_{k+1}, \Theta_k, \bar{x}_k)$
    $P_k \leftarrow \mathcal{T}_P(P_{k+1}, K_k^{(j+1)})$
    $S_k \leftarrow \mathcal{T}_S(P_{k+1}, S_{k+1}, K_k^{(j+1)}, \beta_k^{(j+1)})$
    $c_k \leftarrow \mathcal{T}_c(P_{k+1}, S_{k+1}, c_{k+1}, K_k^{(j+1)}, \beta_k^{(j+1)})$
  end for
  if $\|K_k^{(j+1)} - K_k^{(j)}\| \leq \epsilon_K$ and $\|\beta_k^{(j+1)} - \beta_k^{(j)}\| \leq \epsilon_\beta$ for all k **then**
    `policystable` $\leftarrow$ `true`
  end if
  $j \leftarrow j + 1$
end while

3.4.1 Duality

For a fixed policy (K_k, β_k), the quantities propagated satisfy two coupled Lyapunov-type recursions driven by the same closed-loop matrix $A + BK_kC$. The state covariance evolves forward in time as

$$\Theta_{k+1} = (A + BK_kC)\Theta_k(A + BK_kC)^\intercal + W + BK_kVK_k^\intercal B^\intercal \tag{3.37}$$

with G_k fixed and with $L_k = \Theta_k C^\intercal (C\Theta_k C^\intercal + V)^{-1}$. The quadratic cost matrix evolves backwards as

$$P_k = (A + BK_kC)^\intercal P_{k+1}(A + BK_kC) + Q + C^\intercal K_k^\intercal R K_k C \tag{3.38}$$

with $G_k = -(R + B^\intercal P_{k+1}B)^{-1}B^\intercal P_{k+1}A$ and with L_k fixed. These equations form a dual pair:

$$(A, B, C, Q, R) \longleftrightarrow (A^\intercal, C^\intercal, B^\intercal, W, V).$$

Together, equations (3.37) and (3.38) determine the quadratic form of the observation-action value driving the policy improvement.

Remark 3.1. *Note that if we assume that $\mathcal{E}[x_0] = 0$, then $\bar{x}_k = 0$ for every k and that for the initial policy $\beta_k = 0$ for every k. Therefore $S_k = 0$ for every k, and we do not need the iteration for S_k to compute the policy and* (3.38) *suffices.*

In the infinite-horizon setting these forward and backward recursions are replaced by stationary solutions of the same two Lyapunov-type equations, yielding coupled fixed-point conditions for the static output feedback gain that form the basis of Section 3.5.

3.5 Infinite-horizon static output feedback LQG

We now specialize to the infinite-horizon average-cost version of the static output feedback LQG problem. The problem is formulated in Section 3.5.1. Motivated by the finite-horizon Riccati-like recursions of Section 3.4, we derive stationary fixed-point equations and propose an iterative scheme in Section 3.5.2. The proof of monotonic cost convergence of the algorithm is given in Section 3.5.3, under the assumption that two dual Riccati-like equations convergence. For systems with $n_x = 1$, the convergence of these Riccati-like equations can be fully characterized, which is shown in Section 3.5.4.

3.5.1 Average cost and stationary policies

We consider the infinite-horizon average-cost version of the static output feedback LQG problem. The control policies considered are stationary versions of the control

law defined in (3.19):

$$u_k = Ky_k + \beta.$$

3

The performance index is defined as:

$$\begin{aligned} J_{\text{avg}} = \lim_{N\to\infty} \frac{1}{N+1} \big(& x_0^\intercal P_0 x_0 + 2S_0 x_0 \\ &+ \sum_{i=0}^{N-1} \left(\text{Tr}(P_{i+1}(W + BKV(BK)^\intercal)) + 2S_{i+1}B\beta \right. \\ &\left. +\beta^\intercal(R + B^\intercal P_{i+1} B)\beta + \text{Tr}(RKVK^\intercal)\right). \end{aligned}$$

Since now we consider an average cost problem, the expected initial condition is zero, in which case in the finite-horizon case $\beta_k = 0$ (see Remark 3.1), and therefore here we consider simply

$$u_k = Ky_k. \tag{3.39}$$

Let us define the set of stabilizing control gains K as:

$$\mathbb{K} := \left\{K \in \mathbb{R}^{n_u \times n_y} : \rho(A + BKC) < 1\right\} \tag{3.40}$$

which is assumed to be non-empty as stated next:

Assumption 3.1. The set $\mathbb{K}$ is non-empty.

Furthermore, we have the following standard assumptions for LQG problems:

Assumption 3.2. Pairs (A, B) and $(A, W^{1/2})$ are controllable, pairs (A, C) and $(A, Q^{1/2})$ are observable, $Q \succeq 0$, $R \succ 0$, $W \succeq 0$, $V \succ 0$.

For $K \in \mathbb{K}$, there is a unique covariance matrix $\Theta(K) \succeq 0$ satisfying the discrete-time Lyapunov equation

$$\Theta = (A + BKC)\Theta(A + BKC)^\intercal + W + BKVK^\intercal B^\intercal. \tag{3.41}$$

Moreover, the stationary quadratic cost-to-go associated with a stabilizing policy is characterized by matrix $P(K) \succeq 0$ solving the dual Lyapunov equation

$$P = (A + BKC)^\intercal P(A + BKC) + Q + C^\intercal K^\intercal RKC. \tag{3.42}$$

For any stabilizing K, the corresponding average cost exists and can be expressed as:

$$\begin{aligned} J_{\text{avg}}(K) &= \text{Tr}(P(K)(W + BKVK^\intercal B^\intercal)) + \text{Tr}(RKVK^\intercal) \\ &= \text{Tr}(\Theta(K)(Q + C^T K^T RKC)) + \text{Tr}(RKVK^\intercal). \end{aligned} \tag{3.43}$$

The infinite-horizon static output feedback problem is therefore to find $K \in \mathbb{K}$ that minimizes $J_{\text{avg}}(K)$ subject to (3.41) and (3.42). This is however a hard, non-convex problem, which cannot be solved analytically.

3.5.2 Infinite-horizon policy iteration algorithm

The finite-horizon algorithm of Section 3.4 alternates between forward sweeps updating the state covariance and backward sweeps updating the cost-to-go matrices, while updating the policy at each stage. We describe an extension of Algorithm 3 to the infinite-horizon setting, by replacing the finite-horizon recursions with their infinite-horizon limits.

Under the assumption that at iteration j $\Theta_k^{(j)} \to \Theta^{(j+1)}$ for some $\Theta^{(j+1)}$ as $k \to \infty$, and $P_k^{(j)} \to P^{(j+1)}$ for some $P^{(j+1)}$ as $k \to -\infty$, Algorithm 4 is a natural extension of Algorithm 3 to the infinite-horizon setting. The algorithm keeps on alternating between the forward and backward sweeps, while the sweeps are executed up until convergence of the corresponding Lyapunov-type equations (3.37) and (3.38).

In the next section, we state that the average cost monotonically decreases to a local optimum, provided that the Lyapunov-type equations (3.37) and (3.38) converge. In Section 3.5.4, convergence of Θ_k and P_k to respectively $\Theta^{(j+1)}$ and $P^{(j+1)}$ is shown when $n_x = 1$. For brevity, we only detail the backward sweep and the associated P-iterations, since the forward sweep and the convergence of Θ follow analogously by duality.

3.5.3 Monotonic cost convergence

We now establish the monotonic improvement property of the proposed infinite-horizon policy iteration scheme. Using the finite-horizon arguments of Section 3.3 together with the Riccati-like iterations of Section 3.5.2, we show that, under suitable convergence assumptions on the forward and backward sweeps, the average cost is non-increasing at every iteration.

Theorem 3.2. *Let $K^{(0)} \in \mathbb{K}$ be stabilizing, such that Θ is fixed and satisfies* (3.41) *where $L = \Theta C^\intercal (C\Theta C^\intercal + V)^{-1}$ Suppose that at iteration j the backward sweep generates a sequence $\{K_k^{(j+1)}\}_{k\geq 0}$ such that the associated Riccati-like iteration*

$$P_k = \mathcal{T}_P(P_{k+1}, K_k^{(j+1)}) \tag{3.44}$$

converges to a stationary limit $P^{(j+1)}$ and thus $K_k^{(j+1)} \to K^{(j+1)} \in \mathbb{K}$. Then the average cost satisfies:

$$J_{avg}(K^{(j+1)}) \leq J_{avg}(K^{(j)}). \tag{3.45}$$

▲

The same conclusion holds for the forward sweep by duality. The proof can be found in Section 3.8.2.

Algorithm 4 Policy iteration - infinite-horizon

Require: $K^{(0)}$

Compute $\Theta^{(0)}$ and $P^{(0)}$ using (3.41)-(3.42)
$j \leftarrow 0$
`policystable` $\leftarrow$ `false`
while `policystable` = `false` **do**
    Forward sweep
    $k \leftarrow 0$
    $\Theta_0 \leftarrow \Theta^{(j)}$
    while $\|\Theta_{k+1} - \Theta_k\| > \epsilon_\Theta$ **do**
        $K_k \leftarrow \mathcal{F}_K(P^{(j)}, \Theta_k)$
        $\Theta_{k+1} \leftarrow \mathcal{T}_\Theta(\Theta_k, K_k)$
        $k \leftarrow k + 1$
    end while
    $K^{(j+1)} \leftarrow K_k$
    $\Theta^{(j+1)} \leftarrow \Theta_{k+1}$
    $P^{(j+1)} \leftarrow P^{(j)}$
    $j \leftarrow j + 1$
    Backward sweep
    $k \leftarrow 0$
    $P_0 \leftarrow P^{(j)}$
    while $\|P_{k-1} - P_k\| > \epsilon_P$ **do**
        $K_k \leftarrow \mathcal{F}_K(P_k, \Theta^{(j)})$
        $P_{k-1} \leftarrow \mathcal{T}_P(P_k, K_k)$
        $k \leftarrow k - 1$
    end while
    $K^{(j+1)} \leftarrow K_k$
    $\Theta^{(j+1)} \leftarrow \Theta^{(j)}$
    $P^{(j+1)} \leftarrow P_{k-1}$
    $j \leftarrow j + 1$
    if $\|K^{(j)} - K^{(j-2)}\| \leq \epsilon_K$ **then**
        `policystable` $\leftarrow$ `true`
    end if
end while

3.5.4 Convergence of Riccati-like iterations

It remains to establish convergence of the Riccati-like iteration $P_k = \mathcal{T}_P(P_{k+1}, K_k)$. By eliminating K_k, this iteration can be written as a Riccati-like equation:

$$\begin{aligned} P_k &= \mathcal{F}_P(P_{k+1}) \\ &:= A^\intercal(P_{k+1} - P_{k+1}B(R + B^\intercal P_{k+1}B)^{-1}B^\intercal P_{k+1})A + Q \\ &\quad + (I - LC)^\intercal A^\intercal P_{k+1}B(R + B^\intercal P_{k+1}B)^{-1}B^\intercal P_{k+1}A(I - LC) \end{aligned} \tag{3.46}$$

where $L = \Theta C^\intercal(C\Theta C^\intercal + V)^{-1}$. Comparing (3.46) with the standard discrete-time Riccati equation reveals the presence of an additional term. As a result, classical convergence results for the Riccati recursion do not apply directly.

Remark 3.2. *In case of state feedback ($C = I$, $V = 0$), it can be shown that $L = I$, and* (3.46) *reduces to standard discrete-time Riccati equation.*

Analogously to the standard Riccati recursion, our goal is to show that for every positive semidefinite symmetric initial condition P_0, the backward iteration converges to a unique positive fixed point P^*. In this section, we establish this property when $n_x = 1$.

Lemma 3.1. *Consider a system with $n_x = 1$ satisfying Assumption 3.2 and fix a gain $K^{(j)} \in \mathbb{K}$, yielding a stationary covariance $\Theta^{(j)} > 0$ and Kalman gain $L = \Theta^{(j)}C/(C^2\Theta^{(j)} + V)$. Then the Riccati-like equation*

$$P = \mathcal{F}_P(P) \tag{3.47}$$

admits a unique positive solution $P > 0$. ▲

Theorem 3.3. *Consider a system with $n_x = 1$ satisfying Assumption 3.2 and fix a gain $K^{(j)} \in \mathbb{K}$, yielding a stationary covariance $\Theta^{(j)} > 0$ and Kalman gain $L = \Theta^{(j)}C/(C^2\Theta^{(j)} + V)$. Let $\{P_k\}$ be generated by the Riccati-like backward iteration*

$$P_k = \mathcal{F}_P(P_{k+1}) \tag{3.48}$$

with arbitrary initial condition $P_0 \geq 0$. Then P_k converges as $k \to -\infty$ to the unique positive fixed point P^* characterized in Lemma 3.1. ▲

The proof of Lemma 3.1 and Theorem 3.3 can be found in Sections 3.8.3 and 3.8.4, respectively.

Since the backward sweep converges to the unique positive definite fixed point P^*, the solution satisfies (3.38), and the corresponding policy update produces a stabilizing gain $K^{(j+1)} \in \mathbb{K}$. Consequently, the next forward sweep is well-defined and the policy iteration scheme may proceed.

3.6 Numerical examples

This section presents numerical experiments that illustrate the theoretical results developed in the preceding sections. Section 3.6.1 focuses on finite-horizon problems, and compares the performance and convergence with the method developed in [32]. Section 3.6.2 addresses infinite-horizon scenarios, by applying Algorithm 4 to an LQG system with $n_x = 1$.

3.6.1 Finite-horizon

Consider the following linear system (adopted from [32]):

$$x_{k+1} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 2 & -1 & -1 \end{bmatrix} x_k + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u_k + w_k \tag{3.49a}$$

$$y_k = \begin{bmatrix} 1 & -0.2 & 0 \end{bmatrix} x_k + v_k \tag{3.49b}$$

with

$$\Theta_0 = \begin{bmatrix} 4 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 4 \end{bmatrix}, \quad W = \begin{bmatrix} 10 & 1 & 2 \\ 1 & 5 & 1 \\ 2 & 1 & 10 \end{bmatrix},$$

$V = 10$, $Q = \mathrm{diag}([10, 10, 10])$, $R = 1$, and $N = 50$. To illustrate the convergence, $\mathbb{E}[x_0]$ and the initial policy parameters are varied in the simulation study. We compare Algorithm 3 with the proposed method in [32].

The performance of the proposed method is illustrated in Figure 3.1, where J_π^* is the lowest cost found in simulation. For $\mathbb{E}[x_0] = [0, 0, 0]^\intercal$, the method converges monotonically to numerical precision in around 10 iterations, independent of the initial policy parameters. For the method of [32], the policy converges in significantly more iterations if the initial policy parameters are initialized as $K_k^{(0)} = -2$. One reason for this is that our method updates the policy parameters twice for the same computation effort for Θ and P. For the method in [32], simulation results showed divergence for $K_k^{(0)} = 0$ indicating its sensitivity to initial conditions. Furthermore, the method in [32] is unable to deal with expected initial states, as shown in Figure 3.1. In this example, it converged (not monotonically), to a solution with a significantly higher cost.

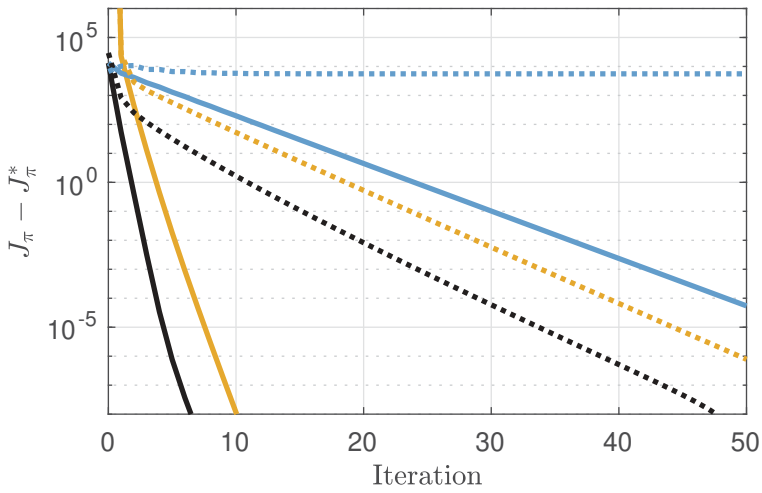


Figure 3.1. Comparison of Algorithm 3 with $K_k^{(0)} = -2$, $\beta_k^{(0)} = 0$ (—), and $K_k^{(0)} = 0$, $\beta_k^{(0)} = 0$ (—), with the method in [32] with $K_k^{(0)} = -2$ (—). Solid lines have expected initial state $\mathbb{E}[x_0] = [0, 0, 0]^\intercal$, whereas $\mathbb{E}[x_0] = [10, 10, 10]^\intercal$ are indicated with dotted lines.

3.6.2 Infinite-horizon

Consider an LQG system with $A = 2$, $B = 1$, $C = 1$, $W = 0.5$, $V = 1$, $Q = 5$, and $R = 2$. Furthermore, we set $\epsilon_\Theta = \epsilon_P = 10^{-12}$ and $\epsilon_K = 10^{-8}$. The set of stabilizing control gains for this system is $\mathbb{K} := (-3, -1)$. We applied Algorithm 4 for 10 random $K^{(0)} \in \mathbb{K}$. Figure 3.2 shows that in this example for all initial stabilizing controllers, the method converges to the same optimal solution ($K^* = -1.3916$). It is not possible to choose $K^{(0)} \notin \mathbb{K}$, because then the Lyapunov equation cannot be solved and the policy iteration scheme cannot be initialized.

To confirm Theorem 3.2, the monotonic cost convergence is illustrated in Figure 3.3. For all $K^{(0)} \in \mathbb{K}$, the method converges monotonically to numerical precision in just a few iterations. By applying Algorithm 4 to n-dimensional LQG systems, simulations showed similar monotonic convergence to local optima.

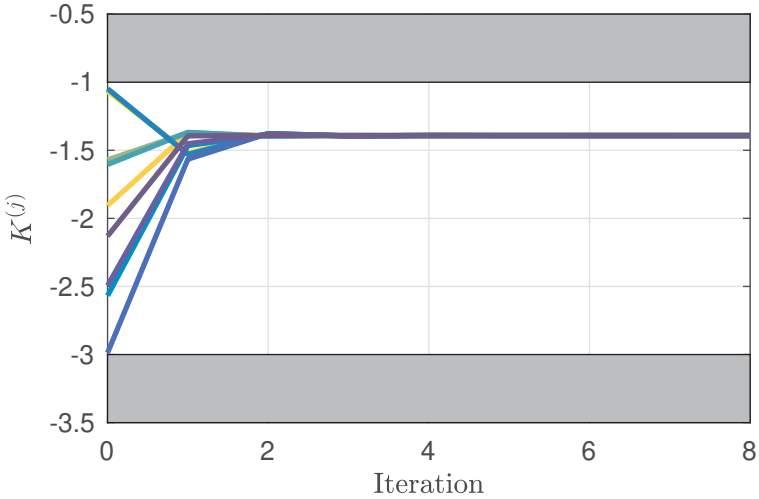


Figure 3.2. Convergence of the controller gains for 10 random $K^{(0)} \in \mathbb{K}$.

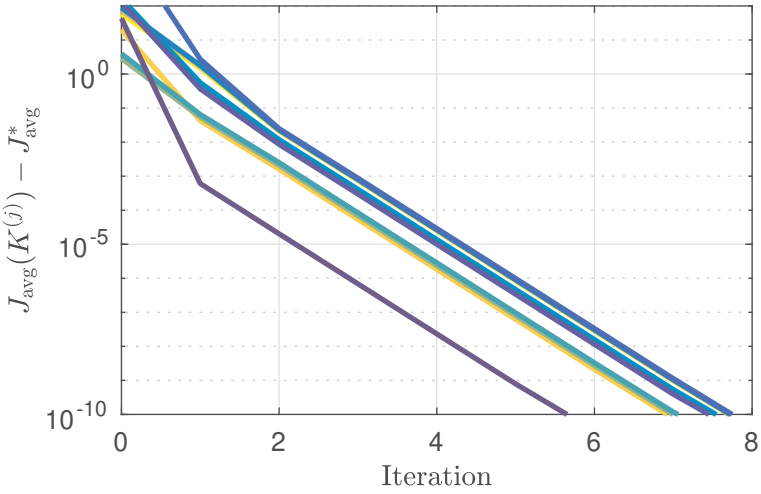


Figure 3.3. Monotonic cost convergence for 10 random $K^{(0)} \in \mathbb{K}$.

3.7 Conclusions and future work

This chapter developed a memoryless policy iteration framework for finite-horizon continuous POMDPs and specialized it to static output feedback LQG control. By explicitly working with observation-action value functions, we obtained a deterministic policy iteration scheme with monotonic cost improvement guarantees and derived closed-form update rules in the LQG case. The resulting algorithm alternates forward propagation of state statistics with backward Riccati-like recursions for the quadratic cost-to-go, revealing a dual Lyapunov structure that underpins both the finite and infinite-horizon developments.

For infinite-horizon problems, we proposed a stationary extension of the algorithm and established monotonic improvement of the average cost under suitable convergence assumptions. When $n_x = 1$, we proved convergence of the associated Riccati-like iterations to a unique positive fixed point. Numerical experiments for both finite and infinite-horizon settings demonstrated monotonic convergence to locally optimal static output feedback policies and illustrated the practical effectiveness of the proposed approach.

There are several directions for future research. One important direction is to extend to the infinite-horizon the convergence analysis of the Riccati-like equations to general n-dimensional systems. Beyond linear systems, it would also be of interest to investigate whether similar forward–backward structures and analytic policy updates can be developed for general nonlinear systems using approximate methods.

3.8 Appendix

3.8.1 Proof of Theorem 3.1

Note that $J_{\pi^{(j)}}$ can be written as:

$$\begin{aligned} J_{\pi^{(j)}} &= \mathbb{E}\left[\sum_{k=0}^{N-1} g_k(x_k, y_k, \mu_k^{(j)}(y_k)) + g_N(x_N, y_N)\right] \\ &= \int_{\mathcal{Y}_k} \bar{Q}_k^{\pi^{(j)}}(y, \mu_k^j(y))\gamma_k^{(j)}(dy) \\ &= \int_{\mathcal{Y}_k}\int_{\mathcal{X}_k} Q_k^{\pi^{(j)}}(\xi, \mu_k^{(j)}(y))q_y(\xi, y)p_k^{(j)}(d\xi)dy. \end{aligned} \tag{3.50}$$

If the policy is only improved at $k = 0$, we have

$$\begin{aligned} J_{\pi^{(j+1)}} &= \int_{\mathcal{Y}_0} \bar{Q}_0^{\pi^{(j+1)}}(y, \mu_0^{(j+1)}(y))\gamma_0^{(j+1)}(dy) \\ &\leq \int_{\mathcal{Y}_0} \bar{Q}_0^{\pi^{(j)}}(y, \mu_0^{(j)}(y))\gamma_0^{(j)}(dy) \\ &= J_{\pi^{(j)}} \end{aligned} \tag{3.51}$$

since $\gamma_0^{(j+1)} = \gamma_0^{(j)}$ is independent of the policy being used and $\bar{Q}_0^{\pi^{(j+1)}}(y, u) = \bar{Q}_0^{\pi^{(j)}}(y, u)$. If the policy is improved at an arbitrary k, we unroll (3.10) up until k, such that we obtain:

$$\begin{aligned} J_{\pi^{(j+1)}} &= \sum_{i=0}^{k-1} \int_{\mathcal{X}_i} \int_{\mathcal{Y}_i} g_i(\xi, y, \mu_i^{(j+1)}(y)) q_y(\xi, y) dy p_i^{(j+1)}(d\xi) \\ &\quad + \int_{\mathcal{X}_k} \int_{\mathcal{Y}_k} Q_k^{\pi^{(j+1)}}(\xi, \mu_k^{(j+1)}(y)) q_y(\xi, y) dy p_k^{(j+1)}(d\xi) \\ &= \sum_{i=0}^{k-1} \int_{\mathcal{X}_i} \int_{\mathcal{Y}_i} g_i(\xi, y, \mu_i^{(j+1)}(y)) q_y(\xi, y) dy p_i^{(j+1)}(d\xi) \\ &\quad + \int_{\mathcal{Y}_k} \bar{Q}_k^{\pi^{(j+1)}}(y, \mu_k^{(j+1)}(y))\gamma_k^{(j+1)}(dy) \\ &= \sum_{i=0}^{k-1} \int_{\mathcal{X}_i} \int_{\mathcal{Y}_i} g_i(\xi, y, \mu_i^{(j)}(y)) q_y(\xi, y) dy p_i^{(j)}(d\xi) \\ &\quad + \int_{\mathcal{Y}_k} \bar{Q}_k^{\pi^{(j+1)}}(y, \mu_k^{(j+1)}(y))\gamma_k^{(j+1)}(dy) \\ &\leq \sum_{i=0}^{k-1} \int_{\mathcal{X}_i} \int_{\mathcal{Y}_i} g_i(\xi, y, \mu_i^{(j)}(y)) q_y(\xi, y) dy p_i^{(j)}(d\xi) \\ &\quad + \int_{\mathcal{Y}_k} \bar{Q}_k^{\pi^{(j)}}(y, \mu_k^{j}(y))\gamma_k^{(j)}(dy) \\ &= J_{\pi^{(j)}}. \end{aligned} \tag{3.52}$$

This follows from that if two policies coincide at all stages $i < k$, they induce the same distributions p_i and γ_i for all $i \leq k$. Thus, because the policies coincide up to stage $k-1$, the induced distributions up to stage k are identical, and the stage-k control $\mu_k^{(j+1)}$ minimizes $\bar{Q}_k^{\pi^{(j)}}$ pointwise, we obtain $J_{\pi^{(j+1)}} \leq J_{\pi^{(j)}}$. ■

3.8.2 Proof of Theorem 3.2

Let us start with a finite horizon $N \in \mathbb{N}$. Consider the finite horizon cost

$$J_N(K) := \mathbb{E}\left[\sum_{k=0}^{N-1} x_k^\intercal Q x_k + u_k^\intercal R u_k\right] \tag{3.53}$$

where $u_k = K y_k$. Let $K_k^{(j)} = K^{(j)}$ for all k, and let the backward sweep produce the sequence $\{K_k^{(j+1)}\}_{k=0}^{N-1}$. By the finite horizon policy iteration result of Section 3.4 each single-stage improvement decreases the cost, hence

$$J_N(\{K_k^{(j+1)}\}_{k=0}^{N-1}) \leq J_N(K^{(j)}). \tag{3.54}$$

By assumption, for large N,

$$K_k^{(j+1)} \to K^{(j+1)} \in \mathbb{K}, \tag{3.55}$$

and the corresponding Riccati-like recursion converges to $P^{(j+1)}$. Since $K^{(j+1)} \in \mathbb{K}$, the average cost of the new policy is well-defined, hence

$$\lim_{N\to\infty} \frac{1}{N} J_N(\{K_k^{(j+1)}\}_{k=0}^{N-1}) = J_{\text{avg}}(K^{(j+1)}) \tag{3.56}$$

and similarly

$$\lim_{N\to\infty} \frac{1}{N} J_N(K^{(j)}) = J_{\text{avg}}(K^{(j)}). \tag{3.57}$$

By using inequality (3.54) divided by N and by letting $N \to \infty$, combined with the two limits above, leads to

$$J_{\text{avg}}(K^{(j+1)}) \leq J_{\text{avg}}(K^{(j)}). \tag{3.58}$$

■

3.8.3 Proof of Lemma 3.1

We first denote a property that will be repeatedly used in the analysis. Since $K^{(j)} \in \mathbb{K}$ is stabilizing, the corresponding stationary covariance satisfies $\Theta^{(j)} > 0$, and hence

$$0 < 1 - LC = \frac{C^2\Theta^{(j)}}{C^2\Theta^{(j)} + V} < 1. \tag{3.59}$$

When $n_x = 1$, the fixed-point equation $P = \mathcal{F}_P(P)$ reduces to the quadratic equation

$$(A^2(1-LC)^2 - 1)B^2P^2 + (R(A^2-1) + QB^2)P + QR = 0. \tag{3.60}$$

Assumption 3.2 ensures that the constant term is strictly positive. Since $0 < 1 - LC < 1$, the discriminant is given by

$$\begin{aligned} D &= (R(A^2-1) + QB^2)^2 - 4(A^2(1-LC)^2 - 1)B^2QR \\ &= R^2(A^2-1)^2 + Q^2B^4 + 2(A^2 - 2A^2(1-LC)^2 + 1)B^2QR \\ &> R^2(A^2-1)^2 + Q^2B^4 + 2(-A^2+1)B^2QR \\ &= (R(A^2-1) - QB^2)^2 > 0 \end{aligned} \tag{3.61}$$

which guarantees the existence of two real roots. To establish uniqueness of a positive solution, it suffices to show that the quadratic coefficient is negative, i.e., $A^2(1-LC)^2 - 1 < 0$. Using $A_{\text{cl}} := A + BK^{(j)}C$, one obtains

$$\begin{aligned} 1 - LC &= 1 - \frac{\Theta C^2}{C^2\Theta + V} \\ &= \frac{V(1 - A_{\text{cl}}^2)}{C^2(W + (BK^{(j)})^2 V) + V(1 - A_{\text{cl}}^2)}. \end{aligned} \tag{3.62}$$

Using $W \succeq 0$, the following inequality can be found:

$$\begin{aligned} A^2(1-LC)^2 - 1 &\leq A^2 \left(\frac{1 - A_{\text{cl}}^2}{(A_{\text{cl}} - A)^2 + 1 - A_{\text{cl}}^2} \right)^2 - 1 \\ &= \frac{\left(A(1 - A_{\text{cl}}^2)\right)^2 - \left((A_{\text{cl}} - A)^2 + 1 - A_{\text{cl}}^2\right)^2}{\left((A_{\text{cl}} - A)^2 + 1 - A_{\text{cl}}^2\right)^2}. \end{aligned} \tag{3.63}$$

Since $K^{(j)}$ is stabilizing $|A_{\text{cl}}| < 1$, and it can be verified that, for all A, the numerator is strictly negative, we have

$$A^2(1-LC)^2 - 1 < 0. \tag{3.64}$$

Therefore the quadratic equation (3.60) has exactly one positive root, which is the unique positive fixed point of $\mathcal{F}_P$. ■

3.8.4 Proof of Theorem 3.3

Recall that the Riccati-like map is

$$\mathcal{F}_P(P) = A^2P + Q + \frac{(A^2(1-LC)^2 - A^2)B^2P^2}{B^2P + R} \tag{3.65}$$

and its derivative w.r.t. P is given by

$$\mathcal{F}'_P(P) = A^2 + A^2\left((1-LC)^2 - 1\right)\frac{B^4P^2 + 2B^2RP}{B^4P^2 + 2B^2PR + R^2}. \tag{3.66}$$

Since $0 < 1 - LC < 1$, the derivative for $P \geq 0$ satisfies $\mathcal{F}'_P(P) > 0$, so that $\mathcal{F}_P$ is monotone increasing on $\mathbb{R}_{\geq 0}$.

By Lemma 3.1, $\mathcal{F}_P$ admits a unique positive fixed point $P^* > 0$, which satisfies

$$(1 - A^2)P^* - Q = \frac{A^2((1 - LC)^2 - 1)B^2(P^*)^2}{B^2P^* + R}. \tag{3.67}$$

Evaluating the derivative at P^* gives:

$$\begin{aligned} \mathcal{F}'_P(P^*) &= A^2 + \frac{A^2\left((1 - LC)^2 - 1\right)B^2(P^*)^2}{B^2P^* + R} \cdot \frac{B^2P^* + 2R}{P(B^2P^* + R)} \\ &= A^2 + ((1 - A^2)P^* - Q) \cdot \frac{B^2P^* + 2R}{P(B^2P^* + R)}. \end{aligned} \tag{3.68}$$

After rewriting and simplifications, one can verify the expression

$$1 - \mathcal{F}'_P(P^*) = \frac{Q(R + B^2P^*) + R(Q - (1 - A^2)P^*))}{P^*(B^2P^* + R)}. \tag{3.69}$$

The denominator is strictly positive. Moreover, (3.67) implies

$$Q - (1 - A^2)P^* = -\frac{A^2((1 - LC)^2 - 1)B^2(P^*)^2}{B^2P^* + R} > 0 \tag{3.70}$$

since $(1 - LC)^2 < 1$. Consequently

$$0 < \mathcal{F}'_P(P^*) < 1, \tag{3.71}$$

thus $\mathcal{F}_P$ is locally contractive at its unique fixed point. Finally, since $\mathcal{F}_P$ is monotone on $\mathbb{R}_{\geq 0}$ and admits a single fixed point in that set, standard fixed-point arguments imply that the backward iteration converges to P^* for any initial $P_0 \geq 0$.

■

II

Data-Efficient Policy Evaluation

Chapter 4

Maximum A Posteriori Least-Squares Temporal Difference

Abstract - Least-Squares Temporal Difference (LSTD) is a model-free method to estimate the cost/value function of a Markov Decision Process (MDP). Interestingly, it is equivalent to estimating the state transition probabilities of the MDP through Maximum Likelihood (ML) and computing the cost/value function with (model-based) policy evaluation. However, LSTD does not incorporate prior knowledge of the state transition probabilities of the MDP. This chapter proposes a data-based method, coined MAP-LSTD, that incorporates such prior knowledge without explicitly estimating the model. This method is analogous to LSTD in the sense that it is equivalent to estimating the state transition probabilities (model) through Maximum A Posteriori (MAP) instead of ML and computing the cost function with policy evaluation. Moreover, as LSTD, it can be implemented recursively. The key to our method is to model the unknown state distributions with Dirichlet distributions and encapsulate prior knowledge in the parameters of these distributions. An important concomitant is that the prior can be obtained from previous data samples and/or simulated samples, leveraging a model/simulator. Through an example, we illustrate how our method can speed up the convergence of the estimated cost function and reduce the estimation error significantly by exploiting prior knowledge derived from a model/simulator.

This chapter is based on: R.A.C. van Zuijlen and D.J. Guerreiro Tomé Antunes, "Maximum A Posteriori Least-Squares Temporal Difference", *American Control Conference*, p. 3533–3538, 2025.

4.1 Introduction

The cost function of a Markov Decision Process (MDP) with a given policy can be computed exactly through policy evaluation [11], when the model (state transition probabilities) is known. When the model is unknown, it can be estimated from data samples of system trajectories. If the computation of the exact cost function is infeasible due to the (possibly infinite) number of states, an approximation of the cost function can be pursued. Computing the cost function of an MDP is important in many contexts, for instance as a step towards finding the optimal policy through (model-based or simulation-based) policy iteration.

There are several methods in the literature to learn the cost function based on data samples without explicitly estimating first a model, e.g., (gradient) temporal difference learning, Monte Carlo methods, and least-squares methods, such as Least-Squares Temporal Difference learning (LSTD) [16], or LSTD with eligibility traces [15]. For overviews of policy evaluation (cost function estimation) methods see e.g. [11, 24, 115]. All these methods are model-free and have the advantage that they estimate the cost function directly from data samples. Another advantage over estimating first a model and then using (model-based) policy evaluation are simplicity and often requiring fewer parameters to be estimated (e.g., in the tabular case with state dimension n, n parameters should be estimated rather than $n \times n$ transition probabilities). LSTD estimates the cost function parameters in a least-squared fashion, based on all available data samples. It handles both an exact tabular representation of the cost function, where each state has a separate cost, and linear combinations of features to represent a continuous cost function. Interestingly, the cost function estimate obtained by LSTD can equivalently be computed by: (i) estimating first state transition probabilities from data based on Maximum Likelihood (ML) and then (ii) using (model-based) policy evaluation. This holds both for the tabular case [15] and the linear representation case, where the model is interpreted through a projection in the feature space [84].

Since obtaining data in real-world applications is often costly, the number of available data samples is limited. This hampers the accuracy of the cost function estimation. One important challenge is therefore to have acceptable performance/accuracy with limited data samples [31]. Conceivably, if prior, even if limited, system knowledge is available, it can be leveraged to mitigate the scarcity of data. This prior knowledge can have different origins, e.g., user experience, available simulators/digital twins, physics-based or logical-based assumptions, etc. Extensive work is available in the literature on adding prior information to reinforcement learning in a Bayesian framework, see, e.g., [38]. However, the main focus so far has been on using priors for the exploration/exploitation trade-off and planning algorithms, and not specifically on priors on the state transition probabilities for cost function estimation [30, 41, 97].

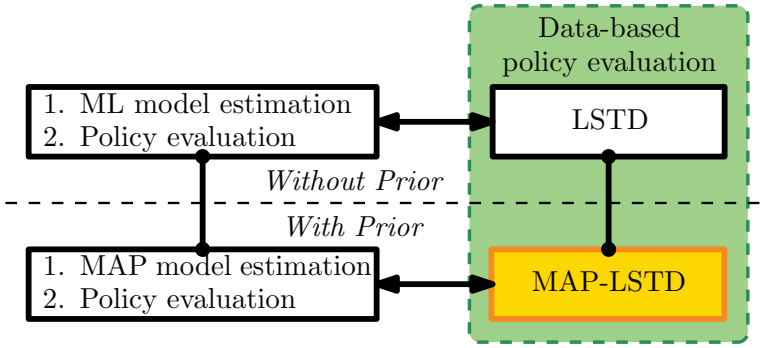


Figure 4.1. Placement of MAP-LSTD in the framework of existing methods.

We propose a new method for data-based policy evaluation that allows for incorporating prior knowledge without explicitly estimating a model. Both tabular and linear architectures are considered. The method has two main key ingredients. The first is to model the unknown state transitions with Dirichlet distributions and encapsulate prior knowledge in the parameters of these distributions. The second is to formulate an optimization problem similar to the one leading to LSTD, where now a penalty regularization term is added that takes into account prior information. This is a usual approach to include prior information in general parameter estimation problems, see e.g. [24, 39].

Our main result establishes that the proposed method is equivalent to learning the state transition probabilities based on Maximum a Posteriori (MAP) estimation (rather than an ML estimation as in LSTD, see e.g. [36]), and using model-based policy evaluation subsequently. In this sense, the proposed method is a natural extension of LSTD and we refer to it as MAP-LSTD. Including prior knowledge (e.g. topological information on the state transition probabilities) to learn state transition probabilities according to MAP prevents overfitting when a limited amount of data samples is available [88]. Figure 4.1 illustrates how to place MAP-LSTD with respect to the other mentioned methods.

Similar to LSTD, we show that MAP-LSTD can also be implemented recursively. Moreover, the method allows to select a non-informative prior, in which case it boils down to LSTD, and a partially informative prior such that only prior information on some (more relevant) state transitions is incorporated. This makes it applicable to large-scale applications where incorporating prior knowledge of all state transitions is unfeasible. Relevant state transitions can be obtained from previous data samples, which is an important concomitant of using Dirichlet distributions. Moreover, whenever a model/simulator is available, these samples can be generated by the model/simulator. Therefore the method provides a way to

leverage a model/simulator as we will illustrate through an example in Section 4.5 (see Figure 4.3).

This chapter is structured as follows. In Section 4.2, the LSTD method is reviewed and the problem considered in this chapter is stated. A suitable prior is selected and justified in Section 4.3. Section 4.4 provides the proposed MAP-LSTD method, the main result, and the most important characteristics. Section 4.5 discusses a numerical example where MAP-LSTD and LSTD are compared in terms of convergence speed and variance in the estimation error. Conclusions are drawn and directions for future research are given in Section 4.6.

Notation: Diagonal matrices are denoted by $\mathrm{diag}([a_1, \ldots, a_n])$, where a_i are the diagonal entries. $\|\cdot\|^2_\Lambda$ denotes the weighted squared norm: $\|x\|^2_\Lambda = x^\top \Lambda x$.

4.2 Background and problem statement

We restrict ourselves to infinite-horizon discounted cost problems for Markov decision problems (MDPs). However, the ideas presented here can be extended to other problems (e.g. shortest path problems). Let $x_k \in X$ denote the state at time $k \in \mathbb{N} \cup \{\infty\}$ and $u_k \in U$ be the control input, where X and U are the state and control spaces, respectively. The cost of a policy $\pi : X \to U$ for each state x_k is given by

$$\begin{aligned} V_\pi(x_k) &= \lim_{H\to\infty} \mathbb{E}\left\{ \sum_{i=k}^{H-1} \alpha^{i-k} g(x_i, x'_i) \right\} \\ &= \mathbb{E}\left\{ g(x_k, x'_k) \right\} + \alpha V_\pi(x'_k), \end{aligned} \tag{4.1}$$

where $x'_k := x_{k+1}$ is the next state, and $\alpha \in (0, 1]$ is the discount factor. The state evolves according to $x'_k = f(x_k, u_k, w_k)$, $f : X \times U \times W \to X$. Disturbances $w_k \in W$ are assumed to be mutually independent. X, U and W are finite sets. The total number of states is denoted by n, $X = \{1, 2, \ldots, n\}$. In (4.1), $g(x_k, x'_k)$ are the state transition costs $g : X \times X \to \mathbb{R}$. The goal of policy evaluation is to find cost (4.1) for every initial state x_0 of the MDP under (fixed) policy π. The cost function is estimated based on data set $\mathcal{X} := \{(x_i, x'_i, g(x_i, x'_i)) | i = 1, ..., n_s\}$ obtained with policy π, where n_s are the number of available data samples. Since the policy is fixed, u_k is not relevant for the cost function estimation.

4.2.1 Least-squares temporal difference learning

The following squared error provides a usual criteria for obtaining an approximation of the exact cost of a given policy

$$\min_r \sum_{x\in X} \Big(V(x) - \tilde{V}(x, r) \Big)^2 \tag{4.2}$$

where $V(x)$ is the exact cost function as defined in (4.1), and $\tilde{V}(x,r)$ is the approximation based on parameters r. In LSTD, the cost function is approximated with a linear architecture, defined as

$$\tilde{V}(x,r) = \phi(x)^\top r, \tag{4.3}$$

where $\phi(x)$ is the feature vector and r is the parameter vector. A cost function will often be identified with an n-dimension vector, e.g. $J = \begin{bmatrix} V(1,r) & \dots & V(n,r) \end{bmatrix}^\intercal$ for given r. The feasible set of vectors representing linear cost function approximations (4.3) is denoted by $\mathcal{J} := \{\Phi r | r \in \mathbb{R}^p\}$, where Φ is the feature matrix. Each row of Φ contains feature vector $\phi(x)^\top$. Feature matrix Φ is assumed to have rank p. The exact tabular case can be captured by picking $p = n$, $\phi_i(x) = 1$ if $x = i$, $\phi_i(x) = 0$ otherwise, in which case each parameter r_i corresponds to the cost of state i.

Since we cannot solve (4.2) exactly, the mean-squared projected Bellman error (MSPBE) is considered as alternative:

$$\tilde{J} = \arg\min_{J \in \mathcal{J}} \|J - \Pi T J\|_\Xi^2 \tag{4.4}$$

where Π is the projection operator

$$\Pi T J = \arg\min_{h \in \mathcal{J}} \|TJ - h\|_\Xi^2 \tag{4.5}$$

and $TJ = g + \alpha P J$ is the Bellman operator; Ξ is a diagonal matrix with the stationary state distribution. $P = \{p_{ij}\}$ is the state transition probability matrix with p_{ij} the probability of the transition from state i to state j ($p_{ij} \in [0,1]$, $\sum_j p_{ij} = 1$). $g = [g_1, \dots, g_n]^\top$ is a vector with the expected immediate state transition costs: $g_i = \sum_j p_{ij} g(i,j)$.

In the LSTD setting, a sample-based version of the MSPBE optimization problem is considered:

$$\theta_r = \arg\min_{\theta \in \mathbb{R}^p} \sum_{i=1}^{n_s} \|g(x_i, x_i') + \alpha \phi(x_i')^\top r - \phi(x_i)^\top \theta\|^2, \tag{4.6}$$

$$r^* = \arg\min_{r \in \mathbb{R}^p} \sum_{i=1}^{n_s} \|\phi(x_i)^\top r - \phi(x_i)^\top \theta_r\|^2. \tag{4.7}$$

Solving these equations leads to the solution to LSTD [16]:

$$r^* = \left(\tilde{\Phi}^\top \tilde{\Phi} - \alpha \tilde{\Phi}^\top \tilde{\Phi}'\right)^{-1} \left(\tilde{\Phi}^\top G\right), \tag{4.8}$$

where

$$\begin{aligned} \tilde{\Phi} &= [\phi(x_1), ..., \phi(x_{n_s})]^\top, \\ \tilde{\Phi}' &= [\phi(x_1'), ..., \phi(x_{n_s}')]^\top, \\ G &= [g(x_1, x_1'), ..., g(x_{n_s}, x_{n_s}')]^\top. \end{aligned} \tag{4.9}$$

4.2.2 Model estimation and policy evaluation

As explained in [15], for the tabular case, LSTD is equivalent to (i): estimate first the state transition probabilities based on Maximum Likelihood (ML) and then (ii) use model-based policy evaluation. The ML model is computed by

$$P_{\mathrm{ML}} = \arg\max_{P} \mathrm{Prob}\left(\mathcal{X}|P\right), \tag{4.10}$$

with the constraint that $\sum_{j=1}^{n} p_{ij,\mathrm{ML}} = 1$. The well-known solution to this maximization problem is $P_{\mathrm{ML}} = \{p_{ij,\mathrm{ML}}\}$, $p_{ij,\mathrm{ML}} = q_{ij} / \sum_{j=1}^{n} q_{ij}$ where q_{ij} are the number of transitions from state i to state j in the data set [3]. By defining $N = \mathrm{diag}([\sum_{j=1}^{n} q_{1j}, \ldots, \sum_{j=1}^{n} q_{nj}])$, and $T = \{q_{ij}\}$, P_{ML} can be written as $P_{\mathrm{ML}} = N^{-1}T$. The expected state transition costs vector g can be found by averaging the transition costs of each state, which is equivalent to $g_{\mathrm{ML}} = N^{-1}s$, where $s = [s_1, ..., s_n]^\top$ is a vector which sums all the state transition costs in the data set corresponding to each state.

When the linear architecture in (4.3) is used, a similar equivalence can be obtained, see [84]. In fact, both P_{ML} and g_{ML} can be projected into the feature space, weighted with N (which is equivalent to weighting with the stationary state distribution Ξ if samples are drawn on-policy):

$$\begin{aligned} P_{\Phi,\mathrm{ML}} &= \left(\Phi^\top N \Phi\right)^{-1} \Phi^\top N P_{\mathrm{ML}} \Phi \\ g_{\Phi,\mathrm{ML}} &= \left(\Phi^\top N \Phi\right)^{-1} \Phi^\top N g_{\mathrm{ML}}. \end{aligned} \tag{4.11}$$

With parameter vector r, the cost vector is given by Φr. Furthermore, by exploiting the fact that next features $P\Phi$ are approximated by $\Phi P_{\Phi,\mathrm{ML}}$, it allows us to rewrite (4.1) as: $\Phi r^*_{\mathrm{ML}} = \Phi g_{\Phi,\mathrm{ML}} + \alpha \Phi P_{\Phi,\mathrm{ML}} r^*_{\mathrm{ML}}$. Therefore, a closed-form expression for r^*_{ML} can be found by:

$$\begin{aligned} r^*_{\mathrm{ML}} &= g_{\Phi,\mathrm{ML}} + \alpha P_{\Phi,\mathrm{ML}} r^*_{\mathrm{ML}} \\ &= (I - \alpha P_{\Phi,\mathrm{ML}})^{-1} g_{\Phi,\mathrm{ML}} \\ &= \left(I - \alpha \left(\Phi^\top N \Phi\right)^{-1} \Phi^\top N P_{\mathrm{ML}} \Phi\right)^{-1} \cdot \left(\left(\Phi^\top N \Phi\right)^{-1} \Phi^\top N g_{\mathrm{ML}}\right) \\ &= \left(I - \alpha \left(\Phi^\top N \Phi\right)^{-1} \Phi^\top T \Phi\right)^{-1} \left(\left(\Phi^\top N \Phi\right)^{-1} \Phi^\top s\right). \end{aligned} \tag{4.12}$$

Therefore, the optimal parameters can be computed as

$$r^*_{\mathrm{ML}} = \left(\Phi^\top N \Phi - \alpha \Phi^\top T \Phi\right)^{-1} \left(\Phi^\top s\right), \tag{4.13}$$

which is equivalent to (4.8) by noting that $\tilde{\Phi}^\top \tilde{\Phi} = \Phi^\top N \Phi$, $\tilde{\Phi}^\top \tilde{\Phi}' = \Phi^\top T \Phi$, and $\tilde{\Phi}^\top G = \Phi^\top s$.

4.2.3 Problem statement

In both LSTD and in the alternative equivalent method discussed in Section 4.2.2, no prior knowledge on state transition probabilities and expected immediate state transition costs is incorporated. However, for the alternative method based on the model estimation, prior knowledge can be incorporated by considering the Maximum a Posterior (MAP) estimation instead of the ML estimation, that is,

$$P_{\text{MAP}} = \arg\max_{P} \text{Prob}(\mathcal{X}|P)\text{Prob}(P) \tag{4.14}$$

where $\text{Prob}(P)$ is the prior probability density function of the state transition probability matrix. Instead of having a data-based method associated with P_{ML}, we aim to obtain a data-based method associated with P_{MAP}. In the next section, a suitable class of priors is found, such that the solution to (4.14) within such a class is tractable and which is later used by the proposed data-based method.

4.3 Adding priors in model estimation

We propose to use a Dirichlet prior for every row of the state transition probability matrix P. Let $\text{Dir}(\omega;\nu)$ denote the Dirichlet distribution, defined by

$$\text{Prob}(\omega_1,\ldots,\omega_D;\nu_1,\ldots,\nu_D) = \frac{\Gamma\left(\sum_{i=1}^{D}\nu_i\right)}{\prod_{i=1}^{D}\Gamma(\nu_i)}\prod_{i=1}^{D}\omega_i^{\nu_i-1}, \tag{4.15}$$

where $\omega = [\omega_1,\ldots,\omega_D]$ is an element of the simplex ($\sum_{i=1}^{D}\omega_i = 1$ with $\omega_i \geq 0$), and $\nu = [\nu_1,\ldots,\nu_D]$, $\nu_i > 0$ are the parameters of the Dirichlet distribution; $\Gamma(\cdot)$ is the Gamma function. Let p_i denote a row of P: $p_i \sim \text{Dir}(p_i;\beta_i)$, then $\beta_i = [\beta_{i1},\ldots,\beta_{in}]$ are the parameters of the prior. Since the Dirichlet prior is the conjugate prior to the categorical distribution, a closed-form expression for the posterior state transition probability matrix P_{MAP} can be obtained [82]:

$$\begin{aligned} P_{\text{MAP}} &= \arg\max_{P} \text{Prob}(\mathcal{X}|P)\text{Prob}(P) \\ &\propto \arg\max_{P} \left(\prod_{i=1}^{n}\prod_{j=1}^{n} p_{ij}^{q_{ij}}\right)\left(\prod_{k=1}^{n}\prod_{\ell=1}^{n} p_{k\ell}^{\beta_{k\ell}-1}\right) \\ &= \arg\max_{P} \prod_{i=1}^{n}\prod_{j=1}^{n} p_{ij}^{q_{ij}+\beta_{ij}-1}. \end{aligned} \tag{4.16}$$

Solving (4.16) with the constraint $\sum_{j=1}^{n} p_{ij,\text{MAP}} = 1$ leads to the closed-form expression

$$p_{ij,\text{MAP}} = \frac{q_{ij}+\beta_{ij}-1}{\sum_{j=1}^{n}(q_{ij}+\beta_{ij}-1)} \tag{4.17}$$

from which it becomes clear that if β_{ij} are integers, they can be seen as a pseudo-counter of transitions from state i to j. Using other prior distributions rather than the Dirichlet distribution, will not lead to such a closed-form expression. This is the main motivation to consider Dirichlet distributions.

Assumption 4.1. $\beta_{ij} \geq 1$ for all i, j.

Assumption 4.1 guarantees that $p_{ij,\text{MAP}} \in [0, 1]$ and $\sum_{j=1}^{n} p_{ij,\text{MAP}} = 1$. If $\beta_{ij} = 1$, it implies no pseudo-counts and a prior state transition probability of 0.

The posterior estimate of the state transition probability (4.17) in matrix notation is given by

$$P_{\text{MAP}} = \{p_{ij,\text{MAP}}\} = (N + N_0)^{-1}(T + T_0) \tag{4.18}$$

where $T_0 = \{\beta_{ij} - 1\}$. By defining $b = [b_1, \ldots, b_n]$, $b_i = \sum_{j=1}^{n}(\beta_{ij} - 1)$, matrix N_0 is given by $N_0 = \text{diag}(b)$. The posterior estimate of the expected immediate transition costs are given by

$$g_{\text{MAP}} = (N + N_0)^{-1}(s + s_0) \tag{4.19}$$

where s_0 captures prior knowledge on the expected immediate state transition costs g. This prior is formulated as $g_0 = N_0^{-1} s_0$, where s_0 is a design variable.

Remark 4.1. *By evaluating* (4.17)*, it can easily be verified that without measurements a prior model is given by $P_0 = N_0^{-1} T_0$, if for all i there is at least one j for which $\beta_{ij} > 1$. By adjusting β_{ij}, the certainty in this prior model can be altered, where higher values of β_{ij} correspond to a higher likelihood of state transition p_{ij}.*

Remark 4.2. *$\beta_{ij} = 1$ for all j, can be used to model no prior information, leading to $p_{ij,MAP} = p_{ij,ML}$ for all j.*

4.4 Maximum a posteriori LSTD and main results

The proposed MAP-LSTD method is introduced in 4.4.1. The main theorem is formulated in Section 4.4.2. In Section 4.4.3, a recursive implementation of MAP-LSTD is given. Section 4.4.4 discusses several characteristics and properties of the proposed method.

4.4.1 MAP-LSTD

LSTD finds the optimal parameters by solving the optimization problem of the MSPBE, given by (4.6) and (4.7). For the method which we coin Maximum a Posteriori LSTD (MAP-LSTD), this optimization problem is altered such that a prior is incorporated. A natural approach is to add a penalty term as a function of

the prior as regularization in the optimization problem [24, 39]. In this section we couple this penalty term to the Dirichlet prior.

We propose the following optimization problem, based on the LSTD optimization problem given by (4.6) and (4.7), where a specific penalty term is added to include the Dirichlet prior:

$$\theta_{r,\mathrm{MAP}} = \arg\min_{\theta\in\mathbb{R}^p} \sum_{i=1}^{n_s} \|g(x_i,x_i') + \alpha\phi(x_i')^\top r - \phi(x_i)^\top\theta\|^2 + \underbrace{\|G_0 + \alpha\bar{\Phi}'r - \bar{\Phi}\theta\|^2}_{\text{Penalty term}} \tag{4.20}$$

$$r^*_{\mathrm{MAP}} = \arg\min_{r\in\mathbb{R}^p} \sum_{i=1}^{n_s} \|\phi(x_i)^\top r - \phi(x_i)^\top\theta_{r,\mathrm{MAP}}\|^2 \tag{4.21}$$

where $\bar{\Phi} := N_0^{\frac{1}{2}}\Phi$, $\bar{\Phi}' := N_0^{-\frac{1}{2}}T_0\Phi$, and $G_0 := N_0^{-\frac{1}{2}}s_0$. Since N_0 is a diagonal matrix with non-negative terms, the computations of $N_0^{\frac{1}{2}}$ and $N_0^{-\frac{1}{2}}$ are relatively easy.

Solving the optimization problem given by (4.20) and (4.21) leads to a closed-form expression for r^*_{MAP}. This can be verified by first taking the partial derivative of (4.20) w.r.t. θ and equating it to 0:

$$0 = -2\tilde{\Phi}^\top\left(G + \alpha\tilde{\Phi}'r - \tilde{\Phi}\theta\right) - 2\bar{\Phi}^\top\left(G_0 + \alpha\bar{\Phi}'r - \bar{\Phi}\theta\right). \tag{4.22}$$

Since

$$\begin{aligned}
\bar{\Phi}^\top\bar{\Phi} &= \left(N_0^{\frac{1}{2}}\Phi\right)^\top\left(N_0^{\frac{1}{2}}\Phi\right) = \Phi^\top N_0\Phi,\\
\bar{\Phi}^\top\bar{\Phi}' &= \left(N_0^{\frac{1}{2}}\Phi\right)^\top\left(N_0^{-\frac{1}{2}}T_0\Phi\right) = \Phi^\top T_0\Phi,\\
\bar{\Phi}^\top G_0 &= \left(N_0^{\frac{1}{2}}\Phi\right)^\top\left(N_0^{-\frac{1}{2}}s_0\right) = \Phi^\top s_0,
\end{aligned} \tag{4.23}$$

then (4.22) can be written as

$$0 = \tilde{\Phi}^\top G + \alpha\tilde{\Phi}^\top\tilde{\Phi}'r - \tilde{\Phi}^\top\tilde{\Phi}\theta + \Phi^\top s_0 + \alpha\Phi^\top T_0\Phi r - \Phi^\top N_0\Phi\theta \tag{4.24}$$

and $\theta_{r,\mathrm{MAP}}$ becomes

$$\theta_{r,\mathrm{MAP}} = \left(\tilde{\Phi}^\top\tilde{\Phi} + \Phi^\top N_0\Phi\right)^{-1}\cdot\left(\tilde{\Phi}^\top G + \Phi^\top s_0 + \alpha\left(\tilde{\Phi}^\top\tilde{\Phi}' + \Phi^\top T_0\Phi\right)r\right). \tag{4.25}$$

Plugging this expression in (4.21), taking the partial derivative w.r.t. r, equating it to 0, and solving for r leads to:

$$r^*_{\text{MAP}} = \left(\tilde{\Phi}^\top \tilde{\Phi} + \Phi^\top N_0 \Phi - \alpha \left(\tilde{\Phi}^\top \tilde{\Phi}' + \Phi^\top T_0 \Phi\right)\right)^{-1} \cdot \left(\tilde{\Phi}^\top G + \Phi^\top s_0\right) \tag{4.26}$$

which is the closed-form solution of the MAP-LSTD optimization problem.

4.4.2 Main theorem

In this section, it is established that MAP-LSTD introduced in Section 4.4.1 is equivalent to building an MAP state transition model, and applying model-based policy evaluation subsequently. Following the steps of Section 4.2.2, but now using P_{MAP} and g_{MAP} according to (4.18) and (4.19) respectively rather than P_{ML} and g_{ML}, a method is obtained that uses system identification with a prior model.

Theorem 4.1. *The optimization problem given by* (4.20) *and* (4.21) *leads to the same estimate of* r *as when a MAP model is estimated according to* (4.16)*, projected as in* (4.11)*, and model-based policy evaluation is applied, given a user-defined Dirichlet prior of the system and data set* $\mathcal{X}$*.* ▲

Proof: Project P_{MAP} and g_{MAP} into the feature space, weighted with $(N + N_0)$:

$$\begin{aligned} P_{\Phi,\text{MAP}} &= \left(\Phi^\top (N + N_0)\Phi\right)^{-1} \Phi^\top (N + N_0) P_{\text{MAP}} \Phi \\ &= \left(\Phi^\top (N + N_0)\Phi\right)^{-1} \Phi^\top (T + T_0)\Phi \\ g_{\Phi,\text{MAP}} &= \left(\Phi^\top (N + N_0)\Phi\right)^{-1} \Phi^\top (N + N_0) g_{\text{MAP}} \\ &= \left(\Phi^\top (N + N_0)\Phi\right)^{-1} \Phi^\top (s + s_0). \end{aligned} \tag{4.27}$$

These two expressions can be used to solve the Bellman equation:

$$\begin{aligned} r^*_{\text{MAP}} &= \left(I - \alpha P_{\Phi,\text{MAP}}\right)^{-1} g_{\Phi,\text{MAP}}, \\ &= \left(I - \alpha \left(\Phi^\top (N + N_0)\Phi\right)^{-1} \Phi^\top (T + T_0)\Phi\right)^{-1} \\ &\quad \cdot \left(\left(\Phi^\top (N + N_0)\Phi\right)^{-1} \Phi^\top (s + s_0)\right). \end{aligned} \tag{4.28}$$

Therefore, the optimal parameters r^*_{MAP} of the system identification with policy evaluation method can be computed as

$$r^*_{\text{MAP}} = \left(\Phi^\top (N + N_0)\Phi - \alpha \Phi^\top (T + T_0)\Phi\right)^{-1} \left(\Phi^\top (s + s_0)\right). \tag{4.29}$$

Comparing (4.29) with (4.26) concludes that the same cost function parameters are obtained.

■

4.4.3 Recursive implementation

Let $\hat{r}_{k,\text{MAP}} = C_{k,\text{MAP}}^{-1} d_{k,\text{MAP}}$ be the estimate of r for MAP-LSTD, where

$$\begin{aligned} C_{k,\text{MAP}} &= \tilde{\Phi}^\top \tilde{\Phi} + \Phi^\top N_0 \Phi - \alpha \left(\tilde{\Phi}^\top \tilde{\Phi}' + \Phi^\top T_0 \Phi \right) \\ &= \sum_{i=1}^{k} \phi(x_i) \left(\phi(x_i) - \alpha \phi(x_i') \right)^\top + \Phi^\top N_0 \Phi - \alpha \Phi^\top T_0 \Phi. \end{aligned} \tag{4.30}$$

By initializing $C_{k,\text{MAP}}$ as $C_{0,\text{MAP}} = \Phi^\top N_0 \Phi - \alpha \Phi^\top T_0 \Phi + \delta I$, where δ is a small number to ensure that $C_{k,\text{MAP}}$ is always invertible, the recursive expression for $C_{k,\text{MAP}}$ becomes

$$C_{k,\text{MAP}} = C_{k-1,\text{MAP}} + \phi(x_k) \left(\phi(x_k) - \alpha \phi(x_k') \right)^\top. \tag{4.31}$$

To reduce the computational complexity of $C_{k,\text{MAP}}^{-1}$ from $\mathcal{O}(d^3)$ to $\mathcal{O}(d^2)$, the Sherman-Morrison formula can be used:

$$\begin{aligned} C_{k,\text{MAP}}^{-1} &= \left(C_{k-1,\text{MAP}} + \phi(x_i) \left(\phi(x_i) - \alpha \phi(x_i') \right)^\top \right)^{-1} \\ &= C_{k-1,\text{MAP}}^{-1} \\ &\quad - \frac{C_{k-1,\text{MAP}}^{-1} \phi(x_k) \left(\phi(x_k) - \alpha \phi(x_k') \right)^\top C_{k-1,\text{MAP}}^{-1}}{1 + \left(\phi(x_k) - \alpha \phi(x_k') \right)^\top C_{k-1,\text{MAP}}^{-1} \phi(x_k)}. \end{aligned} \tag{4.32}$$

The same can be done for $d_{k,\text{MAP}}$:

$$\begin{aligned} d_{k,\text{MAP}} &= \tilde{\Phi}^\top G + \Phi^\top s_0 \\ &= \sum_{i=1}^{k} \phi(x_i) g(x_i, x_i') + \Phi^\top s_0. \end{aligned} \tag{4.33}$$

One can initialize $d_{k,\text{MAP}}$ as $d_{0,\text{MAP}} = \Phi^\top s_0$:

$$d_{k,\text{MAP}} = d_{k-1,\text{MAP}} + \phi(x_k) g(x_k, x_k'). \tag{4.34}$$

The recursive update steps (4.32) and (4.34) are equivalent to the recursive update steps of LSTD. They only differ from the initialization of $C_{0,\cdot}$ and $d_{0,\cdot}$, which now depends on the Dirichlet prior. Due to this, MAP-LSTD converges under the same assumptions as LSTD (see, e.g., [12]).

Algorithm 4.1. (Recursive MAP-LSTD) *Define the prior in terms of N_0, T_0, and s_0. Initialize $C_{k,MAP}$ and $d_{k,MAP}$ as $C_{0,MAP}$ and $d_{0,MAP}$. Then, for every $k \in K$:*

1. *Update $C_{k,MAP}^{-1}$ and $d_{k,MAP}$ according to* (4.32) *and* (4.34).
2. *Compute new parameters estimate $\hat{r}_{k,MAP} = C_{k,MAP}^{-1} d_{k,MAP}$.*

4.4.4 Characteristics of MAP-LSTD

The MAP-LSTD method has interesting characteristics which generalize the ordinary LSTD method. Consider the informativeness of the prior: if there is no prior information, N_0, T_0, and s_0 are identically zero, and MAP-LSTD boils down to LSTD. If only prior information is available on some (more relevant) states, the method allows us to incorporate only this partial informative prior. In states where no prior information is available, the prior knowledge does not have to be considered (see Remark 4.1). This is an important feature to make the MAP-LSTD method applicable to large-scale systems where it is infeasible to define a prior to every state.

Another interesting concomitant of the usage of the Dirichlet prior is that it is possible to learn the prior from data samples. These can for example be previously obtained data samples or samples that are (heuristically) generated by a model or simulator. Evaluating (4.18) and (4.26) show that the Dirichlet prior is equivalent to adding pseudo-counts. In the exact, tabular representation case, pseudo-counts $\bar{q}_{ij}$ can be leveraged to the prior by defining $\beta_{ij} = \bar{q}_{ij} + 1$. In the linear representation case, the terms $\Phi^\top N_0 \Phi$, $\Phi^\top T_0 \Phi$, and $\Phi^\top s_0$ in (4.26), could be replaced by $\breve{\Phi}^\top \breve{\Phi}$, $\breve{\Phi}^\top \breve{\Phi}'$, and $\breve{\Phi}^\top \breve{G}$ respectively. Now, $\breve{\Phi}$, $\breve{\Phi}'$, and $\breve{G}$ are the data samples used to obtain the prior, which have the same structure as in (4.9).

In many contexts one has a simulator/model of a process and experimental data is not available at the design phase. In such cases one can compute the prior by simulated experience. While heuristic, this provides a simple and effective way to compute the prior, as we will show through an example in the next section.

4.5 Numerical example

In this section, MAP-LSTD is compared with LSTD by analyzing a balancing task for a linearized model of a cart-pole, described in [24, 27]. The state x of the system consists of the position (y) and velocity ($\dot{y}$) of the cart, and the angle (ϑ) and angular velocity ($\dot{\vartheta}$) of the pole, while input u is a force acting on the cart. Details of the linearized state-space system can be found in [24, 27]. Parameters of the system are the mass of the cart M, the mass and length of the pole m and l, respectively, and the friction coefficient of the cart on the ground b. The system is discretized with a sample time of 0.1 [s]. A discounted, quadratic cost function is used, given as $V(x_k) = \sum_{i=k}^{\infty} \alpha^{i-k} \left(x_i^\top Q x_i + u_i^\top R u_i \right)$, with $Q = \mathrm{diag}([1, 0, 100, 0])$, $R = 0.1$, and $\alpha = 0.95$. The control input is defined as $u_k = K x_k$.

Two systems are simulated; a reference system to learn the prior and a true system. This setting may be seen as having an (approximate) model of a true, real system, which is not perfectly modelled due to model-mismatch. The overall goal is to estimate the cost function of the true, real system. The reference system is used

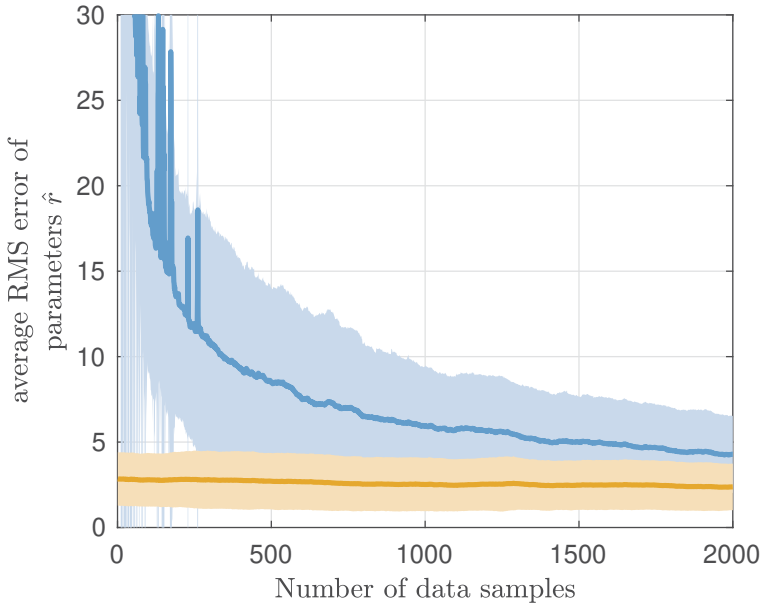


Figure 4.2. Comparison of LSTD (—) with MAP-LSTD (—). Regions below the mean indicate plus or minus the standard deviation of the average RMS error.

to learn the prior, whereas the experiments are executed on the true system. The parameters of the reference system are: $M = 0.50$ [kg], $m = 0.50$ [kg], $l = 0.60$ [m], and $b = 0.10$ [N/(ms)]. On the other hand, the parameters of the true system are: $M = 0.49$ [kg], $m = 0.51$ [kg], $l = 0.61$ [m], and $b = 0.11$ [N/(ms)].

K is the optimal control policy derived from the LQR solution of the reference system. It is well-known that the cost function of both systems can be computed exactly given control policy K. The objective of (MAP-)LSTD is to estimate the cost function of the (unknown) true system. Eleven features are used: the ten possible products of individual state $(y^2, y\dot{y}, \dot{y}^2, \dots)$, plus a constant term.

In Figure 4.2, the average RMS error of the cost function parameters (computed as $\frac{1}{n_{\text{mc}}} \sum_{i=1}^{n_{\text{mc}}} \sqrt{\|r - \hat{r}_k\|^2}$) over $n_{\text{mc}} = 100$ simulations is given, along with its variance. The prior is learned with 4000 samples from the reference system. In the initial phase, the estimation of MAP-LSTD is already of good quality, since the estimate relies mainly on the prior (simulations of the approximate model) instead of the initial data samples of the true system. However, they are not perfect yet, due to the model-mismatch of the reference model. When more data samples become available, the average RMS error remains below the average RMS error of LSTD. In general, the variance of MAP-LSTD is lower than the variance of LSTD. Figure 4.3 shows the convergence of the (normalized) individual parameters of the

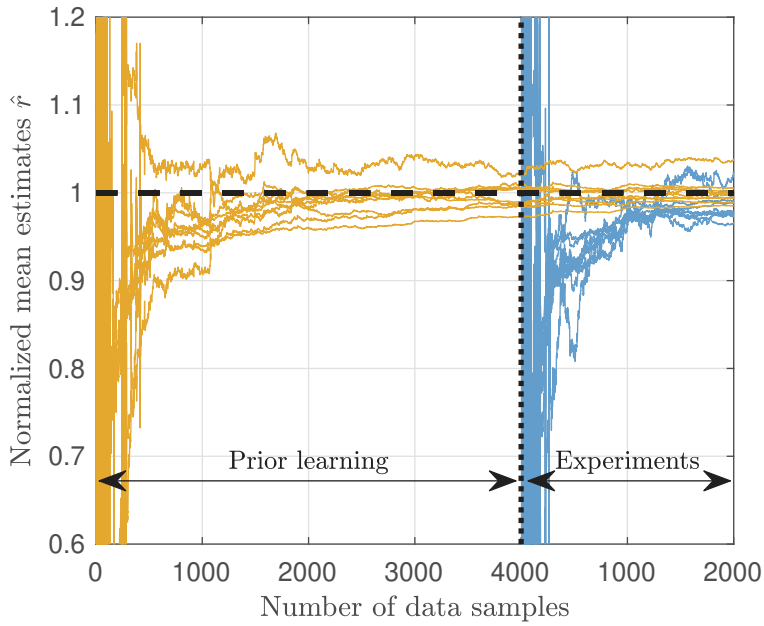


Figure 4.3. Two phases of MAP-LSTD (—) (and LSTD (—)). In the first phase, the prior is learned on the reference system. In the second phase data samples are obtained from the true system.

cost function. Here it is visible that during prior learning, the parameters do not converge to their true values (convergence if normalized value equals 1). After a considerable amount of experimental data, the impact of the prior will fade away and the true values are obtained.

4.6 Conclusions and future work

This chapter proposes MAP-LSTD, a new data-based policy evaluation method that does not explicitly compute a model. MAP-LSTD combines prior knowledge of the system with data samples to estimate the parameters of the cost function with a linear architecture. MAP-LSTD is equivalent to estimating the state transition probabilities of the Markov decision process through MAP, and computing the cost function through model-based policy evaluation. By incorporating prior knowledge it can outperform LSTD, especially in scarce data settings. In fact, through examples, it is shown that MAP-LSTD converges faster with smaller variance when compared to LSTD. Future research directions include generalizations to include semi-gradients [115], and extending the policy evaluation method to policy improvement.

Chapter 5

Least-Squares Temporal Difference with Expected Eligibility Traces

Abstract - Temporal Difference (TD) and Least-Squares Temporal Difference (LSTD) are related methods to estimate the value function of a Markov Decision Process (MDP). While TD is a direct method using local data to update the value function estimate, LSTD is a Bellman projected equation method using full data to compute a one-time estimate. TD(λ) and LSTD(λ) extend TD and LSTD with eligibility traces. While estimating the value function, TD(λ) and LSTD(λ) use actual histories of features as traces. Recently, expected eligibility traces have been proposed for TD(λ) to not only include actual histories, but also all potential histories of features that could have occurred based on the model or the available data. While this idea can account for non-linear feature architectures, here we limit ourselves to linear feature architectures with full data updates in the context of LSTD. We show that, in striking contrast with the direct versions, an extension of LSTD to include the theoretical expected eligibility traces is equivalent to LSTD without eligibility traces (LSTD(0)). We obtain a similar result if we consider mixed eligibility traces; a combination of expected eligibility traces and ordinary eligibility traces. In fact, we show that LSTD with theoretical mixed eligibility traces is equivalent to LSTD(λ') for a given λ' that captures both the decay of the eligibility trace, as well as the balance between the expected eligibility trace and the ordinary trace. Furthermore, we consider alternative methods LSET(λ) and LSET(η,λ), which rely on the empirical means of the eligibility traces rather than the theoretical expected eligibility traces, and show that their value estimates converges to those of LSTD(0) and LSTD(λ').

This chapter is based on: R.A.C. van Zuijlen and D.J. Guerreiro Tomé Antunes, "Least-Squares Temporal Difference with Expected Eligibility Traces", *Machine Learning Journal*, vol. 114, 2025.

5.1 Introduction

For a Markov Decision Process (MDP), the value function under a specified policy can be determined through policy evaluation [11], provided that the model (state transition probabilities) is known. When the model is unknown, policy evaluation can be carried out through data samples obtained from system trajectories. Approximate methods can be pursued when computing the exact value function is impractical. This can happen due to a large, or potentially infinite, number of states. The calculation of the value function in MDPs is important in various applications, serving, e.g., as a step towards identifying the optimal policy through policy iteration.

Popular methods for learning the value function based on data samples are Temporal Difference (TD) and Monte Carlo (MC) methods (for overviews, see e.g. [11, 24, 115]). One-step TD methods process the immediate rewards at each time instance by updating the value function when the state is visited, whereas MC methods wait till the episode is over before an update is made. The advantage of MC methods over TD methods is that they are unbiased, despite having a higher variance with respect to TD methods. Methods that lie between the extremes of one-step TD and MC methods are called multi-step return methods. Through eligibility traces, multi-step return methods can be adjusted for value function updates to be carried out online in a faster way, and the bias-variance trade-off can be balanced [114].

Typically, these direct online methods require an enormous amount of data before convergence is achieved. In many applications (e.g. mechanical systems), however, data-inefficient methods are impractical, since data gathering is often costly and time-consuming. To increase the data efficiency, alternatives such as LSTD(0) for the one-step TD methods, have been developed that use the full data to compute value estimates [16]. LSTD(λ) is an extension of LSTD(0) to include eligibility traces. The parameter λ allows to trade the bias of value approximation of the method by its variance; $\lambda = 0$ corresponds to a one-step TD method that has low variance but high bias, and $\lambda = 1$ corresponds to an MC method that has low bias but high variance [15]. Convergence analyses of LSTD(λ) are given in [12, 81, 128].

Eligibility traces capture actual histories of trajectories as traces that occurred in the data. Recent work uses expected eligibility traces, instead of ordinary eligibility traces, to not only capture actual histories, but also all potential histories in a method coined ET(λ) [121]. Expected eligibility traces are closely related to source traces [89], and successor representations [25]. The advantage of expected eligibility traces is that not only the value is updated for states that have occurred in the current trajectory, but also for states that could have occurred, even if they did not occur in the particular current trajectory. In this way, improvements in

terms of convergence speed can be achieved. In the present chapter, we consider two ways of achieving this, also considered in [121]. The first is to compute a theoretical mean over possible past trajectories that lead to the current state. Here we propose to use a backward model to compute this theoretical mean which can be obtained when the process is ergodic. The second is to use an empirical mean, by which the eligibility traces of past trajectories that also led to the current state are averaged. Figure 5.1 illustrates this latter case. Expected eligibility traces, however, do induce bias. In [121] also mixed eligibility traces are introduced to balance between expected eligibility traces on the one hand, and ordinary eligibility traces on the other hand.

In this chapter, we introduce the concept of expected eligibility traces for the on-policy projected equation (LSTD) setting, that is, when full data rather than local data is used to compute the expected eligibility traces. However, in striking contrast with the direct expected eligibility traces method ET(λ) of [121], we show that its least-squares alternative is mathematically equivalent to LSTD(0) when theoretical expected traces are used to compute the expected traces. The theoretical expected traces are computed using a backward model of the MDP. Since this model is unavailable in practice, we propose LSET(λ) as a least-squares alternative for ET(λ) using empirical means of eligibility traces. While using empirical means for expected eligibility traces to compute a one-time estimate of the value function with full data differs from LSTD(0), we still show that the estimates of the former method converge to the ones of the latter as more data becomes available. The positioning of LSET(λ) with respect to other existing methods in the literature is visualized in Figure 5.2.

Moreover, we show that the least-squares alternative of the direct method with mixed eligibility traces ET(η,λ) is mathematically equivalent to LSTD(λ') when theoretical expected traces are used, and where $\lambda' = \eta \cdot \lambda$. Here, η is a balance parameter used in the mixed eligibility trace. This leads to the following important insight: by tuning the parameter λ in LSTD(λ), we are not only tuning the decay of the eligibility traces, but also balancing between the expected value of the eligibility trace and the value derived from the data set. As shown also in Figure 5.2, we can similarly conceive a method LSET(η,λ) that uses empirical, rather than theoretical, means for mixed eligibility traces to compute a one-time estimate of the value function with full data. Again, this method differs from LSTD(λ'), but the corresponding estimates converge to those of LSTD(λ').

Two interesting conclusions of the chapter are the following. First, even though a model is often not available to exactly compute expected eligibility traces, it is not needed. Even if such a model would be available, the value estimates obtained with (mixed) expected eligibility traces would be equivalent to those of LSTD(0) (LSTD(λ)), which does not require a model. Moreover, since LSET(λ) (or LSET(η,λ)) is an approximate version of the exact (mixed) eligibility traces,

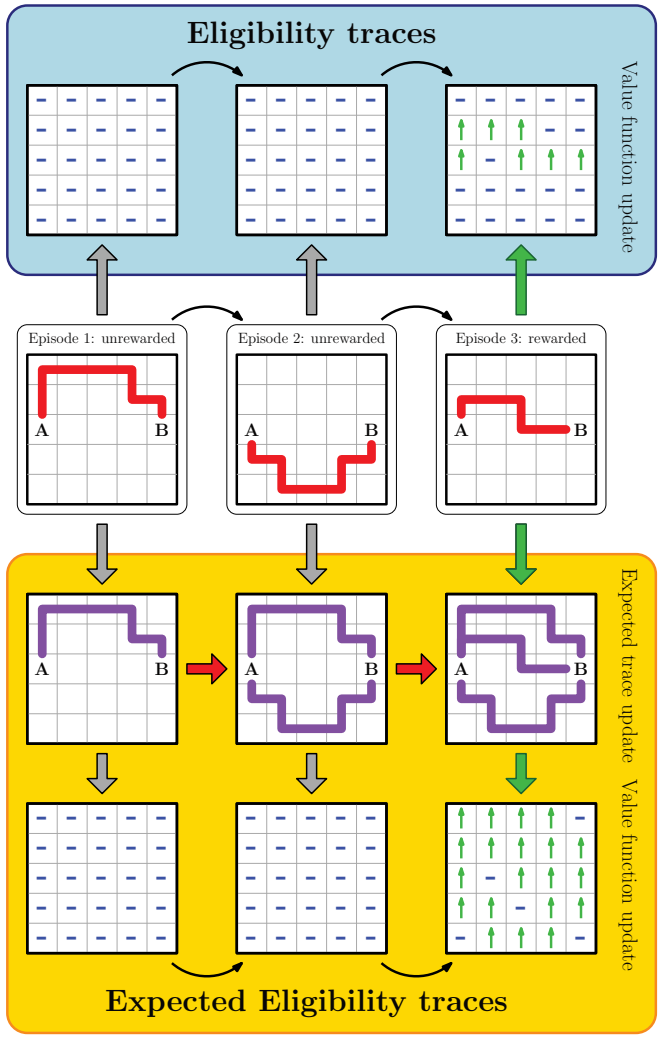


Figure 5.1. Illustration of the concept of expected eligibility traces through a toy problem where an agent starting at A receives a reward when it reaches B with a certain probability. The agent moves with a stochastic policy. In this example, the first two episodes are unrewarded, and credits are only obtained in the third episode. While eligibility traces only assign credits to states visited at rewarded episodes, expected eligibility assigns credits to states that could have also potentially been visited in each episode. One way of asserting these states is based on historical data taking empirical means of previous eligibility traces.

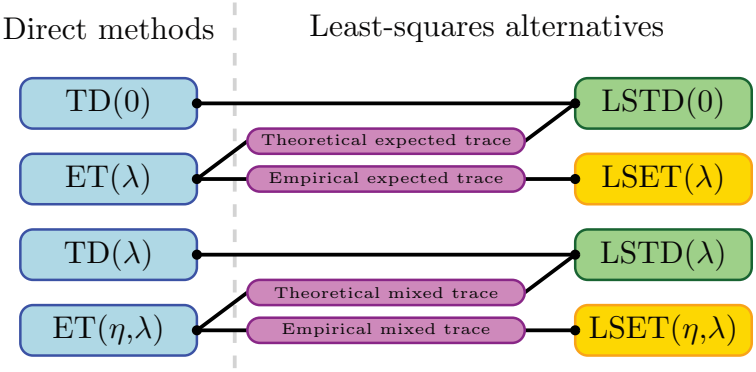


Figure 5.2. Positioning of LSET(λ) and LSET(η,λ) ( ) with respect to other existing methods in literature. The direct methods ( ) use the semi-gradient to update the value function estimate each time a new sample is available, whereas their least-squares alternatives ( ) use all the samples simultaneously (full data) to update the value function estimate.

where empirical means replace the theoretical ones, one can expect better results to be obtained with LSTD(0) (or LSTD(λ)). This is in fact what we observe in the numerical results we provide.

This chapter is structured as follows. In Section 5.2, the λ-weighted Bellman equation and LSTD(λ) are reviewed. LSTD with expected eligibility traces is introduced in Section 5.3, together with the LSET(λ) method. Section 5.4 elaborates on LSTD with mixed eligibility traces, proposes the LSET(η,λ) method, and discusses the impact of the results on the interpretation of eligibility traces in the LSTD setting. Examples to illustrate the performance of LSET(λ) and LSET(η,λ) are given in Section 5.5. Finally, the conclusions are drawn in Section 5.6.

Notation: The spectral radius of a matrix is denoted by $\rho(\cdot)$. $\mathbb{E}[x]$ is the expected value of x. A diagonal matrix with entries $a_1, ..., a_n$ is denoted by $\text{diag}([a_1, ..., a_n])$.

5.2 Background

In this chapter, we focus on Markov decision problems (MDPs) with an infinite-horizon discounted value, although the ideas presented could also be extended to other problems (e.g. shortest path problems). Denoting the state space by X and the control space by U, let $x_t \in X$ and $u_t \in U$ denote the state and control input at time $t \in \mathbb{Z}$, respectively. The value of a policy $\pi : X \to U$ for each state x_t is

given by

$$V_\pi(x_t) = \lim_{H\to\infty} \mathbb{E}\left[\sum_{i=t}^{H-1} \alpha^i r(x_i, x_i')|x_t\right] = \mathbb{E}\left[r(x_t, x_t')|x_t\right] + \alpha V_\pi(x_t') \tag{5.1}$$

where $x_t' := x_{t+1}$ is the next state, and $\alpha \in (0, 1]$ is the discount factor. Disturbances $w_t \in W$ are assumed to be mutually independent, and the state evolves according to $x_t' = f(x_t, u_t, w_t)$, $f : X \times U \times W \to X$, where f is a time-invariant function. $r(x_t, x_t')$ are the state transition rewards $r : X \times X \to \mathbb{R}$. X, U and W are finite sets, where the total number of states is denoted by n. The goal of policy evaluation is to find value function (5.1) for every initial state x_0 of the MDP under (fixed) policy π. The value function is estimated based on data set $\mathcal{X} := \{(x_i, x_i', r(x_i, x_i'))|i = 0, ..., K\}$ obtained under policy π, where K is the time instance where the data set trajectory ends. Although a single trajectory is considered, the ideas can easily be extended to multiple trajectories. Since the policy is fixed, u_t is not relevant for the value function estimation.

Temporal difference learning methods aim to find the solution to the Bellman equation, given in (5.1). This equation can also be written as

$$V = \bar{r} + \alpha P V \tag{5.2}$$

where $V \in \mathbb{R}^n$ is the value vector. $P = \{p_{ij}\} \in \mathbb{R}^{n\times n}$ is the state transition probability matrix with p_{ij} the probability of the transition from state i to state j ($p_{ij} \in [0, 1]$, $\sum_j p_{ij} = 1$). $\bar{r} = [r_1, \ldots, r_n]^\top$ is a vector with the expected immediate state transition rewards: $r_i = \sum_j p_{ij} r(i, j)$. The following standard assumption will be important in the sequel (see Section 5.3.1).

Assumption 5.1. The Markov chain corresponding to the transition probability matrix P is ergodic.

In many practical applications, an exact representation of the value function is infeasible, and an approximation is considered. One possibility to approximate the value function is a linear architecture, defined as

$$\tilde{V}(x, \beta) = \phi(x)^\top \beta \tag{5.3}$$

where $\phi : X \to \mathbb{R}^p$ is the feature vector and β is the parameter vector. With this linear architecture, the feasible set of value functions is $\mathcal{V} := \{\Phi\beta|\beta \in \mathbb{R}^p\}$, where $\Phi \in \mathbb{R}^{n\times p}$ is the feature matrix.

Assumption 5.2. Feature matrix Φ has rank p.

Assumption 5.2 is needed for the representation of a value function in $\mathcal{V}$ by a parameter vector β to be unique. The exact tabular case can be captured by picking $p = n$, $\phi_i(x) = 1$ if $x = i$, $\phi_i(x) = 0$ otherwise, in which case each parameter β_i corresponds to the value of state i.

The remainder of this section is organized as follows. In Sections 5.2.1 and 5.2.2, the λ-weighted multi-step return Bellman equation, which form the fundamental basis of eligibility trace methods, and LSTD(λ) are reviewed respectively. The definition of expected eligibility traces is given in Section 5.2.3.

5.2.1 λ-weighted multi-step return Bellman equation

Equation (5.2) is the one-step Bellman equation. Alternatively, the λ-weighted multi-step return can be defined as

$$V = r^{(\lambda)} + \alpha P^{(\lambda)} V, \tag{5.4}$$

where

$$P^{(\lambda)} = (1-\lambda) \sum_{l=0}^{\infty} \alpha^l \lambda^l P^{l+1}, \tag{5.5}$$

$$r^{(\lambda)} = (I - \alpha\lambda P)^{-1} \bar{r}, \tag{5.6}$$

and $\lambda \in [0, 1)$ is the decay parameter. The advantage of considering (5.4) instead of (5.2) is that the modulus of contraction is smaller when λ increases, and thus the bias of projected equation methods approximating the solution to (5.2) decreases [11]. However, increasing λ is at the expense of an increase in variance when data samples are used to estimate V [11].

When a linear architecture is used as value function approximation, as defined in (5.3), parameters β can be found by solving the projection equation, given by

$$C^{(\lambda)} \beta = d^{(\lambda)} \tag{5.7}$$

where

$$C^{(\lambda)} = \Phi^\top \Xi (I - \alpha P^{(\lambda)}) \Phi, \tag{5.8}$$

$$d^{(\lambda)} = \Phi^\top \Xi r^{(\lambda)}, \tag{5.9}$$

and $\Xi = \text{diag}([\xi_1, ..., \xi_n])$ is a diagonal matrix with the stationary state distribution of the Markov process $(\xi_1, ..., \xi_n)$ as diagonal entries.

5.2.2 LSTD(λ)

LSTD(λ) is a method that uses eligibility traces to estimate $C^{(\lambda)}$ and $d^{(\lambda)}$ to find parameters β [15]. An eligibility trace $z_k \in \mathbb{R}^p$, $k \in \{0, ..., K\}$ is defined as

$$\begin{aligned} z_k &= \sum_{i=0}^{k} (\alpha\lambda)^{k-i} \phi(x_i) \\ &= \alpha\lambda z_{k-1} + \phi(x_k) \end{aligned} \tag{5.10}$$

where z_{-1} is a vector of zeros. Given data set $\mathcal{X}$, the estimates for $C^{(\lambda)}$ and $d^{(\lambda)}$ at time instance k are

$$C_k^{(\lambda)} = \frac{1}{k+1} \sum_{i=0}^{k} z_i (\phi(x_i) - \alpha\phi(x_i'))^\top \tag{5.11}$$

$$d_k^{(\lambda)} = \frac{1}{k+1} \sum_{i=0}^{k} z_i r(x_i, x_i'). \tag{5.12}$$

The estimates of parameters $\hat{\beta}_k$ at time instance k can subsequently be computed as

$$\hat{\beta}_k = \left(C_k^{(\lambda)}\right)^{-1} d_k^{(\lambda)}. \tag{5.13}$$

Note that the value parameters $\hat{\beta}_k$ are updated based on the full data available up to time k, which differs from the direct version TD(λ) where value estimate updates only use data at time k. Equation (5.13) can alternatively be written as

$$\hat{\beta}_k = \left(\mathcal{Z}^\top \tilde{\Phi} - \alpha \mathcal{Z}^\top \tilde{\Phi}'\right)^{-1} \left(\mathcal{Z}^\top R\right) \tag{5.14}$$

where

$$\begin{aligned} \mathcal{Z} &= [z_0, ..., z_k]^\top \\ \tilde{\Phi} &= [\phi(x_0), ..., \phi(x_k)]^\top \\ \tilde{\Phi}' &= [\phi(x_0'), ..., \phi(x_k')]^\top \\ R &= [r(x_0, x_0'), ..., r(x_k, x_k')]^\top \end{aligned} \tag{5.15}$$

to find the parameters $\hat{\beta}_k$ directly on data set $\mathcal{X}$. In ordinary LSTD without eligibility traces ($\lambda = 0$), (5.14) boils down to

$$\hat{\beta}_k = \left(\tilde{\Phi}^\top \tilde{\Phi} - \alpha \tilde{\Phi}^\top \tilde{\Phi}'\right)^{-1} \left(\tilde{\Phi}^\top R\right). \tag{5.16}$$

Convergence of LSTD(λ) is proven in [12, 81, 128], where it is shown that

$$\begin{aligned} C_K^{(\lambda)} &\to C^{(\lambda)}, \\ d_K^{(\lambda)} &\to d^{(\lambda)}, \end{aligned} \tag{5.17}$$

as K converges to infinity with probability 1.

5.2.3 Expected eligibility traces

Eligibility traces, as defined in (5.10), are a function of the immediate history of features. Rather than limiting ourselves to these immediate histories, we can consider all potential histories; histories of features that could have occurred based on the data. One method which accounts for this is the method of expected eligibility traces. The expected eligibility trace $e(x_k)$ for the case of linear features is defined as

$$e(x_k) := \mathbb{E}\left[\sum_{i=0}^{\infty} (\alpha\lambda)^i \phi(x_{k-i}) | x_k\right] \tag{5.18}$$

and was introduced in [121, Equation 6]. Besides the expected value of the trace, another important difference between the ordinary eligibility traces as defined in (5.10) and the expected eligibility traces is the infinite sum. Instead of only considering a finite backward horizon, expected eligibility traces consider an infinite, backward horizon. In the next section, we will show that the main result holds for both finite and infinite, backward horizons. In the remainder of this chapter, we aim to extend the linear architecture ideas of LSTD with expected eligibility traces, while applying the update of the parameters based on full data.

5.3 LSTD with expected eligibility traces

In this section, we investigate the possibilities of applying the concept of expected eligibility traces with full data updates as in LSTD. Firstly, a more convenient formulation of the expected eligibility traces is provided in Section 5.3.1 assuming that the model (state transition probability matrix) is known. Secondly, in Section 5.3.2, the main result of LSTD with expected eligibility traces is given. Section 5.3.3 provides a method to include expected eligibility traces without model knowledge using empirical means.

5.3.1 Preliminary results

Due to Assumption 5.1 and the fact that we consider the underlying Markov chain for all $k \in \mathbb{Z}$ (initialized at minus infinity), we can consider a backward state

transition probability matrix $\bar{P} = \{\bar{p}_{ij}\} \in \mathbb{R}^{n \times n}$ which maps the current state to the possible previous states [98, Sec 4.8]. Recall that $(\xi_1, ..., \xi_n)$ is the stationary state distribution of the Markov chain. Then $\bar{P}$ is also a Markov chain and can be computed as [98, Sec 4.8]:

$$\bar{p}_{ij} = \frac{\xi_j p_{ji}}{\xi_i}. \tag{5.19}$$

Matrix $\bar{P}$ has the same properties as the forward state transition probability matrix P, thus $\bar{p}_{ij} \in [0, 1]$ and $\sum_j \bar{p}_{ij} = 1$ for all i. For the tabular feature setting, it follows that

$$\bar{P}^\top \phi(x_k) = \mathbb{E}\left[\phi(x_{k-1}) | x_k\right]. \tag{5.20}$$

When approximations are used as features, the backward feature transformation matrix can be approximated by projecting $\bar{P}$ into the feature space, weighting with the stationary state distribution

$$\bar{P}_\Phi = (\Phi^\top \Xi \Phi)^{-1} \Phi^\top \Xi \bar{P} \Phi \tag{5.21}$$

where $\bar{P}_\Phi \in \mathbb{R}^{p \times p}$. This projection is a common approach to approximate the feature transformation matrix, see e.g. [84] where the forward state transition probability matrix is projected into the feature space.

Using the transformation matrix $\bar{P}_\Phi$, the expected previous feature vector can be approximated based on the current feature vector:

$$\bar{P}_\Phi^\top \phi(x_k) \approx \mathbb{E}\left[\phi(x_{k-1}) | x_k\right]. \tag{5.22}$$

When features are used to approximate the value function, it is reasonable to use an approximation for the expected previous features. The results hereafter hold exactly for the case of exact features. The approximate case is discussed in the sequel.

Proposition 5.1. *For the tabular feature setting, the expected eligibility trace, as defined in* (5.18)*, can equivalently be written as*

$$e(x_k) = \Psi_\lambda \phi(x_k), \qquad k \in \mathbb{Z}, \tag{5.23}$$

where $\Psi_\lambda := \sum_{i=0}^{\infty} \left(\alpha\lambda\bar{P}^\top\right)^i = \left(I - \alpha\lambda\bar{P}^\top\right)^{-1}$*, since* $\rho(\alpha\lambda\bar{P}^\top) < 1$. ▲

Proof: The expected eligibility trace can be written as

$$\begin{aligned} e(x_k) &= \mathbb{E}\left[\phi(x_k) + \alpha\lambda\phi(x_{k-1}) + (\alpha\lambda)^2 \phi(x_{k-2}) + ... | x_k\right] \\ &= \mathbb{E}\left[\phi(x_k) | x_k\right] + \alpha\lambda\mathbb{E}\left[\phi(x_{k-1}) | x_k\right] + (\alpha\lambda)^2 \mathbb{E}\left[\phi(x_{k-2}) | x_k\right] + ... \end{aligned} \tag{5.24}$$

due to linearity. The expected previous feature vectors can be computed using $\bar{P}$ as a function of $\phi(x_k)$:

$$\begin{aligned}
\mathbb{E}\left[\phi(x_k)|x_k\right] &= \phi(x_k) \\
\mathbb{E}\left[\phi(x_{k-1})|x_k\right] &= \bar{P}^\top \phi(x_k) \\
\mathbb{E}\left[\phi(x_{k-2})|x_k\right] &= \left(\bar{P}^\top\right)^2 \phi(x_k) \\
&\vdots \\
\mathbb{E}\left[\phi(x_{k-\ell})|x_k\right] &= \left(\bar{P}^\top\right)^\ell \phi(x_k)
\end{aligned} \tag{5.25}$$

Then,

$$\begin{aligned}
e(x_k) &= \phi(x_k) + \alpha\lambda\bar{P}^\top\phi(x_k) + \left(\alpha\lambda\bar{P}^\top\right)^2\phi(x_k) + ... \\
&= \left(I + \alpha\lambda\bar{P}^\top + \left(\alpha\lambda\bar{P}^\top\right)^2 + ...\right)\phi(x_k) \\
&= \sum_{i=0}^{\infty}\left(\alpha\lambda\bar{P}^\top\right)^i\phi(x_k).
\end{aligned} \tag{5.26}$$

By defining $\Psi_\lambda := \sum_{i=0}^{\infty}\left(\alpha\lambda\bar{P}^\top\right)^i$, (5.23) is obtained.

By combining the properties of $\bar{P}$ and the Perron-Frobenius theorem eigenvalue inequality, it can be found that $\rho\left(\bar{P}^\top\right) \leq 1$. Since $\lambda \in [0, 1)$ and $\alpha \in (0, 1]$, a strict inequality is obtained, and $\rho\left(\alpha\lambda\bar{P}^\top\right) < 1$. With this strict inequality, it allows us to use the Neumann series.

■

When approximations are used as features, we have

$$\mathbb{E}\left[\phi(x_{k-\ell})|x_k\right] \approx \left(\bar{P}_\Phi^\top\right)^\ell\phi(x_k), \tag{5.27}$$

which leads to

$$e(x_k) \approx \Psi_\Phi\phi(x_k) \tag{5.28}$$

where $\Psi_\Phi = \sum_{i=0}^{\infty}\left(\alpha\lambda\bar{P}_\Phi^\top\right)^i$. Here, we cannot apply the Neumann series directly, since one cannot in general assume that $\rho(\bar{P}_\Phi^\top) \leq 1$. There are several options to circumvent this problem. The first option is to choose α and λ such that $\rho(\alpha\lambda\bar{P}_\Phi^\top) < 1$. Another option is to consider not using the infinite, backward horizon expected eligibility trace, but approximating it with a finite backward horizon $\bar{K}$:

$$\tilde{e}(x_k) := \mathbb{E}\left[\sum_{i=0}^{\bar{K}}(\alpha\lambda)^i\phi(x_{k-i})|x_k\right]. \tag{5.29}$$

This leads to $\tilde{e}(x_k) \approx \bar{\Psi}_\Phi \phi(x_k)$ where $\bar{\Psi}_\Phi := \sum_{i=0}^{\bar{K}} \left(\alpha\lambda \bar{P}_\Phi^\top\right)^i$, which has a bounded solution. This formulation is close to the ordinary eligibility traces, as defined in (5.10), since both have a finite backward horizon. In the remainder we will continue with Ψ_λ, although the same results hold when Ψ_λ is replaced by Ψ_Φ or $\bar{\Psi}_\Phi$.

5.3.2 Main result for expected eligibility traces

Given matrix Ψ_λ and data set $\mathcal{X}$, parameters $\hat{\beta}_k$, $k \in \{0, ..., K\}$ can be estimated as

$$\hat{\beta}_k = (\mathcal{E}^\top \tilde{\Phi} - \alpha \mathcal{E}^\top \tilde{\Phi}')^{-1} \mathcal{E}^\top R \tag{5.30}$$

where

$$\begin{aligned} \mathcal{E} &= [e(x_0), ..., e(x_k)]^\top \\ &= [\Psi_\lambda \phi(x_0), ..., \Psi_\lambda \phi(x_k)]^\top. \end{aligned} \tag{5.31}$$

This solution shows many similarities with LSTD(λ) in (5.14), but uses $e(x_k) = \Psi_\lambda \phi(x_k)$, as derived in Proposition 5.1, instead of z_k. Interestingly, we show that an explicit calculation or estimation of Ψ_λ (which needs model knowledge through $\bar{P}$) is not required. In fact, the next result shows that LSTD with the theoretical expected eligibility traces is equivalent to LSTD(0).

Theorem 5.1. *The value parameter estimates* (5.16) *and* (5.30) *provided by LSTD(0) and LSTD with expected eligibility traces with parameter λ, respectively, are identical for any $\lambda \in [0, 1)$.* ▲

Proof: Write $\mathcal{E}$ as $\mathcal{E} = \tilde{\Phi}\Psi_\lambda^\top$, where $\tilde{\Phi}$ is defined in (5.15). Then, since $\mathcal{E}^\top = \Psi_\lambda \tilde{\Phi}^\top$, it follows that

$$\begin{aligned} \hat{\beta}_k &= (\mathcal{E}^\top \tilde{\Phi} - \alpha \mathcal{E}^\top \tilde{\Phi}')^{-1} \mathcal{E}^\top R \\ &= (\Psi_\lambda \tilde{\Phi}^\top \tilde{\Phi} - \alpha \Psi_\lambda \tilde{\Phi}^\top \tilde{\Phi}')^{-1} \Psi_\lambda \tilde{\Phi}^\top R \\ &= (\tilde{\Phi}^\top \tilde{\Phi} - \alpha \tilde{\Phi}^\top \tilde{\Phi}')^{-1} \Psi_\lambda^{-1} \Psi_\lambda \tilde{\Phi}^\top R \\ &= (\tilde{\Phi}^\top \tilde{\Phi} - \alpha \tilde{\Phi}^\top \tilde{\Phi}')^{-1} \tilde{\Phi}^\top R \end{aligned} \tag{5.32}$$

where the latter equation is equivalent to (5.16) of LSTD(0).

■

It should be noted that the result in Theorem 5.1 is independent of the λ-term. Therefore, if expected eligibility traces are used in the LSTD setting, the feature of using λ to move from one-step TD to MC (and anything in between) is no longer present.

Remark 5.1. *Theorem 5.1 follows from two key facts. First, the fact that the expected eligibility trace can be written as an invertible linear transformation, represented by matrix Ψ_λ, of the feature vector, as shown in Proposition 5.1. Second, the fact that LSTD(0) is invariant for this linear transformation of the feature vectors. Actually, LSTD(0) is invariant for 'any' invertible linear feature transformation, as such a transformation does not affect the linear space spanned by the basis functions.*

5.3.3 LSET(λ)

In this section, we propose the method LSET(λ) to implement expected eligibility traces using least-squares techniques using empirical means for the on-policy setting. As proposed in [121], a simple way to compute the expected eligibility trace is via the empirical mean of past data samples. The empirical mean of the eligibility traces can be calculated as

$$\hat{e}(x_k) = \frac{1}{n_k(x)} \sum_{i=1}^{n_k(x)} z_{k_i}^x \tag{5.33}$$

where $n_k(x)$ is the number of times state x_k has been visited up to time $k \in \{0, ..., K\}$, $z_{k_i}^x$ is the trace of state x_k at time k_i, and k_i are the times at which state x_k has been visited.

Proposition 5.2. *For the tabular feature setting,* (5.33) *can alternatively be written as*

$$\hat{e}(x_k) = \hat{\Psi}_k \phi(x_k) \tag{5.34}$$

where

$$\lim_{K \to \infty} \hat{\Psi}_K = \Psi_\lambda \tag{5.35}$$

and where Ψ_λ is as defined in Proposition 5.1. ▲

Proof: It can easily be verified that for each state its empirical mean of the eligibility trace can be stored in each column of $\hat{\Psi}_k \in \mathbb{R}^{n \times p}$ such that $\hat{e}(x_k) = \hat{\Psi}_k \phi(x_k)$, where $\hat{e}(x_k)$ is equivalent to (5.33). As more samples are acquired, this empirical mean converges to the expected value (theoretical mean), and thus to the expected eligibility trace as in Proposition 5.1.

■

For approximate feature vectors, we aim to find a similar matrix $\hat{\Psi}_{\Phi,k}$ to approximate the expected eligibility trace:

$$\hat{e}(x_k) \approx \hat{\Psi}_k \phi(x_k). \tag{5.36}$$

To this effect we start by noticing (5.34) corresponding to (5.33) can be obtained from the following problem:

$$\hat{\Psi}_{\Phi,k} = \arg\min_{\Psi} \sum_{i=0}^{k} \|z_k - \Psi\phi(x_k)\|^2, \tag{5.37}$$

which is equivalent to

$$\hat{\Psi}_{\Phi,k} = \arg\min_{\Psi} \|\mathcal{Z}^\top - \Psi\tilde{\Phi}^\top\|^2. \tag{5.38}$$

However, we can still apply (5.38) for the approximate case. The closed-form solution of (5.38) is given by

$$\hat{\Psi}_{\Phi,k} = \mathcal{Z}^\top\tilde{\Phi}\left(\tilde{\Phi}^\top\tilde{\Phi}\right)^{-1}. \tag{5.39}$$

Subsequently, the parameters $\hat{\beta}_k$ at time instance k can be computed as

$$\hat{\beta}_k = (\hat{\mathcal{E}}^\top\tilde{\Phi} - \alpha\hat{\mathcal{E}}^\top\tilde{\Phi}')^{-1}\hat{\mathcal{E}}^\top R \tag{5.40}$$

where

$$\begin{aligned}\hat{\mathcal{E}} &= [\hat{e}(x_0), ..., \hat{e}(x_k)]^\top \\ &= [\hat{\Psi}_{\Phi,0}\phi(x_0), ..., \hat{\Psi}_{\Phi,k}\phi(x_k)]^\top.\end{aligned} \tag{5.41}$$

As typically proposed in the LSTD setting, also here we can use the Sherman-Morrison formula to reduce the computational complexity of LSET(λ) from $\mathcal{O}(d^3)$ to $\mathcal{O}(d^2)$. The recursive version of LSET(λ) is given by Algorithm 5, where ϵ_1 and ϵ_2 are small values to ensure that A_k and C_k are always invertible. In Algorithm 5, A_k keeps track of $\tilde{\Phi}^\top\tilde{\Phi}$, while B_k computes $\mathcal{Z}^\top\tilde{\Phi}$.

Algorithm 5 Recursive LSET(λ)

Require: $A_{-1} = \epsilon_1 I$, $B_{-1} = 0$, $C_{-1} = \epsilon_2 I$, $d_{-1} = 0$, $z_{-1} = 0$

1: $z_k = \alpha\lambda z_{k-1} + \phi(x_k)$
2: $A_k^{-1} = A_{k-1}^{-1} - \frac{A_{k-1}^{-1}\phi(x_k)\phi(x_k)^\top A_{k-1}^{-1}}{1+\phi(x_k)^\top A_{k-1}^{-1}\phi(x_k)}$
3: $B_k = B_{k-1} + z_k\phi(x_k)^\top$
4: $\hat{e}(x_k) = B_k A_k^{-1}\phi(x_k)$
5: $C_k^{-1} = C_k^{-1} - \frac{C_k^{-1}\hat{e}(x_k)\left(\phi(x_k)-\alpha\phi(x_k')\right)^\top C_k^{-1}}{1+\left(\phi(x_k)-\alpha\phi(x_k')\right)^\top C_k^{-1}\hat{e}(x_k)}$
6: $d_k = d_{k-1} + \hat{e}(x_k)r(x_k, x_k')$
7: $\hat{\beta}_k = C_k^{-1}d_k$

5.4 LSTD with mixed eligibility traces

LSTD with expected eligibility traces as defined in Section 5.3 fully relies on the expected eligibility traces. Another option is to use mixed eligibility traces, to balance between expected eligibility traces and ordinary eligibility traces [121]. A mixed eligibility trace is defined as

$$y_k = (1-\eta)e(x_k) + \eta(\alpha\lambda y_{k-1} + \phi(x_k)) \tag{5.42}$$

where η is a tuning parameter. If $\eta = 0$, the trace is equivalent to the expected eligibility trace as in (5.18). When $\eta = 1$, the ordinary eligibility trace is obtained as in (5.10). Any value between 0 and 1 balances between the two extremes. In this section, we investigate what happens if mixed traces are used in an LSTD setting. The main result of LSTD with the theoretical mixed traces, where the equivalence with LSTD(λ) is shown, is given in Section 5.4.1. In Section 5.4.2, the method LSET(η,λ) based on empirical means is proposed.

5.4.1 Main result for mixed traces

As shown by [121], (5.42) can be rewritten as

$$\begin{aligned}
y_k &= (1-\eta)e(x_k) + \eta(\alpha\lambda y_{k-1} + \phi(x_k)) \\
&= (1-\eta)\left(e(x_k) + \eta\alpha\lambda e(x_{k-1}) + (\eta\alpha\lambda)^2 e(x_{k-2}) + ...\right) \\
&\quad + \eta\left(\phi(x_k) + \eta\alpha\lambda\phi(x_{k-1}) + (\eta\alpha\lambda)^2\phi(x_{k-2}) + ...\right) \\
&= (1-\eta)\sum_{i=0}^{k}(\eta\alpha\lambda)^{k-i}e(x_i) + \eta\sum_{i=0}^{k}(\eta\alpha\lambda)^{k-i}\phi(x_i) \\
&= \sum_{i=0}^{k}(\eta\alpha\lambda)^{k-i}\left((1-\eta)e(x_i) + \eta\phi(x_i)\right)
\end{aligned} \tag{5.43}$$

where the latter is in similar form as (5.10). Using Proposition 5.1, (5.43) is equivalent to

$$\begin{aligned}
y_k &= ((1-\eta)\Psi_\lambda + \eta I)\sum_{i=0}^{k}(\eta\alpha\lambda)^{k-i}\phi(x_i) \\
&= \mathcal{M}h_k
\end{aligned} \tag{5.44}$$

where $\mathcal{M} := (1-\eta)\Psi_\lambda + \eta I$ and $h_k := \sum_{i=0}^{k}(\eta\alpha\lambda)^{k-i}\phi(x_i)$ is a trace discounted with $\eta\alpha\lambda$ (which differs from z_k in (5.10), which is discounted with $\alpha\lambda$). Based on $\mathcal{M}$ and data set $\mathcal{X}$, the parameters $\hat{\beta}_k$ can be computed in the LSTD-setting as

$$\hat{\beta}_k = (\mathcal{Y}^\top\tilde{\Phi} - \alpha\mathcal{Y}^\top\tilde{\Phi}')^{-1}\mathcal{Y}^\top R \tag{5.45}$$

where

$$\begin{aligned}\mathcal{Y} &= [y_0, ..., y_k]^\top \\ &= [\mathcal{M}h_0, ..., \mathcal{M}h_k]^\top \\ &= \mathcal{H}\mathcal{M}^\top\end{aligned} \tag{5.46}$$

and $\mathcal{H} := [h_0, ..., h_k]^\top$.

Theorem 5.2. *The value parameter estimates* (5.45) *and* (5.14) *provided by LSTD with mixed eligibility traces and LSTD(λ'), with $\lambda' := \eta\lambda$, are identical.* ▲

Proof: Since $\mathcal{Y}^\top = \mathcal{M}\mathcal{H}^\top$, (5.45) can be rewritten as

$$\begin{aligned}\hat{\beta}_k &= (\mathcal{Y}^\top\tilde{\Phi} - \alpha\mathcal{Y}^\top\tilde{\Phi}')^{-1}\mathcal{Y}^\top R \\ &= (\mathcal{M}\mathcal{H}^\top\tilde{\Phi} - \alpha\mathcal{M}\mathcal{H}^\top\tilde{\Phi}')^{-1}\mathcal{M}\mathcal{H}^\top R \\ &= (\mathcal{H}^\top\tilde{\Phi} - \alpha\mathcal{H}^\top\tilde{\Phi}')^{-1}\mathcal{M}^{-1}\mathcal{M}\mathcal{H}^\top R \\ &= (\mathcal{H}^\top\tilde{\Phi} - \alpha\mathcal{H}^\top\tilde{\Phi}')^{-1}\mathcal{H}^\top R.\end{aligned} \tag{5.47}$$

Since h_k is discounted with $\eta\alpha\lambda$, while z_k is discounted with $\alpha\lambda$, LSTD with mixed eligibility traces in (5.47) is equivalent to LSTD(λ') in (5.14) when $\lambda' := \eta\lambda$.

■

With the equivalence of LSTD(λ) (or LSTD(λ')) and LSTD with mixed eligibility traces shown, it gives us a new perspective on the original LSTD(λ). The first insight is that all properties of LSTD(λ) still hold when expected eligibility traces are applied since the methods are equivalent (e.g. convergence and smaller modulus of contraction as λ' increases). The second insight is the rationale behind the tuning of the λ'-parameter. In LSTD-setting, it turns out that policy evaluation based on one-step TD without any expected value ($\lambda = 0$, $\eta = 1$), is exactly equivalent to using all potential histories of trajectories ($\eta = 0$), independent of the eligibility decay factor λ. Figure 5.3 illustrates how λ' affects both λ and η, and which combinations are equivalent. Varying λ without considering expected eligibility traces ($\eta = 1$) is equivalent to varying η when a full MC method is considered ($\lambda = 1$).

An additional insight that expected eligibility traces provide in the LSTD setting, is related to the bias. Although it was stated in [121] that expected eligibility traces induce a bias, it does not induce an extra bias on top of the bias from bootstrapping in LSTD setting. Since implicitly all potential histories are considered in LSTD(0), it motivates why it gives the best convergence results in case of perfect features (and bias is thus not an issue) (see e.g. [24]). In the perfect feature setting, the best one can do is considering all the potential histories that could have occurred based on the data samples. However, this is shown to be equivalent to LSTD(0).

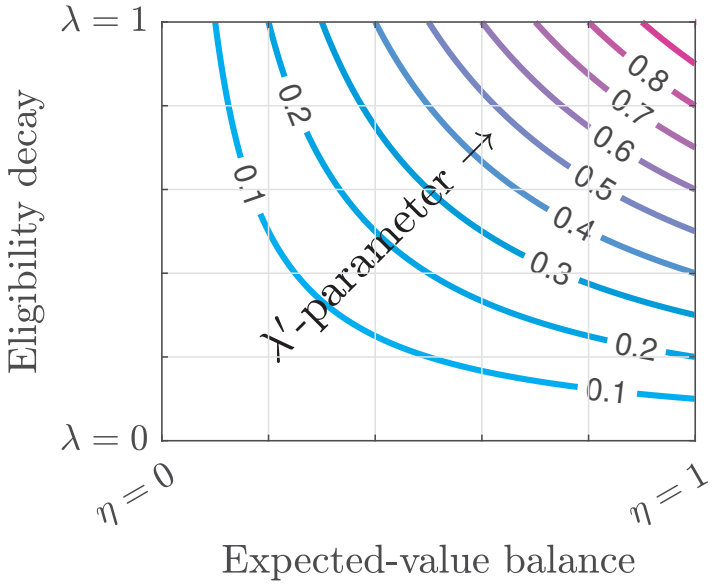


Figure 5.3. Visualization of λ' in LSTD(λ'), where $\lambda' = \eta\lambda$.

5.4.2 LSET(η,λ)

A recursive, on-policy, implementation of LSET(η,λ) can be pursued similar to LSET(λ). In fact, this would entail adding the following instruction between lines 4 and 5 in Algorithm 5:

$$\hat{y}_k = (1-\eta)\hat{e}(x_k) + \eta(\alpha\lambda\hat{y}_{k-1} + \phi(x_k)), \tag{5.48}$$

and by replacing $\hat{e}(x_k)$ by $\hat{y}_k$ in lines 5 and 6. Notice that LSET(0,λ) = LSET(λ), and LSET(1,λ) = LSTD(λ). A similar result to Proposition 5.2, stating that the empirical means converge to the theoretical ones can also be obtained, but is omitted for the sake of brevity.

5.5 Simulation results

In this section, simulation results are presented where the performances of LSET(λ) and LSET(η,λ) are analyzed, and the theoretical results are supported. In Section 5.5.1, the Open World example considered in [121] is used to compare the direct methods with their least-squares alternatives. The second example provided in Section 5.5.2 concerns the value function estimation of a cart-pole balancing task.

5.5.1 Open World

In the Open World example, the agent starts in the top-left corner of the grid and moves either to the right or downwards with equal probability. When reaching the bottom-right corner, the agent receives a reward of 1 with a probability of 0.2. The trace decay parameter λ is set to 0.9, while the discount factor α is set to 0.95. For the mixed traces methods ET(η,λ) and LSET(η,λ), the mix parameter η is set to 0.5. Furthermore, $\lambda' = \eta\lambda$ is used in TD(λ') and LSTD(λ'). In Figure 5.4, a visualization of the value function estimation of the direct methods (TD(0), TD(λ'), ET(λ), and ET(η,λ)), and their least-squares alternatives (LSTD(0), LSTD(λ'), LSET(λ), and LSET(η,λ)) is given. For the direct methods, all stepsizes are set to $\sqrt{1/i}$, where i is the index of the episode.

In TD(0), only the last state is updated when the first reward is received in episode 4. TD(λ') updates all the states of the path in the 4th episode. ET(λ) updates also the states of previous paths, although the impact of the update decreases for episodes that occurred earlier. A similar observation is shown for ET(η,λ), where, as expected, even more emphasis is placed on the rewarded trace. When receiving the first reward, LSTD(0) updates all the states that occurred in the data. LSTD(λ') updates the states in the episode where the reward is received in the same manner as TD(λ'), whereas states of previous episodes are also updated slightly. The updates of LSET(λ) seem initially to be more similar to the updates of LSTD(λ'), although after 100 episodes LSET(λ) is almost similar to LSTD(0). LSET(η,λ) on the other hand, visually mixes the ordinary eligibility traces of LSTD(λ') with the expected eligibility traces of LSET(λ).

These observations become even more clear when analyzing the RMS error of the value function estimate. The average RMS errors along with the standard deviations of the value function estimates over 100 Monte Carlo simulations are given in Figure 5.5. LSTD(0) has the best performance, and, after approximately 100 episodes, LSET(λ) has a similar average RMS error. The performance of LSET(η,λ) is comparable with the performance of LSTD(λ'). When LSTD with theoretical expected eligibility traces is implemented, the exact same results as LSTD(0) are obtained for any λ (which can be checked by sweeping over different values of λ). Similar holds for LSTD with theoretical mixed eligibility traces and LSTD(λ'). This confirms the findings in Theorem 5.1 and 5.2.

Figure 5.4. Visual comparison in the Open World example of direct methods (TD(0), TD(λ'), ET(λ), and ET(η,λ)), and their least-squares alternatives (LSTD(0), LSTD(λ'), LSET(λ), and LSET(η,λ)). Rewards are received in the right-bottom corner with a probability of 0.2, while the agent starts the episode always in the left-top corner. The first reward is received in episode 4, and the second reward in episode 8. LSTD(0) shows the best performance since it implicitly uses all the expected traces in the estimation approach.

5.5.2 Cart-pole balancing

In this example, the least-squares methods are compared by analyzing a balancing task for a linearized model of a cart-pole, described in [24, 27]. The input u is a force acting on the cart and is defined as $u_k = Kx_k$, where K is the feedback control law. The state x of the system consists of the position (s) and velocity ($\dot{s}$) of the cart, and the angle (θ) and angular velocity ($\dot{\theta}$) of the pole. By defining $M = 0.5$ [kg] as the mass of the cart, $m = 0.5$ [kg] and $l = 0.6$ [m] as the mass and length of the pole, respectively, $b = 0.1$ [N/(ms)] as the friction coefficient of the cart on the ground, and $g = 9.81$ [m/s^2], the linearized continuous-time state-space system is given by

$$\begin{bmatrix} \dot{s} \\ \ddot{s} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-4b}{4M+m} & \frac{3mg}{4M+m} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{6b}{4Ml+ml} & \frac{-6g(M+m)}{4Ml+ml} & 0 \end{bmatrix} \begin{bmatrix} s \\ \dot{s} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{4}{4M+m} \\ 0 \\ \frac{-6}{4Ml+ml} \end{bmatrix} u. \tag{5.49}$$

The system is discretized with a sample time of 0.1 [s].

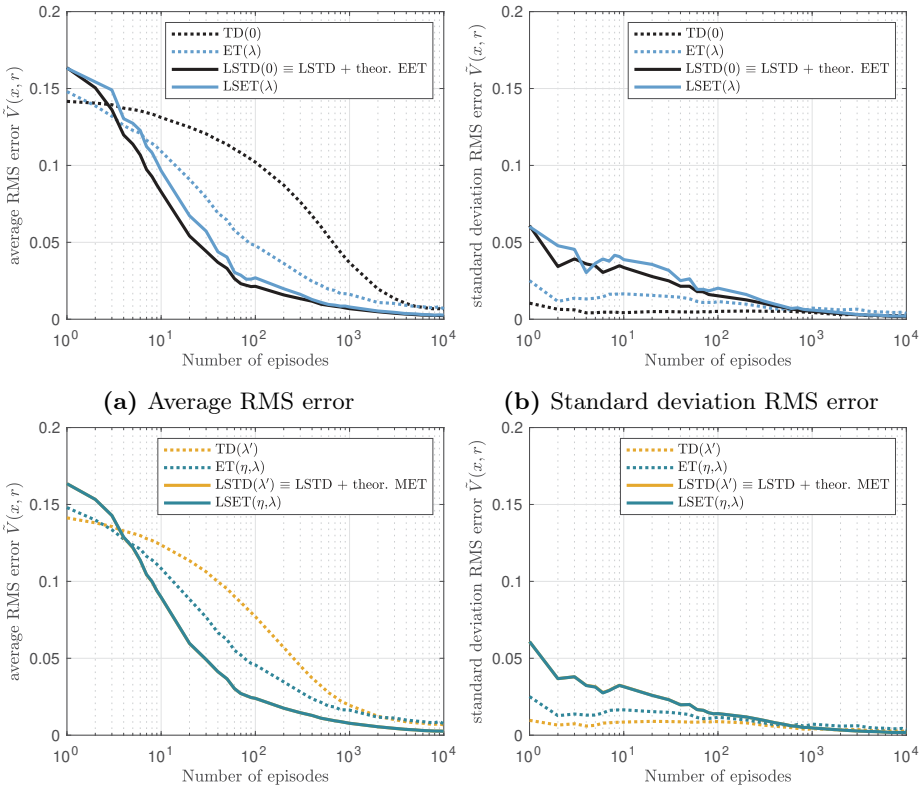
A discounted, quadratic value function is used, given as

$$V(x_k) = \sum_{i=k}^{\infty} -\alpha^i \left(x_i^\top Q x_i + u_i^\top R u_i \right) \tag{5.50}$$

with $Q = \mathrm{diag}([1, 0, 100, 0])$, $R = 0.1$, and $\alpha = 0.95$. The exact value function can be derived by solving the algebraic Riccati equation. Feedback control matrix K subsequently follows from the LQR solution. For the perfect feature setting, the ten possible products of individual states $(s^2, s\dot{s}, \dot{s}^2, \dots)$ plus a constant term are used.

The four methods that are compared are LSTD(0), LSTD(λ'), LSET(λ), and LSET(η,λ), where $\lambda = 0.9$, $\eta = 0.5$, and $\lambda' = \eta\lambda$. The average RMS error of the value function parameters ($\frac{1}{n_{\mathrm{mc}}} \sum_{i=1}^{n_{\mathrm{mc}}} \sqrt{\|\beta - \hat{\beta}_k\|^2}$) over $n_{\mathrm{mc}} = 100$ simulations is given in Figure 5.6, along with its standard deviation. Similar to the Open World example in Section 5.5.1, LSTD(0) and LSTD(λ') have the best convergence performance, followed by LSET(λ) and LSET(η,λ). After approximately 1000 data samples, all methods show a similar error as expected. In the limit, they all converge to the same RMS error due to the selected features, which can exactly describe the value function. Similar conclusions can be found in literature (e.g. [15] and [37]), where the value for λ barely affects the convergence in case of the perfect feature setting.

We turn now to the case where the features can only approximately and not exactly describe the value function. For this setting, we choose the feature vectors to be the following quadratic terms on the state $(s^2, \dot{s}^2, \theta^2, \dot{\theta}^2)$ plus a constant

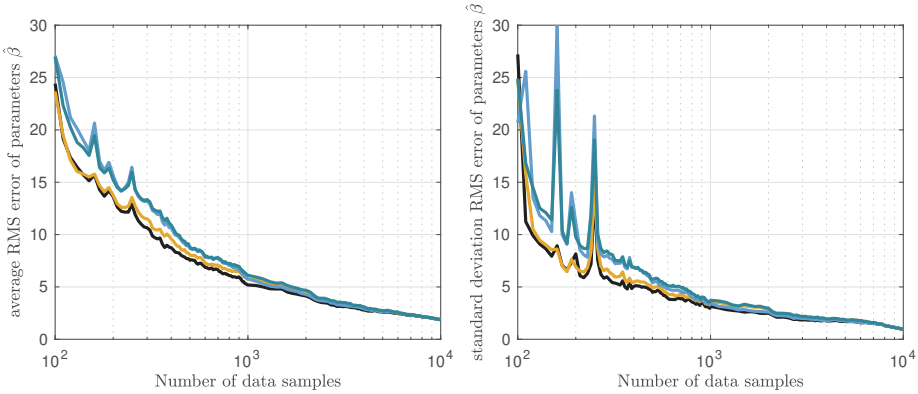


(a) Average RMS error **(b)** Standard deviation RMS error

(c) Average RMS error **(d)** Standard deviation RMS error

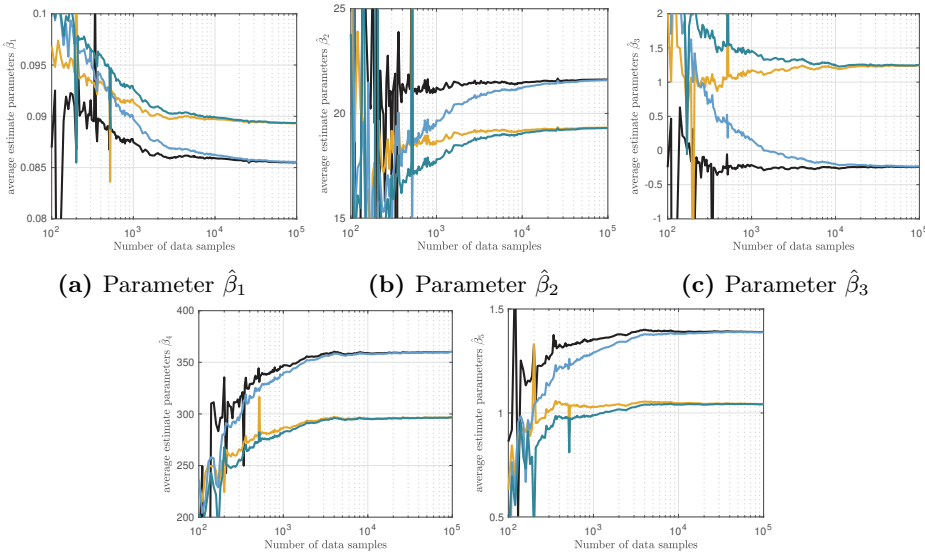
Figure 5.5. Average RMS error and its standard deviation of the value function in the Open World example. The results for LSTD with the theoretical expected eligibility traces (theor. EET) are exactly equivalent to each other (as expected from Theorem 5.1). There is a similar equivalence for LSTD with the theoretical mixed eligibility traces (theor. MET) are equivalent to LSTD(λ'), with $\lambda' = \eta\lambda$ (as expected from Theorem 5.2). The results are obtained from 100 Monte Carlo simulations.

term. Results on the parameter convergence are given in Figure 5.7. These results clearly indicate that in the limit, the parameters found with LSET(λ) converge to the same values as LSTD(0), and the parameters found with LSET(η,λ) converge to the same values as LSTD(λ'), with $\lambda' = \eta\lambda$. They do not converge to the same values as the methods have different biases. This follows from the methods having different modulus of contraction, see [11, Section 6.3.6]. Therefore, also this approximate case supports the claims of Proposition 5.2 that is only formally stated for the tabular case.



(a) Average RMS error **(b)** Standard deviation RMS error

Figure 5.6. Average RMS error and its standard deviation of the value function parameters over 100 Monte Carlo simulations in the cart-pole example with perfect features. LSTD(0) (—) and LSTD(λ') (—) outperform LSET(λ) (—) and LSET(η,λ) (—).



(a) Parameter $\hat{\beta}_1$ **(b)** Parameter $\hat{\beta}_2$ **(c)** Parameter $\hat{\beta}_3$

(d) Parameter $\hat{\beta}_4$ **(e)** Parameter $\hat{\beta}_5$

Figure 5.7. Average value function parameters $\hat{\beta}$ over 100 Monte Carlo simulations in the cart-pole example with imperfect features. The parameters of LSET(λ) (—) converges to the same values as LSTD(0) (—). Furthermore, the parameters of LSET(η,λ) (—) converges to the same values as LSTD(λ') (—).

5.6 Conclusions

This chapter analyses the use of expected eligibility traces and mixed eligibility traces in the LSTD setting. It is shown that when theoretical expected eligibility traces with parameter λ are considered in the LSTD setting, the method boils down to LSTD(0), irrespective of λ. Moreover, using a combination of the theoretical expected eligibility traces and ordinary eligibility traces in the form of mixed eligibility traces with balance parameter η is equivalent to LSTD($\eta\lambda$). This chapter gives therefore insight into how the parameter $\lambda' = \eta\lambda$ in LSTD(λ') balances between expected eligibility traces, ordinary eligibility traces, one-step TD and MC. Another important insight of the equivalences between methods shown in this chapter is that expected eligibility traces can be seen as a way to leverage the data efficiency of LSTD for nonlinear feature settings where LSTD is not applicable. Furthermore, the methods LSET(λ) and LSET(η,λ) are presented, using the empirical mean of the eligibility traces to obtain the expected value. Their applicability and performance are assessed based on two examples; Open World and the cart-pole balancing task. In these examples, LSTD(0) is found to be superior to LSET(λ), and LSTD(λ) is found to be superior to LSET(η,λ). This can be interpreted in the light of the equivalences between methods shown in this chapter.

IV

State Estimation for Output-Feedback Systems

Chapter 6

A Bayesian Approach to Pose Estimation

Abstract - Vision-based estimation algorithms play a critical role in automating industrial processes. In this chapter we propose and compare two vision-based algorithms to automate the in-feed of manufactured objects to a Coordinate Measuring Machine (CMM). The first algorithm is based on the Bayes' filter and Gaussian process regression. We propose to use similarity scores using pre-rendered views of the object in a template matching approach to estimate the likelihood function. This template matching is carried out efficiently through Bayesian optimization. The second algorithm relies on deep learning to directly convert camera images to parameterized likelihood functions. The algorithms are validated in both a simulated and an industrial environment, demonstrating their capability of providing reliable and consistent pose estimates.

This chapter is based on: S.J.A. van Esch, R.A.C. van Zuijlen and D.J. Antunes, "A Bayesian approach to pose estimation using Gaussian processes and deep learning", *(submitted)*.

6.1 Introduction

In recent years, the transition to Industry 4.0 has garnered significant attention, with companies automating processes to reduce cycle times, lower costs, and eliminate human error, resulting in repeatable quality. As part of this shift, there is a growing emphasis on advancing technological capabilities and reducing reliance on external manufacturing hubs, making it essential to prioritize the development of Industry 4.0 technologies [62]. In this context, advancements in better hardware and sensors, particularly vision-based approaches, offer a low-cost, universal, and practical means of automating industrial processes.

A task that has garnered significant attention in various domains is the automatic handling of objects, which requires estimation of their poses. Beyond industrial applications, pose estimation is a critical challenge, for example, in autonomous driving [91], robotic grasping [126, 131, 133], and human-robot interaction [34]. An essential step in pose estimation is acquiring data, where common options include normal images and line scans. An alternative representation of an object's data is through 3D reconstruction, created from multiple viewpoints. Unless an advanced 3D camera is utilized, the quality of the reconstruction is often compromised due to inaccuracies introduced when converting 2D images into its 3D reconstruction, thereby limiting the performance in accuracy.

Pose estimation techniques can be divided from classic heuristic approaches to advanced deep learning approaches. Classic approaches often rely on geometric principles and template matching [45], which quantifies the similarity between templates and query images based on pixel differences. While effective in controlled environments, this method is sensitive to variations in rotation, scaling, and lighting, making it less reliable in dynamic settings. Besides its environmental sensitivity, template matching can become computationally expensive, especially with large image-to-template ratios and numerous evaluations. To reduce computation time, resolution pyramids can be used [118]. Another classical approach is the Iterative Closest Point (ICP) [13], an iterative algorithm that minimizes the distance between a point cloud and a reference, typically a CAD model.

More recent research is increasingly adopting machine learning techniques, with Convolutional Neural Networks (CNNs) emerging as a dominant architecture in image processing. CNNs are widely used across a range of applications, such as image segmentation [96], object detection [93, 94], and pose estimation. In the domain of object-level pose estimation, networks such as PVNet [86], DenseFusion [123], and PoseCNN [127] have been proposed to predict the full 6DOF pose of objects using RGB or RGB-D data. These approaches demonstrate the growing potential of deep learning in pose estimation in complex and variable environments. While these methods cater for generality, their performance on their dataset, let alone in real-world environments, often falls short of the precision needed for

high-accuracy applications. State-of-the-art deep learning methods often yield pose estimation errors in the centimeter range [123]. Furthermore, their accuracy degrades significantly due to different backgrounds, variations in lighting, or object texture [86, 127]. These limitations highlight the challenge of relying solely on neural networks for high-precision tasks and motivate the development of model-based or hybrid approaches.

By repositioning objects and acquiring multiple measurements, the precision of pose estimation can be improved. To integrate a series of movements and measurements, well-established filtering techniques, such as the Bayes' filter [33], and its special case, the Kalman filter [56], are commonly used. In general, the Bayes' filter places no constraints on the probability density function, making it versatile for handling non-linear and non-Gaussian systems. The Kalman filter (and its variants) simplifies computations by assuming Gaussian distributions, and is limited to unimodal distributions. To capture multimodality, mixture model based techniques can be pursued, such as the Gaussian Sum Filter (GSF) [1] relying on a Gaussian Mixture Model (GMM). All filters follow a two-step process: the prediction step, which propagates the system's dynamics and accounts for the disturbances, and the correction step, which updates the state estimate based on new sensor data.

This chapter proposes a general model-based framework for estimating an object's orientation using data from a 2D camera. The framework integrates a probabilistic filter, which serves as its core. A first algorithm is proposed based on the Bayes' filter and Gaussian process regression [92]. We propose to use similarity scores using pre-rendered views of the object in a template matching approach to obtain a likelihood function. This template matching is carried out efficiently through Bayesian optimization. A second algorithm relies on deep learning to directly convert camera images to parameterized likelihood functions.

Beyond proposing the framework, this chapter provides a practical evaluation of its effectiveness in accurately pose estimation in a real-world factory environment. As a motivating example, the chapter focuses on automating the supply of objects to a CMM. Since the objects arrive in an upright position, the problem reduces to a 1DOF yaw estimation task. This use case is illustrated by the workcell in Figure 6.1, which includes a robotic manipulator with a 2D camera and a CMM. The CMM offers precise measurements for accurate quality control, while the manipulator handles object movement. Accurate yaw estimation is critical, as the CMM requires objects to be positioned in a specific pose to align with its inspection program. The nature of the objects in this objective can be described, but is not limited to, cylindrical, metallic, and textureless objects manufactured by Computer Numerical Control (CNC) machines.

The contributions in this chapter are therefore threefold:

1) An active likelihood estimation approach that reduces computational cost by

using Bayesian optimization to efficiently approximate a similarity function, which can be converted into a likelihood function for Bayesian filtering.

2) A deep learning architecture using two neural networks: an auto-encoder to map the query image to a latent space, and a second network to predict a parameterized likelihood function, enabling efficient filtering.

3) An application of the developed methods to automate the supply of objects to a CMM.

The structure of this chapter is as follows. Section 6.2 outlines the problem and objective, and preliminaries are provided regarding the Bayes' filter. In Section 6.3, an active likelihood estimation method via Bayesian optimization is introduced. A deep learning method where an image is converted to a parameterized likelihood function, suitable for the Bayes' filter, is given in Section 6.4. In Section 6.5 the results in a simulated environment are given, and Section 6.6 showcases the experimental results for both proposed methods. In Section 6.7, the chapter concludes and provides recommendations for future research.

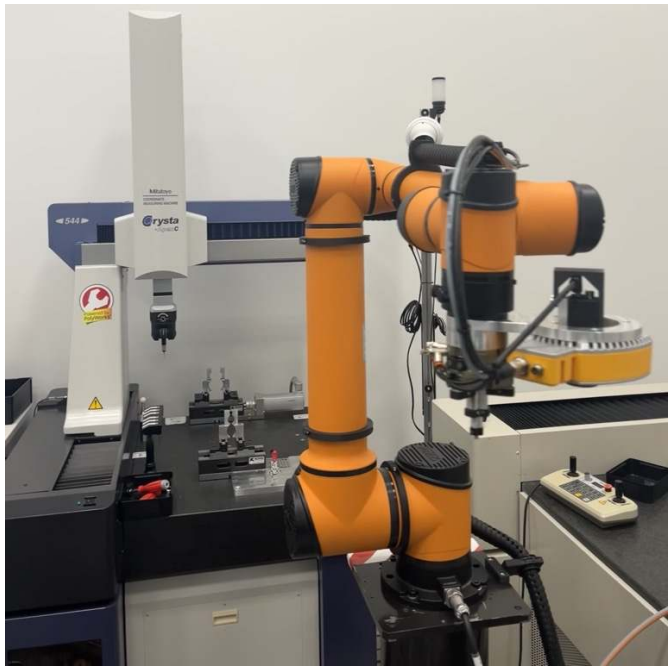


Figure 6.1. Representative view of the industrial workcell showing the robotic manipulator and CMM.

6.2 Problem formulation and preliminaries

In this chapter, we aim to retrieve the pose of an object, denoted by state $x_k \in \mathbb{R}^{n_x}$ at time k from images/measurements, denoted by $y_k \in \mathbb{R}^{n_y}$. The objects considered may can contain ambiguities or non-informative views, may have distinctive features but also repeated patterns that introduce ambiguity, or may contain unique features across the full yaw range. For generalization across various objects, access is given to their CAD model. The method should be fast in terms of computation time, while bridging the gap between real images and CAD models.

We take the standard notation from the literature to model our problem [11]. The state evolves according to $x_{k+1} = f(x_k, u_k, w_k)$, where $u_k \in \mathbb{R}^{n_u}$ are known manipulator actions. The images/measurements follow from the output function $y_k = g(x_k, v_k)$. In both the state evolution and output function, noise terms are present, denoted by $w_k \in \mathbb{R}^{n_x}$ and $v_k \in \mathbb{R}^{n_y}$, respectively. The noise term w_k introduces uncertainty into the system during the execution of the robotic manipulator's actions. The observation noise v_k accounts for pixel-level distortions or external disturbances on the image, such as those caused by lighting variations, visual artifacts, or environmental reflections.

A fundamental component of the proposed framework for probabilistic state estimation is the Bayes' filter. The goal of the Bayes' filter is to obtain a belief on the state, which is the probability distribution over the state affected by uncertainty. This belief state is denoted by

$$bel(x_k) = p(x_k|y_0, ..., y_k, u_0, ..., u_k) \tag{6.1}$$

and conditions the distribution on all available measurements and control inputs.

The Bayes' filter can be written as a two-step recursive estimation method, consisting of a prediction and an update step. In the prediction step, the system dynamics $p(x_{k+1}|x_k, u_k)$ are used to estimate the probability of the next state x_{k+1}, based on the previous state x_k and the applied control input u_k. The prediction integrates over all possible previous states, weighted by the belief, yielding the predicted belief:

$$\overline{bel}(x_{k+1}) = \int p(x_{k+1}|x_k, u_k)\, bel(x_k) dx_k. \tag{6.2}$$

Upon a new observation y_k, the update step incorporates the measurement to refine the predicted belief. It adjusts the belief by weighting each state x_k according to how likely it is to produce the observed measurement. This likelihood is captured by the sensor model $p(y_k|x_k)$. The updated belief is obtained by applying Bayes' rule, which combines the likelihood of the measurement and the prior (predicted) belief as

$$bel(x_k) = \eta\ p(y_k|x_k)\overline{bel}(x_k), \tag{6.3}$$

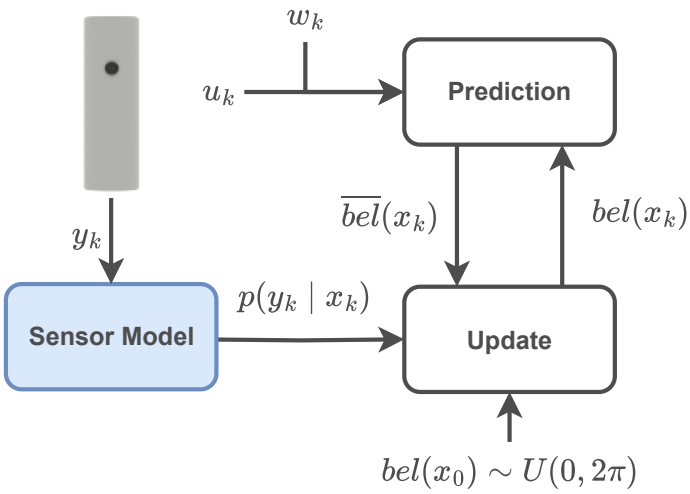


Figure 6.2. The Bayesian filtering diagram consisting of the prediction and update steps. It includes the sensor model, with the observation as input. In this framework, the sensor model serves as an interchangeable component.

where η is a normalization factor to ensure that the posterior belief remains a valid probability distribution. The filtering steps are illustrated in Figure 6.2.

In the remainder of this chapter, the focus will be on proposing models and strategies to compute the likelihood function $p(y_k|x_k)$, while pursuing efficient representations of the probability density functions in the Bayes' filter steps.

6.3 Active likelihood estimation via Bayesian optimization

An effective approach to obtain the likelihood function $p(y_k|x_k)$ is by pursuing template matching. In template matching the output y_k is compared with templates T_j of which the underlying state x_k is known. The highest probabilities are then assigned to the states of which the templates show highest similarities with the output. Comparing y_k with all templates to obtain an accurate estimate of x_k however is very time consuming and computationally expensive. It requires comparing an image of an object's pose with images of all possible objects' poses (digitally generated, e.g. via Blender [http://www.blender.org]). In order to obtain a high accuracy (required in industrial environments) the amount of templates is typically very large, leading to unfeasible computation time for real-time applications or

severely limiting the throughput.

In this section, a method is developed to estimate a valid likelihood function $p(y_k|x_k)$, building upon Gaussian process regression (GPR) and Bayesian optimization. In the first part, a procedure is described to build a likelihood function based on templates using a similarity function. In the second part, GPR and Bayesian optimization are introduced to effectively and accurately estimate the likelihood function, which reduces the computational time significantly. This method can be used to pursue the Bayes' filter with a discretized state space. Furthermore, it can be used for training data generation for the deep learning method proposed in Section 6.4.

6.3.1 Similarity function

In this section, a likelihood function is computed via similarity scores between the measurement and a set of templates. Let the set of N templates be defined as $\mathcal{T} := \{T_1, T_2, \ldots, T_N\}$, generated according to $T_j = \bar{g}(x_j)$, where x_j is a known orientation, $T_j \in \mathbb{R}^{n_y}$. $\bar{g}(\cdot)$ is the generator for the templates. Based on this set of templates, one way for computing a similarity score is via the normalized cross-correlation (NCC) coefficient. The NCC coefficient is denoted by

$$s(y, x) = \frac{\langle T'(x), y' \rangle}{\|T'(x)\| \cdot \|y'\|}. \tag{6.4}$$

Here, $T'(x) = \bar{g}(x) - \bar{T}$ and $y' = y - \bar{y}$ denote mean-centered images, where the means are by $\bar{T} = \frac{1}{|D|} \sum_{x \in D} \bar{g}(x)$ and $\bar{y} = \frac{1}{|D|} \sum_{x \in D} y(x)$ and D is the $w \times h$ spatial support of the template, and y is the measured image. The inner product $\langle \cdot, \cdot \rangle$ computes the measure of similarity between the two images, whereas the magnitude $\| \cdot \|$ provides the normalization.

To facilitate the construction of a Gaussian likelihood model, a distance function is defined as

$$d(y, x) = 1 - |s(y, x)|, \tag{6.5}$$

where a perfect match yields $d(y, x) = 0$. The absolute value accounts for both strong positive and negative similarities, treating them equally as good matches. Low similarities on the other hand results in higher distances.

These distances are converted into likelihoods by applying:

$$p(y|x) = \frac{1}{Z} \exp\left(-\frac{d(y, x)^2}{2\sigma^2}\right) \tag{6.6}$$

where

$$Z = \sum_{j=1}^{N} \exp\left(-\frac{d(y, x_j)^2}{2\sigma^2}\right). \tag{6.7}$$

Here, σ controls the uncertainty in the measurement noise. Smaller values yield sharper distributions, while larger values allow for greater uncertainty and smoother likelihood profiles.

This approach yields a flexible likelihood over the discretized states, capable of adopting any shape, including multimodal functions. Unlike Kalman filters, which assume unimodal Gaussian distributions, the Bayes' filter can represent and propagate such multimodality, making it more appropriate in this context of highly non-linear observations. In particular, if the distance function is small for distinct templates, the obtained probability distribution (6.6) will be multimodal.

To evaluate observation y_k, one approach is to perform an exhaustive matching procedure over all templates $T_j \in \mathcal{T}$. As mentioned above, a major drawback of this is its computational burden. Evaluating the observation y_k against every template T_j becomes very time-consuming when the state grid is finely discretized, which is required to obtain the desired accuracy. Not all template evaluations are necessary, as neighboring templates typically yield similar scores. By strategically selecting a subset of templates, the number of evaluations can be reduced. However, a likelihood value is still required for each state in the Bayes' filter. This can be achieved by applying Bayesian optimization techniques to estimate the similarity function efficiently, as discussed in the next section.

6.3.2 Similarity estimation using Bayesian optimization

GPR is commonly employed to approximate unknown functions in the presence of noisy observations. Gaussian processes (GPs) are a class of probabilistic models that assume any finite set of function samples that follow a multivariate Gaussian distribution [92]. Via Bayesian optimization, relevant function samples are selected to obtain an accurate approximation of the underlying, true function [35].

We fix y and define a function of x given y, denoted by $L_y(x) := s(y, x)$. In this context, the objective is to estimate $L_y(x)$, denoted by $\hat{L}_y(x)$, where the measurements are influenced by disturbances. An estimate of $L_y(x)$ can be fully characterized by the input set $X = \{\bar{x}_0, ..., \bar{x}_n\}$ and the corresponding observations $Y = \{L_y(\bar{x}_0), ..., L_y(\bar{x}_n)\}$. This estimate also depends on prior beliefs about the mean function $\mu(\cdot)$ and the covariance function $c(\cdot, \cdot)$. Given the dataset $\mathcal{D} := \{X, Y\}$ of function evaluations and corresponding observations, the joint distribution can be written as

$$\begin{bmatrix} Y \\ \hat{L}_y(x) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(X) \\ \mu(x) \end{bmatrix}, \begin{bmatrix} k(X,X) + \sigma^2 I_n & k(x,X) \\ k(X,x) & k(x,x) \end{bmatrix} \right) \tag{6.8}$$

where $k(\cdot, \cdot)$ is a kernel function, σ^2 is the variance of observation noise, and I_n is the identity matrix of size n, corresponding to the number of training points. GPR is performed by conditioning the joint distribution in (6.8) on observed data

$\mathcal{D}$ using Bayes' theorem. The conditioned posterior distribution remains Gaussian, with the mean and covariance estimates given by

$$\begin{aligned}\hat{L}_y(x) &:= \mathbb{E}\left[L_y(x) \mid \mathcal{D}\right] \\ &= \mu(x) + k(x, X)W(x, X)\left(Y - \mu(X)\right)\end{aligned} \tag{6.9}$$

$$\begin{aligned}\hat{c}(x, x') &:= \operatorname{cov}\left(L_y(x), L_y(x') \mid \mathcal{D}\right) \\ &= k(x, x') - W(x, X)k(X, x'),\end{aligned} \tag{6.10}$$

where $W(x, X) = k(x, X)\left[k(X, X) + \sigma^2 I_n\right]^{-1}$.

Let $\mathcal{S}_k := \{S_1, ..., S_n\} \subseteq \mathcal{T}$ be the subset of templates used to estimate $\hat{L}_y(x)$. Within the field of Bayesian optimization, there are several acquisition functions for selecting the templates for $\mathcal{S}_k$ to estimate $L_y(x)$ accurately. Here, we will use a variant of the Upper Confidence Bound (UCB), based on the mean, standard deviation, and belief. This prioritizes templates that reduce uncertainty, are likely to have high similarity, and actively steers the sampling technique to use more templates in regions with a high belief, discarding irrelevant regions as a result of previous filtering steps. The acquisition function is given by

$$\alpha(x) = \lambda_1 \hat{L}_y(x) + \lambda_2 \hat{\sigma}(x) + \lambda_3 bel(x), \tag{6.11}$$

where $\lambda_i \in [0, 1]$ with $\sum_{i=1}^{3} \lambda_i = 1$ define the weight coefficient for the exploration-exploitation trade-off with the belief state. By reducing the number of template evaluations, it reduces the computational cost from $|\mathcal{T}|$ to $|\mathcal{S}_k|$.

In summary, the proposed active likelihood estimation approach is to run the Bayes' filter (6.2), (6.3), where (i) in order to compute the update step, the likelihood is parameterized as a function of $s(y, x)$ as in (6.6); and (ii) given a measurement y, $L_y(x)$ is estimated via Bayesian optimization, where the choice of samples to estimate $L_y(x)$ is optimized instead of using an exhaustive sweep over x.

6.4 Deep learning Bayes' filter

The aforementioned proposed method predominantly relies on image-to-template evaluations, which, despite advanced sampling techniques, still require a certain number of template evaluations. Moreover, these approaches are inherently sensitive to environmental variations, such as lighting or background changes. These limitations motivate the transition toward a deep learning approach that retains the probabilistic filtering structure while addressing these challenges.

To integrate such a learning-based method, adopting a compact and expressive representation of the multimodal likelihood distributions, produced by the sensor model, is crucial. Mixture models offer a natural solution to this challenge.

6.4.1 Filtering with mixture models

Let the likelihood function in (6.3) be approximated by a general mixture model defined by

$$\hat{p}_{mix}(y \mid x) = \sum_{i=1}^{K} \pi_i \; P(x \mid \theta_i), \quad \text{where } \sum_{i=1}^{K} \pi_i = 1. \tag{6.12}$$

Here, π_i denotes the weight associated with the i^{th} component. $P(x \mid \theta_i)$ represents the probability density function on the interval x and parametrized by θ, and K is the total number of mixture components.

In the remainder of the chapter, we will use the von Mises distribution as mixture component, although the framework generalizes to other distributions, such as, e.g., Gaussian distributions. The von Mises distribution is chosen since it handles a circular state space, and particularly useful for the problem of estimating the yaw. The von Mises distribution is defined by

$$p_{vm}(x \mid \mu, \kappa) = \frac{1}{2\pi I_0(\kappa)} \exp\left(\kappa \cos(x - \mu)\right), \tag{6.13}$$

where μ is the mean, $\kappa \geq 0$ is the concentration parameter (analogous to σ^{-2} in the Gaussian distribution), and $I_0(\kappa)$ is the modified Bessel function, which serves as a normalization factor. The general Bessel function is given by

$$I_n(\kappa) = \frac{1}{2\pi} \int_0^{2\pi} \exp\left(\kappa \cos x\right) \cos(nx)\, dx. \tag{6.14}$$

To integrate mixtures of von Mises distributions into the filtering framework, the steps in (6.2) and (6.3) are expressed in terms of von Mises (see e.g. [59, 60, 74]). Specifically, the prediction step in (6.2) involves the convolution of two probability density functions, denoted by $*$, such that

$$p_{vm}^{(1*2)}(x \mid \mu_{12}, \kappa_{12}) = p_{vm}^{(1)}(x \mid \mu_1, \kappa_1) \; * \; p_{vm}^{(2)}(x \mid \mu_2, \kappa_2). \tag{6.15}$$

Although this convolution does not produce a von Mises distribution, [74] and [73] demonstrate that $p_{vm}^{(1*2)}(x|\mu_{12}, \kappa_{12})$ can be well approximated by the following von Mises

$$p_{vm}^{(1*2)} \approx p_{vm}\left(x \mid \mu_1 + \mu_2,\; A^{-1}\left(A(\kappa_1)A(\kappa_2)\right)\right), \tag{6.16}$$

where $A(\kappa) = \frac{I_1(\kappa)}{I_0(\kappa)}$ is the ratio of modified Bessel functions, and $A^{-1}(\cdot)$ denotes the inverse of this function.

For the update step in (6.3), its von Mises equivalent consists of a product of von Mises distributions. As shown in [80], this operation results in an (unnormalized) von Mises distribution, and is given by

$$p_{vm}^{(1 \cdot 2)} = Z_{12} p_{vm}(x \mid \mu_{12}, \kappa_{12}). \tag{6.17}$$

Here, μ_{12}, κ_{12} are the updated mean and concentration parameters and Z_{12} is a normalization constant. These are described by

$$\mu_{12} = \arg\left(\kappa_i e^{i\mu_1} + \kappa_2 e^{i\mu_2}\right) \tag{6.18a}$$

$$\kappa_{12} = \left|\kappa_1 e^{i\mu_1} + \kappa_2 e^{i\mu_2}\right| \tag{6.18b}$$

$$Z_{12} = \frac{I_0(\kappa_{12})}{I_0(\kappa_1) I_0(\kappa_2)}. \tag{6.18c}$$

In the convolution step, it is assumed that the number of components remains constant by describing the noise term with a single von Mises component. The prediction step however, leads to an increase in components when the size of both mixture models is greater than one. To highlight this, consider the two general von Mises mixture models with M_1 and M_2 components.

In terms of mixture models, the update step in (6.17) involves a component-wise product. For two mixture models with M_1 and M_2 components respectively, this results in

$$p_{mix}(x) = \sum_{i=1}^{M_1} \sum_{j=1}^{M_2} \pi_{ij} p_{vm}(x \mid \theta_{ij}), \tag{6.19}$$

where $\pi_{ij} = \pi_i \pi_j Z_{ij}$, and Z_{ij} is as the normalization constant obtained as in (6.18).

Note the increase in individual components of the form $M_1 \cdot M_2$. To mitigate this effect, a pruning and merging strategy is defined. First, a pruning step removes components with negligible weight, i.e., those below a predefined threshold π_{th}. These components contribute minimally to the overall distribution and can be discarded without significantly affecting accuracy. Next, the remaining components are iteratively merged to reduce redundancy. At each iteration, the pair of von Mises components with the smallest distance is merged into a single distribution. The pairwise distance is computed by

$$d_{ij} = \frac{\arccos\left(\cos(\mu_i - \mu_j)\right)}{\sqrt{\kappa_i + \kappa_j}}. \tag{6.20}$$

This metric accounts for both angular proximity and concentration, ensuring merging similar distributions. A resulting merged pair can be parameterized into one component by computing the weighted average of the weight, mean, and concentration parameters,

$$\pi_{ij} = \pi_i + \pi_j, \tag{6.21a}$$

$$\mu_{ij} = \frac{\pi_i \mu_i + \pi_j \mu_j}{\pi_i + \pi_j}, \tag{6.21b}$$

$$\kappa_{ij} = \frac{\pi_i \left(\kappa_i + \Delta\mu_i^2\right) + \pi_j \left(\kappa_j + \Delta\mu_j^2\right)}{\pi_i + \pi_j}, \tag{6.21c}$$

where $\Delta\mu_i = (\mu_i - \mu_{ij})$ and $\Delta\mu_j = (\mu_j - \mu_{ij})$.

These expressions represent a weighted average of the parameters, preserving the overall influence of the original components in the merged result. The merging process continues until the total number of components is reduced to a predefined maximum N_{target}.

6.4.2 Learning mixture models

The multimodal representation relying on von Mises distributions can be seamlessly integrated into a deep learning architecture. Rather than predicting an entire probability density function over the yaw domain, the network outputs the parameters of a set of von Mises components, collectively approximating the likelihood $p(y|x)$, denoted by $\hat{p}_{mix}(y|x)$. This strategy keeps the network output compact while retaining a continuous and interpretable density.

The architecture is composed of two stages as illustrated in Figure 6.3. The first stage is a convolutional auto-encoder. An encoder compresses the input image into a low-dimensional latent vector z. Then, a decoder reconstructs the input image from that vector. The reconstruction ensures interpretability of the first stage and

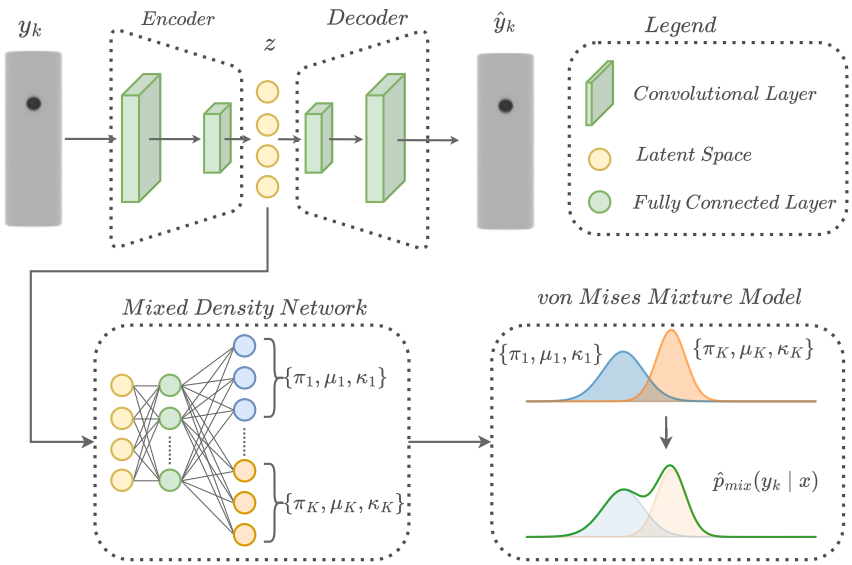


Figure 6.3. Schematic deep learning architecture for learning the von Mises Mixture Model. The first stage is visualized on top, with an encoder and decoder part. The second stage uses the latent space z as input and predicts the individual von Mises components.

confirms expressive features of the latent space. Both the encoder and decoder consist of four convolutional layers with LeakyRelu as the activation layer. Training data for the autoencoder is generated using synthetic images from the digital map with data augmentation applied to enhance generalization. The auto-encoder is implemented and trained using the Pytorch toolbox.

The latent vector z in the auto-encoder is passed to the second stage, named a mixture density network, composed of four fully connected layers. With feature extraction isolated in the first network, the second network solely needs to learn the mapping from z to $\{\pi_k, \mu_k, \kappa_k\}_{k=1}^K$. The network is designed such that the final activation layers ensure valid properties of the von Mises mixture model while remaining fully differentiable for back-propagation. For this, a softmax layer guarantees that the mixture weights π_k sum to one, ensuring a valid probability distribution. A linear layer leaves the means μ_k unconstrained, and a softplus activation enforces strictly positive concentration parameters κ_k.

Attempting to learn the parameter set directly from T_j, and thus avoiding latent vector z, would entangle feature learning and density estimation, making the model's learning process less interpretable and harder to control.

6.4.3 Training targets generation

Since the proposed architecture consists of a two-stage network design, different training data formats are required. In this case, the templates from $\mathcal{T}$ are used. For the auto-encoder of the first stage, creating training data is simple. Namely, its task is to map an input image to the latent space z and decode it into the original image. This means that the input is also the training target.

For the mixed-density network in the second stage, a mixture model needs to be linked to the same input image passed to the auto-encoder. One method to obtain a parameterized likelihood function $p(y|x)$ is by first evaluating each T_j with the template set $\mathcal{T}$ or using the active likelihood estimation method proposed in Section 6.3, and then fit K von Mises components. For the latter, the nonlinear optimization is defined as

$$\begin{aligned} \min_{\{\pi_k,\mu_k,\kappa_k\}_{k=1}^K} \quad & \mathcal{L}(\{\pi_k, \mu_k, \kappa_k\}) \\ \text{s. t.} \quad & \sum_{k=1}^{K} \pi_k = 1, \\ & \pi_i > 0, \quad \kappa_i > 0 \quad \forall i, \end{aligned} \tag{6.22}$$

where the loss function is defined as $\mathcal{L} := \left(p(y|x) - \hat{p}_{mix}(y|x)\right)^2$ with a fine discretization on x. Solving (6.22) for each T_j yields a corresponding set of von Mises mixture parameters, thereby establishing a direct mapping from each template

to its compact probabilistic description. Additionally, to the natural constraints of this optimization problem, extra constraints are defined to ensure that each component may not be a uniform von Mises distribution, and each component must have a certain minimum weight. These constraints ensure a consistent fit for each template.

Generalization is not an inherent outcome of training neural networks, it must be enforced through the generation of training data. To promote generalization in the deep learning method, data augmentation techniques are applied. A targeted strategy involves encouraging the model to focus on the object's shape and structural features rather than its visual appearance. This is achieved by artificially expanding the training set through random hue and saturation changes applied to each template, resulting in a visually diverse dataset. Importantly, the original set of stored templates remains fixed at $|\mathcal{T}| = 720$, and the training set can be replicated N times to generate varied image instances. The associated von Mises components are duplicated alongside each augmented template, but are computed only once from the original, unaltered set, as they remain invariant under visual augmentations.

To assess generalization, the validation and test sets consist exclusively of grayscale images, which are unseen during training, to evaluate the model's ability to infer likelihoods based solely on shape and structure, independent of color variations. This approach is designed to ensure that, although the architecture is trained entirely on synthetic renders, it learns a robust representation of the object to make reliable predictions in real-world settings, including varying lighting conditions and surface reflections.

6.5 Simulation results

This section presents the results obtained from a simulated environment exclusively consisting of synthetic images. A simulated environment allows us to consider the exact pose of the object. In this section we limit ourselves to only estimating the yaw of the object. The object under consideration is a cylindrical tube with a single hole in it, as visualized in Figure 6.4 for $k = \{0, 1, 2\}$. Here it is clearly visible that observations may exhibit ambiguities or non-informative views, and that a sequence of images and rotations is required to uniquely define the object's pose. The templates are generated in Blender. For each template, the object is rotated with 0.5 degrees, resulting in a total of 720 templates.

The setups for the active likelihood estimation and deep learning methods are given in Section 6.5.1 and 6.5.2, respectively. In Section 6.5.3, the performances of the two methods in a simulated environment are demonstrated and evaluated.

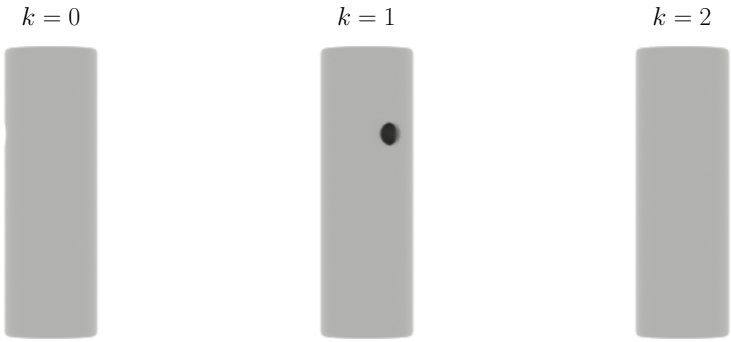


Figure 6.4. Three example synthetic measurements of an arbitrary object.

6.5.1 Active likelihood estimation setup

The presented active likelihood estimation technique via Bayesian optimization in Section 6.3 for similarity estimation can significantly reduce the number of template evaluations while leveraging the filtering principles. The simulation results are obtained by the evaluation of a total of 20 templates. In this case, the computational cost is significantly reduced from 720 image comparisons in an exhaustive search, resulting in a 36-fold reduction. For the GPR, a Matérn32 kernel with a length scale of $\ell = 2.5$ is used.

The procedure starts by evaluating uniformly distributed samples across the yaw domain. These initial evaluations establish a coarse estimate of the similarity function. To ensure well-scaled observation training data, min-max normalization is applied to the current and to-be observed similarity scores using the coefficients derived from this uniform grid. In this case, the active sampling process begins with five templates uniformly distributed across the yaw domain. These serve as the initial point for fitting the GP. Guided by the acquisition function defined in (6.11), the algorithm iteratively selects new templates based on the index that reduces the highest uncertainty steered by the current mean and belief state.

After reaching the maximum number of samples to be taken, the predicted min-max normalized similarity function can be denormalized by the same constants. This predicted similarity function $\hat{L}_y(x)$ is evaluated at discrete state space, and converted into the likelihood function using (6.6).

6.5.2 Deep learning setup

The proposed deep learning method introduced in Section 6.4 is chosen to achieve generalization across varying conditions, which would be an essential condition to implement this kind of approach in real-world applications. A training set is

Table 6.1. Neural network architecture details.

Layer Type	Output Size / Features	Activation
Autoencoder		
Input layer	$3 \times 256 \times 96$	–
C1	$32 \times 128 \times 48$	LeakyReLU
C2	$64 \times 64 \times 24$	LeakyReLU
C3	$128 \times 32 \times 12$	LeakyReLU
C4	$128 \times 16 \times 6$	LeakyReLU
FC1	4	Linear
FC2	3072	Linear
C5	$128 \times 32 \times 12$	LeakyReLU
C6	$64 \times 64 \times 24$	LeakyReLU
C7	$32 \times 128 \times 48$	LeakyReLU
C8	$3 \times 256 \times 96$	Sigmoid
Mixture Density Network		
Input layer	4	–
FC3	128	LeakyReLU
FC4	64	LeakyReLU
FC5	32	LeakyReLU
FC6	3×5	Linear

Notes: All convolutional layers use a kernel size of 3, stride of 2, and padding of 1. Further, the output of FC6 is split into three vectors for each mixture component: mixture weights (softmax normalized), means (sigmoid scaled to $[0, 2\pi]$), and kappas (softplus + 1 to enforce positivity).

constructed that links each template to its corresponding mixture model. In this application for yaw estimation, five components are found to be sufficient for a well-defined approximation of the true similarity function.

Details of the employed neural network architecture are presented in Table 6.1. The table summarizes the network specifications, including the layer types, output sizes (or feature map dimensions), and activation functions. Convolutional layers are abbreviated as C, and fully connected layers as FC throughout the table. For the yaw estimation task, both the autoencoder and mixture density network were trained separately for 120 minutes each on a training set of approximately 1000 images and their linked mixture models.

6.5.3 Model comparison

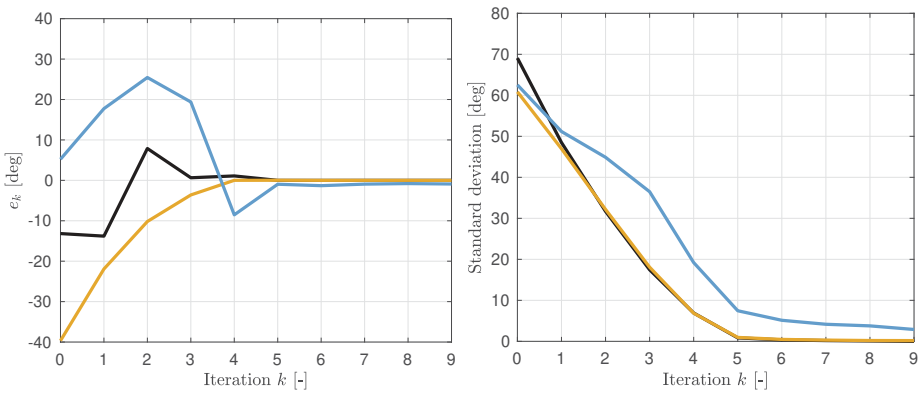
To compare the results of the two methods, both the state estimation error and computation time are considered as important metrics. The state estimation error needs to take the periodic nature of the yaw into account, and is therefore defined as

$$e_k = \operatorname{atan2}\left(\sin(\hat{x}_k - x_k^{true}), \cos(\hat{x}_k - x_k^{true})\right), \tag{6.23}$$

where atan2 is the two-argument arctangent and $\hat{x}_k$ equals the state with the highest belief: $\hat{x}_k = \arg\max_x bel(x)$.

By taking a closer look at the estimation error over the filtering steps for all templates present in $\mathcal{T}$, Figure 6.5 shows the mean error and the standard deviation for each proposed method of the sensor model. Furthermore, Table 6.2 provides the values for the estimation error after 10 filtering steps and the average computation time per filtering step. As is visible, both the active likelihood estimation method and the deep learning method can reduce the estimation error over the 10 time steps of the simulation. The exhaustive search is considered as the best possible filter, as it has access to the true values for all states. The active likelihood estimation approach also performs well, with steadily improving accuracy as the filtering progresses. The deep learning method maintains a slightly larger estimation error and standard deviation compared to the other heuristics. Despite the higher estimation error of the deep learning method, it is still showing its potential, since the filtering computation times are significantly lower in comparison with the other methods. For this yaw estimation problem it is not an immediate issue, but when more degrees of freedom are considered, this advantage becomes more significant.

Finally, this section demonstrated that the two proposed methods are effective in a simulation environment, achieving the necessary accuracies within significantly faster times compared to an exhaustive search approach. However, since the considered use case takes place in a real-world environment, it is interesting how the methods behave in such an industrial setting.



(a) Estimation error. **(b)** Standard deviation.

Figure 6.5. Estimation error and standard deviation of exhaustive search (—), Gaussian process regression (—), and the deep learning method (—).

Table 6.2. Final estimation error (in degrees) and average computation time per filtering step.

Algorithm	Estimation Error ($\mu \pm \sigma$) [deg]	Time [s]
Exhaustive Search	-	24.324
Active Likelihood Estimation	-0.0069 ± 0.18	0.896
Deep Learning	-0.92 ± 2.93	0.077

6.6 Experimental results

This section presents the results obtained from the experimental setup described in the introduction. The setups for the active likelihood estimation and deep learning method are similar to those in Section 6.5. We focus on the yaw estimation of an object which has a similar appearance as shown before in Figure 6.4. A representative hardware setup ensures robust and stable observations, controlled by a software pipeline that is designed and implemented to move the manipulator, take measurements, and ultimately estimate the yaw. The experimental setup is described in Section 6.6.1. In Section 6.6.2, the approximation of the similarity function using GPR, given a real image, is evaluated. A comparison of both the active likelihood estimation via Bayesian optimization and deep learning method on an experimental setup is showcased in Section 6.6.3.

6.6.1 Experimental setup

The core hardware components in the experimental setup are the robotic manipulator and the CMM. The manipulator has 6DOFs and is from AUBO Robotics, the CMM is from Mitutoyo. Both are shown in Figure 6.1.

The software for the experimental setup is built on the Robot Operating System 2 (ROS2) [71], which provides a modular and scalable middleware framework for robotic development. ROS2 enables robust inter-process communication through nodes, topics, and action servers, facilitating seamless coordination between perception, planning, and robot actuation. Further, MoveIt2 [21] is integrated for motion planning and execution. Trajectory generation employs the default Open Motion Planning Library (OMPL) based planner [22], together with a rapidly exploring random tree (RRT) algorithm [64].

The software components can be classified into three layers. A perception layer includes the camera driver, responsible for capturing the images. Another node receives the observation and performs the filtering. The second layer includes the planning layer implements the MoveIt2 framework, which plans trajectories and checks them for any collisions with the environment. Lastly, the control layer includes the robot driver, which executes the planned trajectory on the actual

hardware.

To improve image quality and robustness in the industrial setup, both the camera and external lights are equipped with cross-polarizing filters. A cross-polarization effect is achieved by orienting the camera's polarizing filter perpendicular to those on the light sources. This configuration effectively suppresses reflections from external light sources in the environment, isolating the influence of the bar lights and thereby enhancing environmental robustness.

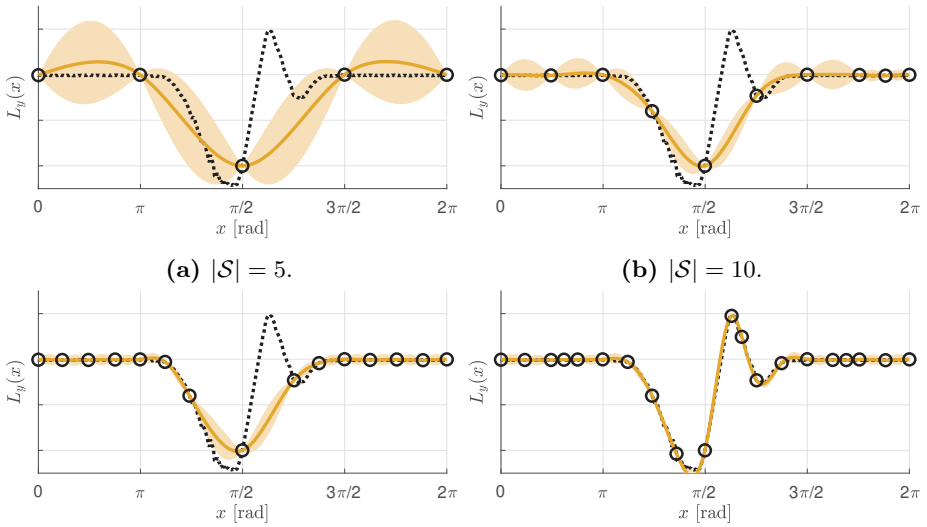
6.6.2 Active likelihood estimation

Using the previously described software and setup, the captured images exhibit a uniform appearance, enabling a reliable sensor model that allows useful comparisons with the synthetic template set $\mathcal{T}$. During runtime, the image region is selected to be slightly larger than the object to account for minor tilt or positional offsets. While this ensures full object visibility, it also increases the number of template shifts evaluated during matching, leading to higher computational cost. A single exhaustive search step can take over 20 seconds when evaluating against a template set of 720 entries, making the approach inefficient, particularly when combined with the time required for robot movements and the potential need to process multiple parts for two CMMs.

This active likelihood estimation approach significantly reduces the number of template evaluations required, while preserving estimation accuracy. In Figure 6.6, the similarity functions $L_y(x)$ and their GPR-based predictions $\hat{L}_y(x)$ are visualized, where $L_y(x)$ is computed by comparing the real image with all 720 templates (as exhaustive search). In the experimental setup, the similarity function is effectively approximated using only 20 evaluations, far fewer than the 720 required for the exhaustive search. Consequently, the acquisition time for the similarity function estimation is reduced from approximately 20 seconds to under 1 second per filtering step. Despite the drastic reduction in evaluations, the resulting estimation closely matches the performance achieved with the full template set, thus preserving filtering quality.

The evolution of the probability distribution estimation using the active likelihood estimation method for 5 filtering steps is visualized in Figure 6.7. Starting from a uniform distribution, the certainty increases after each update step. Eventually, a high peak occurs just after π [rad], indicating the most likely yaw estimate.

Next, the same observations as used in the active likelihood estimation technique are passed to the trained deep learning method. This architecture is trained on synthetically rendered images. The training set consists of 1000 augmented images and their linked mixture models. The filtering steps using the deep learning method are illustrated in Figure 6.8. Initial observations indicate that the filter's propagation behavior are broadly consistent with the results obtained from the



(a) $|\mathcal{S}| = 5$. **(b)** $|\mathcal{S}| = 10$.

(c) $|\mathcal{S}| = 15$. **(d)** $|\mathcal{S}| = 20$.

Figure 6.6. Convergence of the similarity function estimation by GPR. $L_y(x)$ is denoted by (····), the samples $\mathcal{T}_j$ by (○), $\hat{L}_y(x)$ by (—), and (▭) indicates $\pm 2\sigma$.

active sampling technique, and the most likely yaw estimate occurs at the same location.

In Figure 6.9, the obtained likelihoods for both the active likelihood estimation via Bayesian optimization and deep learning method are visualized for every image of the five update steps. For the first three steps, the deep learning likelihood shows peaks at odd locations. Despite that, it is able to assign a low probability to states that are unlikely to occur. In the last two steps, the hole becomes visible, and the deep learning method shows the peaks in probability at the right locations. Even though the deep learning method is not as accurate as the active likelihood estimation method, the main advantage is its computation time, which becomes apparent when one scales to 6DOF pose estimation.

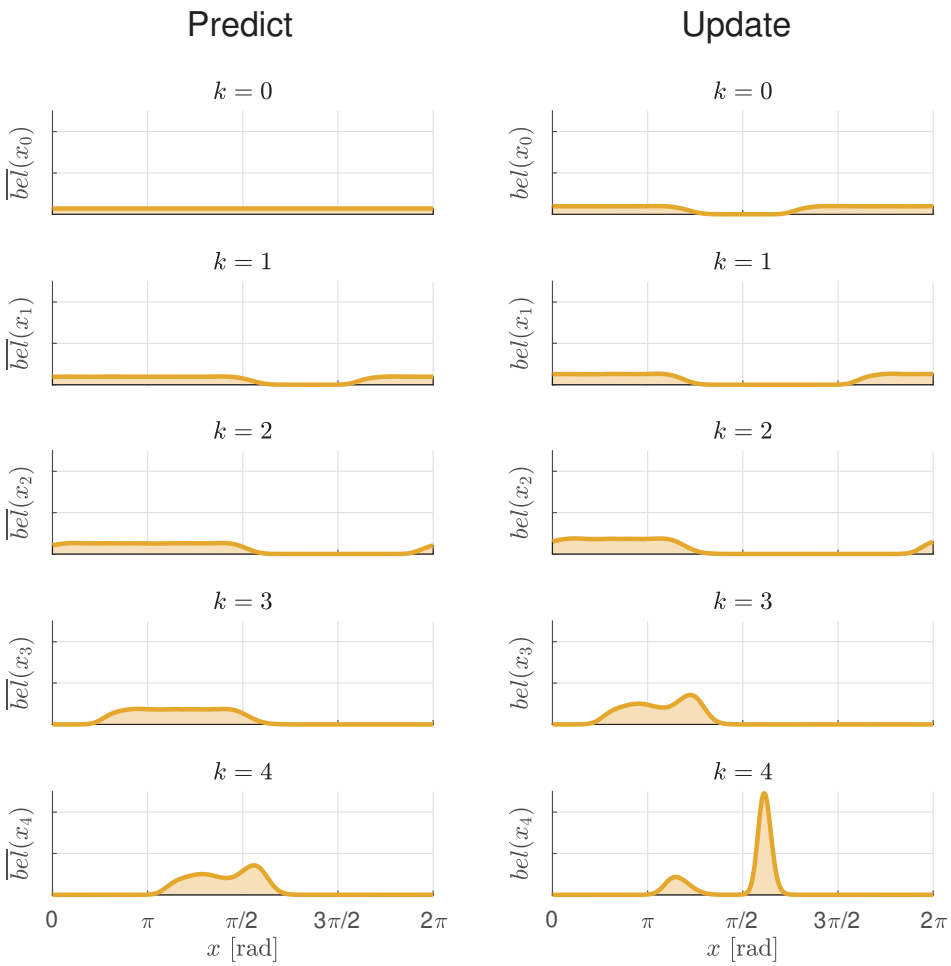


Figure 6.7. Five consecutive filtering steps using active likelihood estimation via Bayesian optimization. Real observations are compared to synthetic templates.

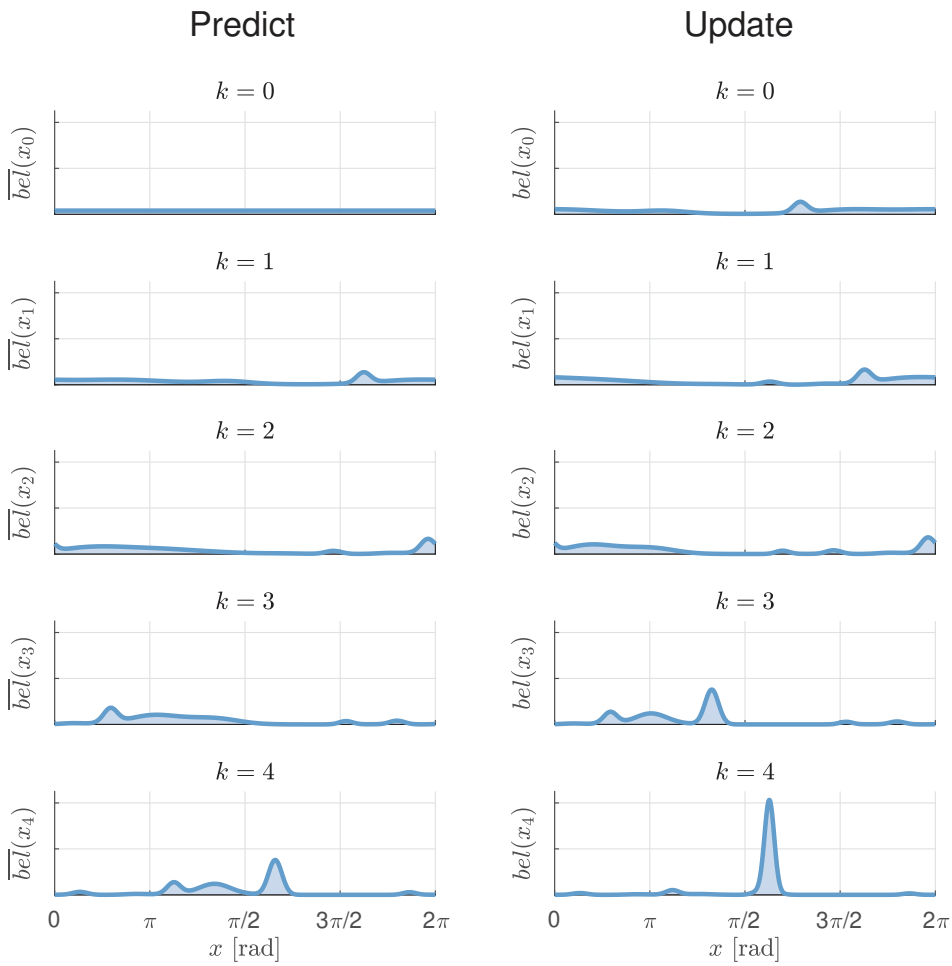


Figure 6.8. Five consecutive filtering steps using the Bayes' filter and the two-staged deep learning method with real object observations.

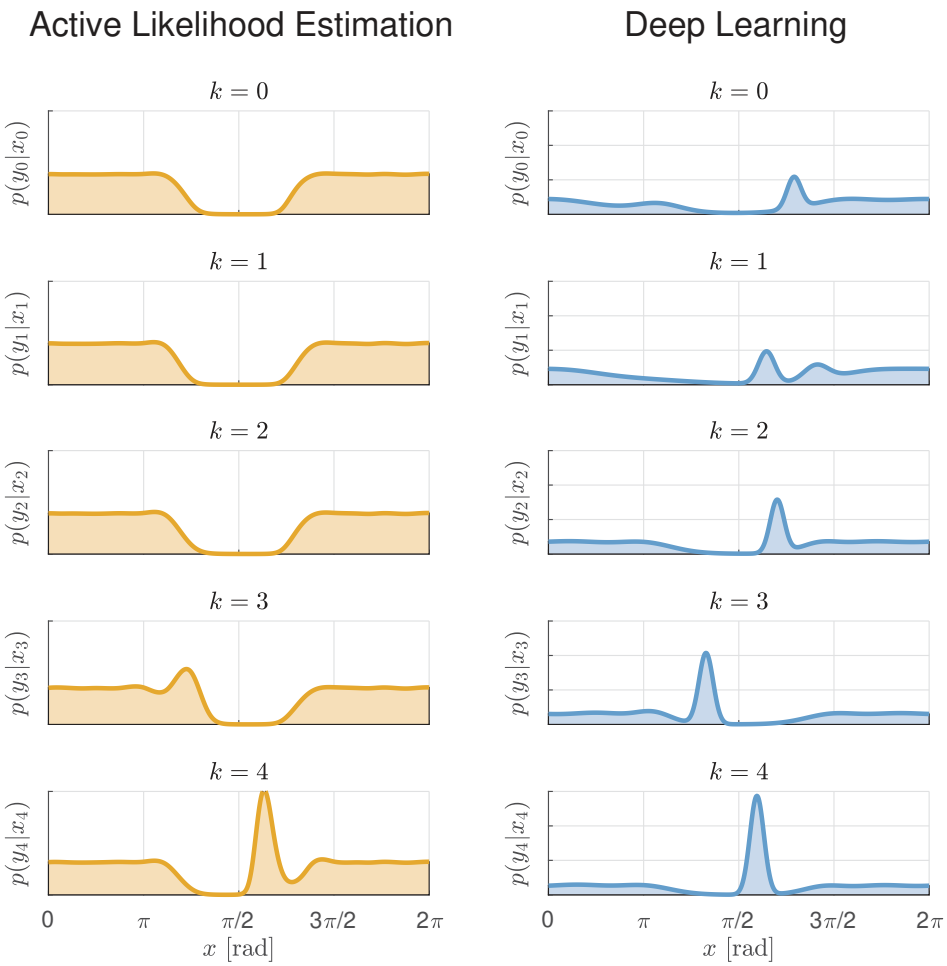


Figure 6.9. The likelihoods used in the filtering propagation for both active likelihood estimation and the deep learning method.

6.6.3 Predictions on real observations

As demonstrated, both the active likelihood estimation and deep learning method yield sufficiently accurate pose estimates. Actually, in the example shown, the estimated pose allows the CMM to successfully execute its inspection program, thereby fulfilling the primary objective of the framework. The active likelihood estimation method is heuristic and based on template matching. As a result, if the observed image differs significantly from the predefined digital map, this method is prone to errors. The deep learning method on the other hand shows predictive capabilities when deployed in the industrial setting. Especially, considering that it is trained on a limited synthetic data set, and the hyperparameters of the neural network architecture and its training procedure are not optimized yet.

6.7 Conclusions and future work

This chapter proposes a fully automated pose estimation framework based on Bayesian filtering, which effectively incorporates multiple observations for improved accuracy. Two methods are introduced within this framework, all compatible with the recursive filtering strategy. The first is a heuristic, template-matching approach that computes similarity scores between observed and pre-rendered views and converts them into likelihoods using a Gaussian formulation. To improve efficiency, Bayesian optimization is integrated for similarity estimation, leading to a significant reduction in template evaluations compared to a method using all templates. The second method introduces a deep learning architecture that maps input images to a latent space and predicts a von Mises mixture model, enabling efficient likelihood prediction using a compact output.

Together, the proposed methods form a scalable and adaptable framework for probabilistic state estimation. The integration of Bayesian filtering with both active likelihood estimation and deep learning predictions balances computational efficiency with estimation accuracy, offering a practical solution for industrial automation tasks involving vision-based yaw estimation. The efficacy of both methods is shown using simulation and experimental results. Both methods are able to accurately and quickly estimate the object's yaw, while bridging the gap between synthetic and real images.

Future work may extend the implementation to full 6DOF pose estimation, including both 3D position and orientation, enabling wider use in robotic tasks requiring full object localization. While this increases computational demands for template matching and GPR, deep learning offers a promising alternative for improved efficiency. However, the performance gap caused by training solely on synthetic data must be addressed. Enhancing generalization remains crucial, either by generating more realistic synthetic data that captures real-world noise (e.g.,

lighting, reflections, and visual artifacts) or by employing advanced augmentation strategies to reduce the domain gap.

Chapter 7

Bayesian Estimation using Fourier Basis Functions

Abstract - Bayesian estimation can be used to estimate the state of dynamical systems, but its applicability is hampered due to the curse of dimensionality. This chapter aims to mitigate this bottleneck for a relevant class of systems consisting of a linear plant with bounded input, driven by stochastic disturbances with non-linear noisy output; the distributions of the disturbances and noise have a bounded support but are otherwise general. Using a frequency-domain interpretation of the operations of the Bayes' filter, we show that, under mild assumptions, exact Bayesian estimation can be pursued in a countable space of Fourier series coefficients, rather than in the usual functional space of probability densities. This fact leads to a natural approximate method, where the Fourier series coefficients corresponding to high frequencies are discarded. For this approximate method, the complexity of the conditioned state distribution, measured by the number of Fourier coefficients, remains constant at prediction steps and grows only linearly at each update step. The applicability of the results is illustrated in the context of electron microscopy, where a residual error analysis indicates that the approximation is accurate.

This chapter is based on: R.A.C. van Zuijlen, J.S. van Hulst, W.P.M.H. Heemels, and D.J. Guerreiro Tomé Antunes, "Bayesian Estimation for Linear Systems with Nonlinear Output and General Noise Distribution using Fourier Basis Functions", *IEEE Conference on Decision and Control*, p. 2166–2171, 2023.

7.1 Introduction

Bayes' filtering is a general approach to the state estimation problem of dynamical systems [33]. It provides the state probability density function (pdf) given previous control inputs and measurements based on two recursive steps: update and prediction. The update step relies on the most recent measurement and the prediction step on the dynamical model.

When the dynamical and output equations are linear and the noise statistics are Gaussian, the Bayes' filter boils down to the Kalman filter. In such a case, the conditioned state distribution is Gaussian at every time step, provided that this is the case at time zero [56]. More generally, the conditioned state distribution is a sum of Gaussians, when the distributions of the initial state, the noise, and the disturbances are sums of Gaussians, although if the noise statistics cannot be represented by a single Gaussian, the number of Gaussians grows exponentially with time [1]. The Kalman filter is one of the rare cases in the context of multivariable systems with general state dimension n_x for which Bayesian estimation is tractable. In fact, without the linear and Gaussian assumptions, one can typically only rely on discretization of the pdf; however, when $n_x \geq 3$ is large, keeping track of a pdf in $\mathbb{R}^{n_x}$ easily becomes computationally intractable.

There are many *approximate* methods, such as the Extended Kalman Filter [2], Unscented Kalman Filter [55], Particle Filter [40], Multiple Model Estimation Algorithm [5], Sequential Monte Carlo Algorithm [53], Variational Bayes' filtering [110], or methods that use Fourier densities to approximate pdfs and perform Bayes' filtering [17]. Although the results are certainly of interest, in general it is hard to quantify the quality of the approximation for these methods.

In this chapter we are interested in a class of systems for which we show that *exact* Bayesian estimation can be pursued and accurately approximated in a tractable way. This class is characterized by:

(i) A linear system with a bounded set of admissible initial states.

(ii) General (non-linear) output equations with bounded independent and identically distributed measurement noise with a general distribution.

(iii) Bounded, independent, and identically distributed process noise with a general distribution.

(iv) Bounded control inputs.

Conceivably, ignoring the assumptions regarding the bounds on the sets constraining the inputs, noise and disturbances for a moment, this is a much broader class of systems when compared to the underlying assumptions in the context of the Kalman filter (or the sums of Gaussian approach).

For the class of systems satisfying (i)-(iv) above, we will interpret the operations of the Bayes' filter in the frequency domain through the Fourier transform of the conditioned pdf. In general, keeping track of such a Fourier transform would be intractable. Since the assumptions render bounded sets, it is possible to represent it in a lossless manner, with samples corresponding to the Fourier series. This means that exact Bayesian estimation can be pursued in the countable space of Fourier series coefficients uniquely representing the conditioned state pdf, rather than in the usual functional space of pdfs. This leads to a natural approximate method, where the Fourier series coefficients corresponding to high frequencies are discarded. For this approximate method, the complexity of the conditioned state distribution, measured by the number of Fourier coefficients, remains constant at prediction steps and grows only linearly at each update step, as we will see.

The use of Fourier series in a Bayesian estimation context can lead to large data efficiency gains (see the numerical examples in Section 7.4) as well-known in other domains (compression into JPEG-images and MP3-files). Due to this, it can provide a viable solution for medium-scale problems $4 \leq n_x \leq 6$ where discretization would not be feasible. The use of Fourier series in this context has also been proposed in [17]. Contrary to [17], our proposed method uses Fourier series rather than their approximated Fourier densities, and does not require a transition density function but uses the linear system dynamics directly to compute the pdf of the state in the prediction step.

The proposed model and estimation method were inspired by a problem in the electron microscopy field. Electron microscopes require tuning by an expert operator to compensate for present aberrations [46]. The estimation of the aberration relies on the visual inspection of a so-called ronchigram [95, 101]. Based on one single ronchigram, it is impossible to uniquely identify the aberrations [100]. To address this problem in a systematic manner, it turns out to be useful to consider the electron microscope as an integrator (and thus linear) system, where the (unknown) states describe the aberrations and the control inputs are the tuning knobs to compensate for those aberrations. The output is a complex non-linear function based on the states. As this setting fits the assumptions of our framework, we apply the proposed method in this context and discuss its effectiveness through a residual error analysis. This analysis indicates that the approximation is accurate.

This chapter is structured as follows. The proposed model is introduced in Section 7.2, where the Bayes' filter in the time-domain is reviewed. The exact Bayes' filter written with Fourier transforms and Fourier series, and the approximate Bayes' filter using only a finite number of Fourier coefficients are given in Section 7.3. Two numerical examples of the approximate method that demonstrates the performance of the Bayes' filter with Fourier basis functions are given in Section 7.4, including a comparison with the Unscented Kalman Filter. Finally, in Section 7.5, conclusions are drawn and directions for future research are given.

7.2 Proposed model and Bayes' filter

Consider the system

$$x_{t+1} = f(x_t, u_t, w_t) \tag{7.1a}$$
$$y_t = g(x_t, v_t) \tag{7.1b}$$

with linear dynamics, i.e.,

$$f(x, u, w) = Ax + Bu + w \tag{7.2}$$

and where $x_t \in \mathbb{R}^{n_x}$ is the state, $u_t \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$ is the control input, $y_t \in \mathbb{R}^{n_y}$ the measured output at time $t \in \mathcal{T} := \{0, 1, \ldots, h\}$, where $h \in \mathbb{N} \cup \{\infty\}$ is a time horizon of interest. The initial state is assumed to belong to a state $x_0 \in \mathcal{X}_0 \subseteq \mathbb{R}^{n_x}$. Moreover, $w_t \in \mathcal{W} \subseteq \mathbb{R}^{n_x}$ and $v_t \in \mathcal{V} \subseteq \mathbb{R}^{n_y}$, $t \in \mathcal{T}$, form sequences of zero-mean, independent and identically distributed, and mutually independent, random variables, representing the process disturbances and the measurement noise, respectively. Disturbances and measurement noises w_t and v_t, $t \in \mathcal{T}$, can be described by any probability density function (pdf), and are therefore not restricted to zero-mean Gaussian noise. We denote these pdfs by $p_w(w)$ and $p_v(v)$, respectively. Hence, for all $t \in \mathcal{T}$, $\mathcal{A} \subseteq \mathcal{W}$, $\mathcal{B} \subseteq \mathcal{V}$, it holds that

$$\int_{\mathcal{A}} p_w(w)dw = \text{Prob}[w_t \in \mathcal{A}], \tag{7.3}$$
$$\int_{\mathcal{B}} p_v(v)dv = \text{Prob}[v_t \in \mathcal{B}]. \tag{7.4}$$

The function $g(\cdot, \cdot)$ may be a general nonlinear function, and we do not put any particular restrictions on it. We can equivalently characterize the output equation (7.1b) by a pdf $p_y(y; x)$ such that, for all $t \in \mathcal{T}$,

$$\int_{\mathcal{C}} p_y(y; x)dy = \text{Prob}[y_t \in \mathcal{C} | x_t = x] = \int_{\mathcal{B}_{\mathcal{C}}} p_v(v)dv, \tag{7.5}$$

where $\mathcal{B}_{\mathcal{C}} := \{v \in \mathcal{V} | g(x, v) \in \mathcal{C}\}$.

Similarly, we can equivalently characterize the process equation (7.1a) by a pdf $q(\bar{x}; x, u)$, such that, for all $t \in \mathbb{N}_0$,

$$\int_{\mathcal{D}} q(\bar{x}; x, u)d\bar{x} = \text{Prob}[x_{t+1} \in \mathcal{D} | x_t = x, u_t = u] = \int_{\mathcal{E}_{\mathcal{D}}} p_w(\bar{x})dw \tag{7.6}$$

with $\mathcal{E}_{\mathcal{D}} := \{w \in \mathcal{W} | f(x, u, w) \in \mathcal{D}\}$.

The initial state x_0 is assumed to be distributed according to a pdf $\bar{p}_0(x_0)$, with support $\mathcal{X}_0$. A standing assumption is the following.

Assumption 7.1. The sets $\mathcal{W}$, $\mathcal{V}$, $\mathcal{U}$ and $\mathcal{X}_0$ are bounded.

A direct consequence of this assumption is that, for all $t \in \mathcal{T}$,

$$x_t \in \mathcal{X}_t, \text{ for some bounded sets } \mathcal{X}_t. \tag{7.7}$$

The sets $\mathcal{X}_t$ are assumed to be minimum volume sets such that $x_t \in \mathcal{X}_t$. We define the set of rectangular sets in $\mathbb{R}^{n_x}$ as

$$\mathsf{R} = \{[\underline{L}_1, \bar{L}_1] \times \cdots \times [\underline{L}_{n_x}, \bar{L}_{n_x}] | \bar{L}_i > \underline{L}_i, i \in \{1, ..., n_x\}\}. \tag{7.8}$$

Here $\underline{L}_i$ and $\bar{L}_i$ are the lower- and upper-bounds, respectively, for the i-th entry of $\mathcal{X}_t$. The volume of $[\underline{L}_1, \bar{L}_1] \times \cdots \times [\underline{L}_{n_x}, \bar{L}_{n_x}] \in \mathsf{R}$ is $\tilde{L} = \Pi_{i=1}^{n_x} L_i$, $L_i = \bar{L}_i - \underline{L}_i$. Let $\mathcal{R}_t$ be the set in R with minimal volume such that $\mathcal{X}_t \subseteq \mathcal{R}_t$. Let

$$\mathcal{R} := \cup_{t=1}^{h} \mathcal{R}_t. \tag{7.9}$$

Assumption 7.2. When $h = \infty$, $\mathcal{R}$ is bounded.

A sufficient condition for this assumption to hold is that the system matrix A is Schur. If this holds, the system is input-to-state stable, which implies that when $\mathcal{X}_0$, $\mathcal{U}$ and $\mathcal{W}$ are bounded, $\mathcal{X}_t$ is also bounded.

Assumption 7.3. A is invertible.

Assumption 7.3 is met, for instance, when the dynamical model results from an exact time-discretisation of a continuous-time dynamical model.

The Bayes' filter allows us to keep track of conditioned pdfs $p_{i|j}(\cdot)$, where $i \in \mathcal{T}$ is the timestep to be estimated and $j \leq i$ the input and output information available at timestep i, that is, $\{u_0, ..., u_{j-1}, y_0, ..., y_j\}$. The Bayes' filter recursively performs the following two steps for every $t \in \mathcal{T}$ with $p_{0|-1}(x) = \bar{p}_0(x)$, and given u_t, for every $x \in \mathcal{X}_0$.

(i) Update step:

$$p_{t|t}(x_t) = \frac{1}{\alpha} p_y(y_t; x_t) p_{t|t-1}(x_t), \quad x_t \in \mathcal{X}_t, \tag{7.10}$$

with $\alpha = \int_{\mathcal{X}_t} p_y(y_t; s) p_{t|t-1}(s) ds$.

(ii) Prediction step:

$$p_{t+1|t}(x_{t+1}) = \int_{\mathcal{X}_t} q(x_{t+1}; x_t, u_t) p_{t|t}(x_t) dx_t, \quad x_{t+1} \in \mathcal{X}_{t+1}. \tag{7.11}$$

For the linear model considered, we can write the prediction step as follows. Consider two scalar functions $a : \mathcal{A} \to \mathbb{R}$, $b : \mathcal{B} \to \mathbb{R}$ with domains in sets $\mathcal{A} \subseteq \mathbb{R}^n$, $\mathcal{B} \subseteq \mathbb{R}^n$. If $\mathcal{A}$ and $\mathcal{B}$ are bounded sets, extend these functions to $\mathbb{R}^n$ by setting $a(x) = 0$, $x \in \mathbb{R}^n \setminus \mathcal{A}$, $b(x) = 0$, $x \in \mathbb{R}^n \setminus \mathcal{B}$, respectively. Then, convolution $c = a \otimes b$ is denoted by $c(x) = (a \otimes b)(x) = \int_{\mathbb{R}^n} a(x-s)b(s)ds$ for $x \in \mathbb{R}^n$.

Proposition 7.1. *The prediction step can equivalently be written as*

$$p_{t+1|t}(x) = \frac{1}{\det(A)} p_{t|t}(A^{-1}(x - Bu_t)) \otimes p_w(w), \quad x \in \mathcal{X}_{t+1} \tag{7.12}$$

for every $t \in \mathcal{T}$. ▲

Proof: First, consider the transformation without disturbances. Let $x_{t+1} = h(x_t, u_t)$, where $h(\cdot)$ is a one-to-one differentiable function on support $\mathcal{X}_t$, while u_t is fixed. Then $\bar{p}_{t+1|t}(x)$ is given by

$$\bar{p}_{t+1|t}(x) = p_{t|t}\left(h^{-1}(x, u_t)\right) \left| \frac{d[h^{-1}(x, u_t)]}{dx} \right|, \tag{7.13}$$

where $|\cdot|$ denotes the determinant. For proof, see [48]. Due to the linear system dynamics $x_{t+1} = Ax_t + Bu_t$,

$$h^{-1}(x, u_t) = A^{-1}(x - Bu_t), \tag{7.14}$$

and

$$\left| \frac{d[h^{-1}(x, u_t)]}{dx} \right| = \left|A^{-1}\right| = |A|^{-1} = \frac{1}{\det(A)}. \tag{7.15}$$

The addition of disturbances w_t to the linear system dynamics can be seen as the sum of two random vectors. Let $x_{t+1} = f(x_t, u_t) + w_t$, and the pdf of x_{t+1}, $f(x_t, u_t)$ and w_t be denoted by $p_{t+1|t}(x)$, $\bar{p}_{t+1|t}(x)$ and $p_w(w)$, respectively, then

$$p_{t+1|t}(x) = \bar{p}_{t+1|t}(x) \otimes p_w(w). \tag{7.16}$$

Combining (7.13) and (7.16) concludes the proof.

■

While $p_{t+1|t}(x)$ and $p_{t|t}(x)$ have only been defined by the Bayes' filter in the intervals $\mathcal{X}_{t+1}$ and $\mathcal{X}_t$, it is convenient to define them for every $x \in \mathbb{R}^{n_x}$ by setting $p_{t+1|t}(x) = 0$ if $x \in \mathbb{R}^{n_x} \setminus \mathcal{X}_{t+1}$, $p_{t|t}(x) = 0$ if $x \in \mathbb{R}^{n_x} \setminus \mathcal{X}_t$. In turn $p_y(y; x)$ and $p_w(w)$ only needs to be defined for $x \in \mathcal{X}$ and $w \in \mathcal{W}$, but we also define them for every $x \in \mathbb{R}^{n_x}$ by setting $p_y(y; x) = 0$, if $x \in \mathbb{R}^{n_x} \setminus \mathcal{X}$, $p_w(w) = 0$ if $w \in \mathbb{R}^{n_w} \setminus \mathcal{W}$.

Performing the Bayes' filter operations is in general computationally unfeasible when n_x is large (typically when $n_x \leq 3$ one can still rely on spatial discretization, but for larger n_x this becomes unfeasible). This motivates the problem considered here which is to represent $p_{t+1|t}(x)$ and $p_{t|t}(x)$ in an efficient way and to provide a numerically efficient method to perform these operations.

7.3 Bayes' filter with Fourier transform and Fourier basis functions

In this section, the Bayes' filter is pursued using the Fourier transform and Fourier basis functions. Section 7.3.1 provides an *exact* method, which requires an infinite number of coefficients. Section 7.3.2 provides an *approximate* method with a finite number of coefficients.

7.3.1 Exact method

For every $t \in \mathcal{T}$, $\omega \in \mathbb{R}^{n_x}$, let

$$\begin{aligned} P_{t|t}(\omega) &:= \int_{\mathbb{R}^{n_x}} p_{t|t}(x) e^{-j\omega \cdot x} dx, \\ P_{t|t-1}(\omega) &:= \int_{\mathbb{R}^{n_x}} p_{t|t-1}(x) e^{-j\omega \cdot x} dx, \\ P_w(\omega) &:= \int_{\mathbb{R}^{n_w}} p_w(w) e^{-j\omega \cdot w} dw, \\ P_y(y;\omega) &:= \int_{\mathbb{R}^{n_y}} p_y(y;x) e^{-j\omega \cdot y} dy, \end{aligned} \tag{7.17}$$

be the Fourier transforms of the pdfs $p_{t|t}(x)$, $p_{t|t-1}(x)$, $p_w(w)$ and of $p_y(y;x)$, where $a \cdot b$ denote the inner production of two vectors $a \in \mathbb{R}^n$, $b \in \mathbb{R}^n$. Then, the operations of the Bayes' filter can be written as follows.

Lemma 7.1. *The Fourier transforms* (7.17) *are related for every* $t \in \mathcal{T}$ *by*

- *Update step:*

$$P_{t|t}(\omega) = \frac{1}{\alpha} P_y(y_t;\omega) \otimes P_{t|t-1}(\omega) \tag{7.18}$$

  for all $\omega \in \mathbb{R}^{n_x}$ *with* $\alpha = P_{y|x}(y_t;\omega) \otimes P_{t|t-1}(\omega)_{|\omega=0}$.

- *Prediction step:*

$$P_{t+1|t}(\omega) = P_{t|t}(A^\top \omega) e^{-j\omega \cdot (Bu_t)} P_w(\omega), \quad \omega \in \mathbb{R}^{n_x}. \tag{7.19}$$

▲

Proof: We use the notation $x(t) \overset{\mathscr{F}}{\longleftrightarrow} X(\omega)$ to indicate that $X(\omega)$ is the Fourier transform of $x(t)$. (7.18) follows from applying the linearity and multiplication property of the Fourier transform on (7.10):

$$\frac{1}{\alpha} p_y(y_t;x_t) p_{t|t-1}(x_t) \overset{\mathscr{F}}{\longleftrightarrow} \frac{1}{\alpha} P_y(y_t;\omega) \otimes P_{t|t-1}(\omega) \tag{7.20}$$

Since $P_{|\cdot|}(0) = \int_{-\infty}^{\infty} p_{|\cdot|}(x)e^{j0t}dx = \int_{-\infty}^{\infty} p_{|\cdot|}(x)dx$, α is given by $\alpha = P_{y|x}(y_t;\omega) \otimes P_{t|t-1}(\omega)_{|_{\omega=0}}$. (7.19) follows from applying the linearity, time-shifting, time-scaling, and convolution properties of the Fourier transform on Proposition 7.1:

$$\begin{aligned}
\frac{1}{\det(A)} p_{t|t}(x) &\overset{\mathscr{F}}{\longleftrightarrow} \frac{1}{\det(A)} P_{t|t}(\omega), \\
p_{t|t}(A^{-1}x) &\overset{\mathscr{F}}{\longleftrightarrow} \frac{1}{\det(A)^{-1}} P_{t|t}(A^{\top}\omega) \\
&\qquad = \det(A)\, P_{t|t}(A^{\top}\omega), \\
p_{t|t}(x - Bu_t) &\overset{\mathscr{F}}{\longleftrightarrow} P_{t|t}(\omega) e^{-j\omega\cdot(Bu_t)}, \\
p_{t|t}(x) \otimes p_w(x) &\overset{\mathscr{F}}{\longleftrightarrow} P_{t|t}(\omega) P_w(\omega).
\end{aligned} \tag{7.21}$$

■

Interestingly, while the update and predictions steps in the time domain correspond to a product and a convolution (besides the affine transformation), respectively, in terms of the Fourier transform the update and predictions steps in the frequency domain correspond to a convolution and a product (besides the affine transformation). Hence, running the Bayes' filter with Fourier transforms is equally intractable. However, since the state belongs to a bounded set we only need enough samples to reconstruct the Fourier transform, and we can compute the Fourier series instead.

Let $k = [k_1 \ \dots \ k_n]^{\top}$ with $k_i \in \mathbb{Z}$, so that $k \in \mathcal{K} := \{[k_1 \ \dots \ k_n]^{\top} | k_i \in \mathbb{Z}\}$. Moreover, let $k./L := [k_1/L_1 \ \dots \ k_n/L_n]^{\top}$. Suppose that we sample and scale the Fourier transforms $P_{t|t}(\omega)$, $P_{t|t-1}(\omega)$, $P_y(y;\omega)$ obtaining the Fourier coefficients for every $k \in \mathcal{K}$, for every $t \in \mathcal{T}$

$$\begin{aligned}
c_{t|t}[k] &= \frac{1}{\tilde{L}} P_{t|t}(\omega)_{|_{\omega=(\frac{2\pi k_1}{L_1},\dots,\frac{2\pi k_n}{L_n})}}, \\
c_{t|t-1}[k] &= \frac{1}{\tilde{L}} P_{t|t-1}(\omega)_{|_{\omega=(\frac{2\pi k_1}{L_1},\dots,\frac{2\pi k_n}{L_n})}}, \\
c_y[y;k] &= \frac{1}{\tilde{L}} P_y(y;\omega)_{|_{\omega=(\frac{2\pi k_1}{L_1},\dots,\frac{2\pi k_n}{L_n})}}.
\end{aligned} \tag{7.22}$$

These Fourier coefficients are enough to write the conditioned pdfs in $x_t \in \mathcal{R}$, $x_{t+1} \in \mathcal{R}$,

$$\begin{aligned}
p_{t|t}(x_t) &= \sum_{k\in\mathcal{K}} c_{t|t}[k] e^{j2\pi(k./L)\cdot x_t}, \quad x_t \in \mathcal{R}, \\
p_{t|t-1}(x_{t+1}) &= \sum_{k\in\mathcal{K}} c_{t|t-1}[k] e^{j2\pi(k./L)\cdot x_{t+1}}, \quad x_{t+1} \in \mathcal{R}, \\
p_y(y;x) &= \sum_{k\in\mathcal{K}} c_y[y;k] e^{j2\pi(k./L)\cdot x}, \quad x \in \mathcal{R}.
\end{aligned} \tag{7.23}$$

Outside $\mathcal{R}$, the formulas on the right-hand side define periodic replications of these pdfs.

Due to the crucial fact that $x_t \in \mathcal{R}$ for every $t \in \mathcal{T}$, where $\mathcal{R}$ is a bounded set, which follows from Assumption 7.1, we can use the sampling theorem, to conclude that these Fourier coefficients fully characterize the Fourier transforms $P_{t|t}(\omega)$, $P_{t|t-1}(\omega)$. In fact, let

$$B_1(\omega, L, k) := \prod_{i=1}^{n_x} \frac{2\sin((\omega_i - 2\pi/L_i k_i)\frac{Li}{2})}{(\omega_i - 2\pi/L_i k_i)} e^{-j(\omega_i - 2\pi k_i/L_i)(\bar{L}_i + \underline{L}_i)/2} \tag{7.24}$$

where $\frac{\sin(x)}{x}$ is assumed to be extended by continuity so that $\frac{\sin(x)}{x}|_{x=0} = 1$. The derivation of this expression can be found in Section 7.6.1. Then, for every $\omega \in \mathbb{R}^{n_x}$, $t \in \mathcal{T}$,

$$\begin{aligned} P_{t|t}(\omega) &= \sum_{k\in\mathcal{K}} c_{t|t}[k] B_1(\omega, L, k), \\ P_{t|t-1}(\omega) &= \sum_{k\in\mathcal{K}} c_{t|t-1}[k] B_1(\omega, L, k), \\ P_y(y;\omega) &= \sum_{k\in\mathcal{K}} c_y[y;k] B_1(\omega, L, k). \end{aligned} \tag{7.25}$$

To perform the Bayes' filter operations with the Fourier series coefficients requires the discrete convolution operation. The convolution of $c : \mathcal{K} \to \mathbb{R}$ and $d : \mathcal{K} \to \mathbb{R}$, is denoted by $g[k] = c[k] \otimes d[k] = \sum_{r\in\mathcal{K}} c[k-r]d[r], \quad k \in \mathcal{K}$.

Lemma 7.2. *The Fourier coefficients* (7.22) *are related for every* $t \in \mathcal{T}$ *by*

- *Update step: for* $k \in \mathcal{K}$

$$c_{t|t}[k] = \frac{1}{\alpha} c_y[y_t;k] \otimes c_{t|t-1}[k] \tag{7.26}$$

  with $\alpha = c_y[y_t;k] \otimes c_{t|t-1}[k]_{|k=0}$.

- *Prediction step: for* $k \in \mathcal{K}$

$$\begin{aligned} &c_{t+1|t}[k] = \\ &\frac{1}{\tilde{L}} \left(\sum_{k\in\mathcal{K}} c_{t|t}[k] B_1(A^\top \omega, L, k) \right) e^{-j\omega\cdot(Bu_t)} P_w(\omega)_{|_{\omega=(\frac{2\pi k_1}{L_1},\ldots,\frac{2\pi k_n}{L_n})}}. \end{aligned} \tag{7.27}$$

▲

Proof: The update step is derived similarly as in (7.18), but with the properties of the Fourier series rather than the Fourier transform. The prediction step of (7.27) follows from

$$\begin{aligned} c_{t+1|t}[k] &= \frac{1}{\tilde{L}} P_{t+1|t}(\omega)|_{\omega=(\frac{2\pi k_1}{L_1},\ldots,\frac{2\pi k_n}{L_n})} \\ &= \frac{1}{\tilde{L}} P_{t|t}(A^\top \omega) e^{-j\omega\cdot(Bu_t)} P_w(\omega)|_{\omega=(\frac{2\pi k_1}{L_1},\ldots,\frac{2\pi k_n}{L_n})} \\ &= \frac{1}{\tilde{L}} \left(\sum_{k\in\mathcal{K}} c_{t|t}[k] B_1(A^\top \omega, L, k) \right) e^{-j\omega\cdot(Bu_t)} P_w(\omega)|_{\omega=(\frac{2\pi k_1}{L_1},\ldots,\frac{2\pi k_n}{L_n})}. \end{aligned} \tag{7.28}$$

■

Note that when $A = I$, the prediction step (7.27) boils down to a much simpler expression

$$c_{t+1|t}[k] = c_{t|t}[k] e^{-j\omega\cdot(Bu_t)} P_w(\omega)|_{\omega=(\frac{2\pi k_1}{L_1},\ldots,\frac{2\pi k_n}{L_n})}, \tag{7.29}$$

where each $c_{t+1|t}[k]$ only depends on $c_{t|t}[\ell]$ if $k = \ell$.

7.3.2 Approximate method

Suppose we approximate the initial conditioned state distribution with a finite number of Fourier coefficients

$$\hat{c}_{0|-1}[k] := \begin{cases} c_{0|-1}[k] \text{ if } k \in \mathcal{K}_{0|-1}, \\ 0 \text{ otherwise }, \end{cases} \tag{7.30}$$

where

$$\mathcal{K}_{0|-1} = \{-m_{0|-1}, -m_{0|-1}+1 \ldots, m_{0|-1}-1, m_{0|-1}\} \tag{7.31}$$

with $m_{0|-1} \in \mathbb{N}$, i.e., we discard the coefficients corresponding to $k \in \mathcal{K}^c_{0|-1} = \mathcal{K} \setminus \mathcal{K}_{0|-1}$. Then

$$\hat{p}_{0|-1}(x) = \begin{cases} \sum\limits_{k\in\mathcal{K}_{0|-1}} \hat{c}_{0|-1}[k] e^{j2\pi(k./L)\cdot x}, & \text{if } x \in \mathcal{R}, \\ 0, \quad \text{if } x \in \mathbb{R}^{n_x} \setminus \mathcal{R}. \end{cases} \tag{7.32}$$

Note however that $\hat{p}_{0|-1}(x)$ is in general not a pdf, since it is not necessarily positive and does not necessarily integrate to one. However, given any $\hat{p}(x)$

in the space $\mathcal{P}$ of bounded functions in the bounded set $\mathcal{R}$ we can defined a normalization operator $\mathsf{N} : \mathcal{P} \to \mathcal{P}$ that to each $\hat{p}(x) \in \mathcal{P}$ provides $\tilde{p}(x) = \frac{\hat{p}(x)+\epsilon}{\alpha(\epsilon)}$ with $\epsilon = \min\{c \geq 0 | \hat{p}(x) + c \geq 0 \text{ for all } x \in \mathcal{R}\}$ and $\alpha(\epsilon) = \int_{\mathcal{R}} (\hat{p}(x) + \epsilon) dx$. If $\hat{p}(x) = \sum_{k \in \mathcal{K}} c[k] e^{2\pi (k./L) \cdot x}$ for a set $\mathcal{K}$ with finite cardinality $n_K = |\mathcal{K}|$ this induces a normalization procedure to the $\hat{c}[k]$, defined by map $\mathsf{N}_c : \mathbb{R}^{n_K} \to \mathbb{R}^{n_K}$,

$$\tilde{c}[k] = \begin{cases} \dfrac{\hat{c}[k]}{\alpha(\epsilon)}, \text{ if } k \in \mathcal{K} \setminus \{0\}, \\ \dfrac{\hat{c}[0] + \epsilon}{\alpha(\epsilon)} \text{ if } k = 0. \end{cases} \tag{7.33}$$

Thus, $\tilde{p}_{0|-1}(x) = \mathsf{N}(\hat{p}_{0|-1}(x))$ is described by

$$\tilde{p}_{0|-1}(x) = \begin{cases} \displaystyle\sum_{k \in \mathcal{K}_{0|-1}} \tilde{c}_{0|-1}[k] e^{j2\pi (k./L) \cdot x}, & \text{if } x \in \mathcal{R}, \\ 0, \quad \text{if } x \in \mathbb{R}^{n_x} \setminus \mathcal{R}, \end{cases} \tag{7.34}$$

where

$$\tilde{c}_{0|-1}[k] = \mathsf{N}_c(\hat{c}_{0|-1}[k]), \tag{7.35}$$

which is a pdf.

Likewise suppose that we approximate $p_y(y; x)$ with a finite number of Fourier coefficients

$$\hat{c}_y[y; k] := \begin{cases} c_y[y; k] \text{ if } k \in \mathcal{K}_y, \\ 0 \text{ otherwise }, \end{cases} \tag{7.36}$$

where $\mathcal{K}_y = \{-m_y, -m_y + 1 \ldots, m_y - 1, m_y\}$ with $m_y \in \mathbb{N}$. Then, letting $\tilde{c}_y[y; k] = \mathsf{N}_c(\hat{c}_y[y; k])$, $k \in \mathcal{K}_y$, results in a pdf

$$\tilde{p}_y(y; x) = \begin{cases} \displaystyle\sum_{k \in \mathcal{K}_y} \tilde{c}_y[y; k] e^{j2\pi (k./L) \cdot x}, & \text{if } x \in \mathcal{R}, \\ 0, \quad \text{if } x \in \mathbb{R}^{n_x} \setminus \mathcal{R}. \end{cases} \tag{7.37}$$

This leads naturally to the following approximate Bayes' algorithm, which provides estimates $\tilde{p}_{t|t}(x_t)$ and $\tilde{p}_{t+1|t}(x_{t+1})$ of $p_{t|t}(x_t)$ and $p_{t+1|t}(x_{t+1})$. Let $\mathcal{K}_{t|t} = \{-m_{t|t}, -m_{t|t} + 1 \ldots, m_{t|t}\}$, $\mathcal{K}_{t+1|t} = \{m_{t+1|t}, -m_{t+1|t} + 1 \ldots, m_{t+1|t} - 1, m_{t+1|t}\}$.

Algorithm 7.1. (Approximate Bayesian estimation)
Consider initially (7.35) and then, for every $t \in \mathcal{T}$:

- Update step: Set $m_{t|t} = m_{t|t-1} + m_y$ and for $k \in \mathcal{K}_{t|t}$ compute

$$\tilde{c}_{t|t}[k] = \mathsf{N}_c(\tilde{c}_y[y_t; k] \otimes \tilde{c}_{t|t-1}[k]) \tag{7.38}$$

and set

$$\tilde{p}_{t|t}(x_t) = \sum_{k \in \mathcal{K}_{t|t}} \tilde{c}_{t|t}[k] e^{j2\pi(k./L)\cdot x_t}, \quad x_t \in \mathcal{X}. \tag{7.39}$$

- Prediction step: Set $m_{t+1|t} = m_{t|t}$ and for $k \in \mathcal{K}_{t+1|t}$ compute

$$\tilde{c}_{t+1|t}[k] = \mathsf{N}_c\left(\frac{1}{\tilde{L}}\left(\sum_{k \in \mathcal{K}_{t|t}} \tilde{c}_{t|t}[k] B_1(A^\top \omega, L, k)\right) e^{-j\omega\cdot(Bu_t)} P_w(\omega)\Big|_{\omega=(\frac{2\pi k_1}{L_1},\ldots,\frac{2\pi k_n}{L_n})}\right) \tag{7.40}$$

and set

$$\tilde{p}_{t+1|t}(x_{t+1}) = \sum_{k \in \mathcal{K}_{t+1|t}} \tilde{c}_{t+1|t}[k] e^{j2\pi(k./L)\cdot x_{t+1}}, \quad x_{t+1} \in \mathcal{X}. \tag{7.41}$$

□

Note that the factor $m_{t|t} = m_{t+1|t}$ determining the number of coefficients $(2m_{t|t-1} + 1)$ only grows linearly with time t, $m_{t|t-1} = m_{0|-1} + tm_y$. In fact, the complexity of the conditioned state distribution, measured by the number of coefficients, remains constant at prediction steps and grows only linearly at each update step. The number of coefficients $m_{0|-1}$ and m_y to be selected depends on the balance between accuracy and computational effort; more coefficients increase the accuracy of the pdf at the expense of computational effort. A natural choice is to remove the coefficients, whose power is below a predefined threshold after the update step.

7.4 Numerical examples

In this section, two examples are discussed. The first example considers a scalar linear system with Gaussian noise and process disturbances, which allows us to compare the accuracy of the proposed approximate method of Section 7.3.2 to the exact conditioned pdfs obtained using the Kalman filter. In the second example, we consider an example inspired by electron microscopy.

7.4.1 Comparison with the Kalman filter for simple example

Consider the following first-order linear system

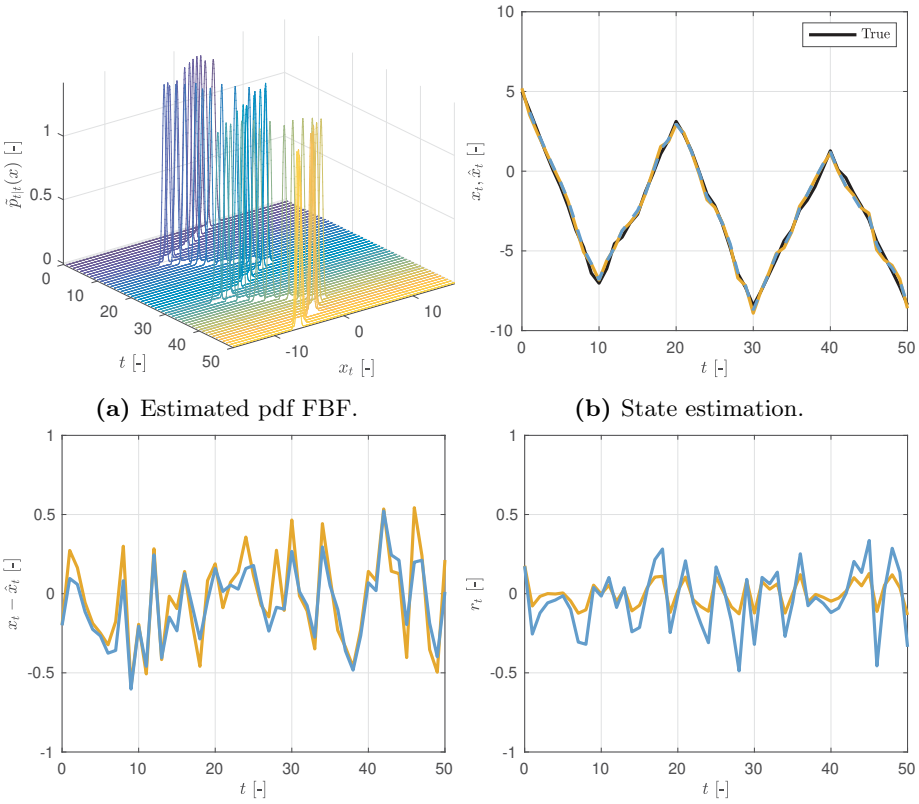
$$f(x_t, u_t, w_t) = x_t + u_t + w_t, \tag{7.42a}$$
$$g(x_t, v_t) = x_t + v_t, \tag{7.42b}$$

where $\mathcal{X} := \{x_t \in \mathbb{R} | -10 \leq x_t \leq 10\}$, $\mathcal{U} := \{u_t \in \mathbb{R} | -1 \leq u_t \leq 1\}$, and $t \in \mathbb{N}_0$. The disturbances and measurement noises are independent, zero mean, and distributed according to a Gaussian: $w_t \sim \mathcal{N}(0, 0.1)$ and $v_t \sim \mathcal{N}(0, 0.1)$. The Gaussian distributions are truncated and scaled such that $w_t \in [-5, 5]$ and $v_t \in [-5, 5]$. We set $\hat{x}_0 = 0$, and $\mathbb{E}[(x_0 - \hat{x}_0)^2] = 3$. The Fourier series coefficients that characterize the output equations $\tilde{c}_y[y_t; k]$ are obtained by sampling the pdf's of the output equations $p_y(y_t; x_t)$ for several values of $x_t \in \mathcal{X}$, and computing the Fourier series coefficients of these samples. The factor that determines the initial number of Fourier series coefficients is set to $m_{0|-1} = 1600$, and $m_y = m_{0|-1}$.

Figure 7.1a shows how the approximated pdfs $\tilde{p}_{t|t}(x)$ vary over time t. Figure 7.1b compares $\hat{x}_t = \mathbb{E}[x_t]$, computed from these pdfs, with the actual state x_t in. The estimation closely follows the actual state. For comparison, the state estimates and residual error of both the Kalman filter as well as the Fourier basis method are visualized in Figure 7.1c. The performance is assessed based on the residual errors, similar to [1]. The residual errors are calculated according to $r_t = y_t - \mathbb{E}[g(\hat{x}_t, 0)]$, and shown in Figure 7.1d. From Figure 7.1, it can be concluded that the Fourier basis functions method has a similar performance to the (optimal) Kalman filter benchmark.

7.4.2 Application to electron microscopy

Part of the tuning of an electron microscope is determining the level of aberrations (modeled as states), which are coupled to features of a ronchigram (outputs) [100]. It is not possible to estimate the level of aberrations from a single ronchigram. In general multiple hypotheses on this level should be tracked. In this numerical example, two types of aberrations are considered: focus and astigmatism (modelled x_1 and x_2, respectively). Those two aberrations are linked to two outputs (y_1 and y_2) by non-linear output equations. The control inputs (u_1 and u_2) act on each aberration directly without couplings, while also each aberration is completely decoupled, thus the system has integrator dynamics. This leads to a simplified model of the electron microscope calibration problem, which considers the problem



(a) Estimated pdf FBF. **(b)** State estimation.

(c) State estimation error. **(d)** Residual error.

Figure 7.1. Comparison of the Fourier basis functions (FBF) method (—) with the Kalman filter (—) for the first example.

of inferring the state from features, and given by

$$f_1(\cdot) = x_{1,t} + u_{1,t} + w_{1,t}, \tag{7.43a}$$
$$f_2(\cdot) = x_{2,t} + u_{2,t} + w_{2,t}, \tag{7.43b}$$
$$g_1(\cdot) = \alpha_1 x_{1,t}^2 + \alpha_2 x_{2,t}^2 + v_{1,t}, \tag{7.43c}$$
$$g_2(\cdot) = \frac{\alpha_3 x_{1,t}}{\alpha_4 + \alpha_5 (x_{2,t}^2 - x_{1,t}^2)^2} + v_{2,t}, \tag{7.43d}$$

where $\mathcal{X} := \{x_t \in \mathbb{R}^2 | [-5, 5] \times [-5, 0]\}$, $\mathcal{U} := \{u_t \in \mathbb{R}^2 | [-1, 1]^2\}$, $t \in \mathbb{N}_0$, and α_i with $i \in \{1, .., 5\}$ are constants. This model was obtained from system identification from available data; the details of the identification procedure are omitted here. The disturbances are independent and uniformly distributed in

the sets: $w_{i,t} \in [-0.2, 0.2]$, $i \in \{1, 2\}$. The measurement noises are independent, zero mean and distributed according to a Gaussian with a variance of 0.1: $v_{i,t} \sim \mathcal{N}(0, 0.1)$, $i \in \{1, 2\}$. Furthermore, they are truncated and scaled such that $v_{i,t} \in [-1, 1]$, $i \in \{1, 2\}$. The initial position estimate $\tilde{p}_{0,-1}(x)$ is considered as a uniform distribution with support $\mathcal{X}$. In the simulations, the control inputs are selected such that the states do not leave set $\mathcal{X}$. Both $\tilde{p}_{0|-1}(x)$ and $\tilde{p}(y; x)$ are sampled on a grid of 125×75 Fourier series coefficients.

The evolution of the approximated pdf is visualized in Figure 7.2 for the first 3 timesteps. The figure shows clearly the development from an initial uniform distribution towards a small region with high probability, indicating the certainty in the estimate improves over time. A more detailed figure on the quality of the estimation is given in Figure 7.3, where a comparison is made with the Unscented Kalman filter (UKF), as proposed in [55]. Since the UKF cannot deal with uniform distributions, the disturbances are estimated by $\tilde{w}_{i,t} \sim \mathcal{N}(0, 0.15)$, $i \in \{1, 2\}$, the initial state estimate is set on $\hat{x}_0 = [0, -2.5]^\top$, and

$$P_0 = \mathbb{E}\left[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^\top\right] = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix}. \tag{7.44}$$

The Fourier basis function method's estimate converges faster to the true state than the UKF. In addition to the faster convergence, the proposed method has a better tracking performance, since it does not have the requirement of Gaussian noise statistics, as the UKF has. The computation time of the UKF on the other hand is less than the Fourier basis function method. Furthermore, Figure 7.3d shows clearly that the residual error does not grow unboundedly.

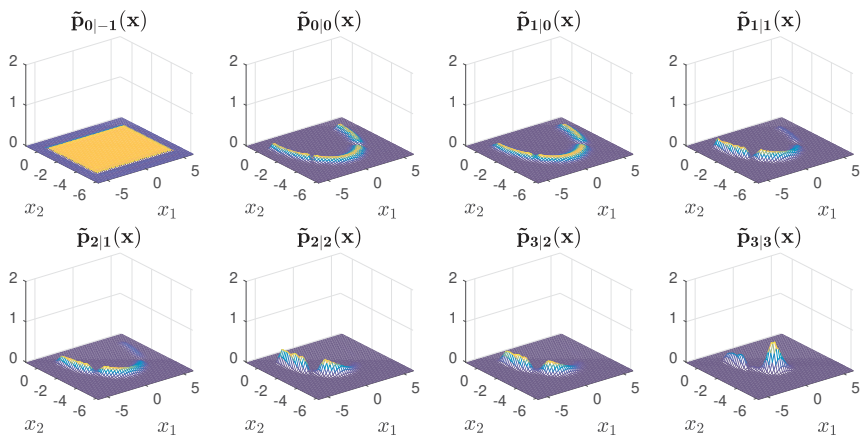
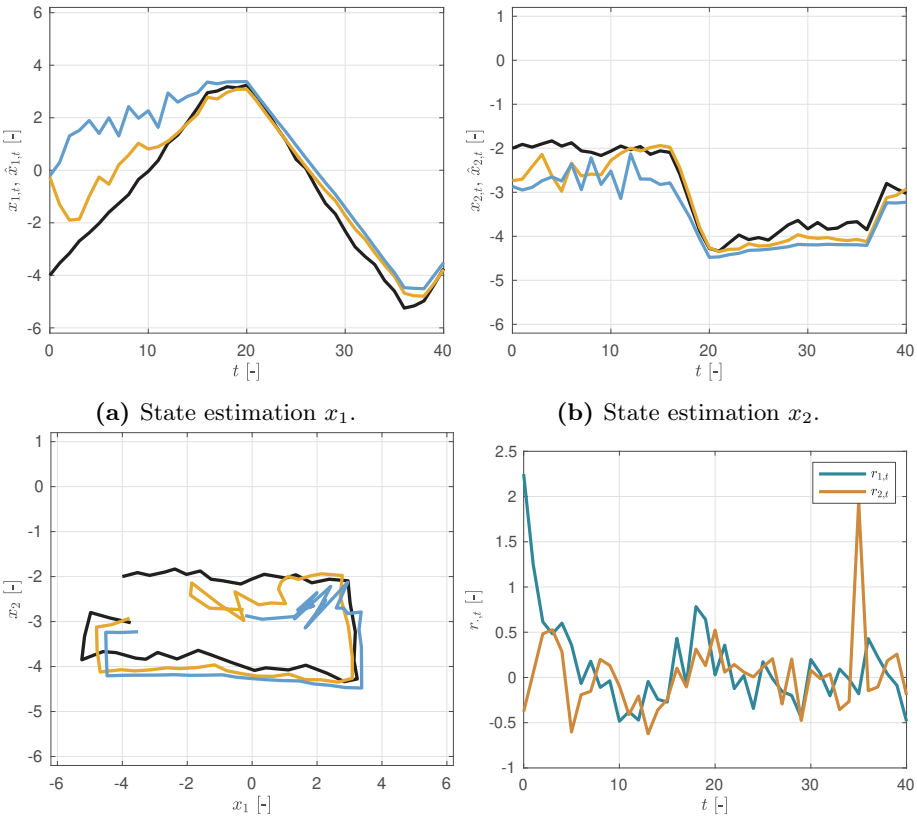


Figure 7.2. Evolution of $\tilde{p}_{t|t}(x)$ over time.



(a) State estimation x_1.

(b) State estimation x_2.

(c) State trajectory.

(d) Residual error FBF.

Figure 7.3. Comparison of the true state (—) with the Fourier basis functions (FBF) method (—) and the Unscented Kalman filter (—) of the second example.

7.5 Conclusions and future work

In this chapter, *exact* Bayesian estimation is pursued for the class of systems with linear dynamics and arbitrary (non-linear) output equations in the frequency domain through the Fourier transform, using the countable number of Fourier coefficients. This perspective naturally leads to an *approximate* method, where high frequencies are disregarded to obtain a finite number of Fourier coefficients. The approximate method is computationally tractable, since the number of coefficients grows only linearly with time. Future research directions include analyzing the estimation error, in order to give error bounds for the approximate method.

7.6 Appendix

7.6.1 Justification for the expression of B_1

Let us start by considering $n_x = 1$. We can derive the expression with frequency domain concepts and then double check with time.

We have

$$\begin{aligned} p_{t|t}(x) &\overset{\mathscr{F}}{\longleftrightarrow} P_{t|t}(\omega) \\ \sum_{k=-\infty}^{\infty} c_{t|t}[k] e^{-j\frac{2\pi k x}{L}} &\overset{\mathscr{F}}{\longleftrightarrow} 2\pi \sum_{k=-\infty}^{\infty} c_{t|t}[k]\delta(\omega - 2\pi/Lk) \\ \mathbf{1}_{[\underline{L},\bar{L}]}(x) &\overset{\mathscr{F}}{\longleftrightarrow} \frac{2\sin(\omega L/2)}{\omega} e^{-j(\omega(\bar{L}+\underline{L})/2)} \end{aligned} \tag{7.45}$$

where $c_{t|t}[k]$ are the Fourier coefficients of the periodic replication of $P_{t|t}$ with period L so that $\mathbf{1}_{[\underline{L},\bar{L}]}(x)\sum_{\underline{k}=-\infty}^{\infty} c_{t|t}[k]e^{-j\frac{2\pi k}{L}x} = p_{t|t}(x)$. A key remark is that we can assume always that $\bar{L} = -\underline{L}$. This is because now we are talking about periodic replications of $p_{t|t}$ so we can take any period we like. Due to the product property of the Fourier transforms ($f(x)g(x) \overset{\mathscr{F}}{\longleftrightarrow} \frac{1}{2\pi}F(j\omega) * G(j\omega)$) we get

$$\begin{aligned} &\mathbf{1}_{[\underline{L},\bar{L}]}(x) \sum_{k=-\infty}^{\infty} c_{t|t}[k] e^{-j\frac{2\pi k}{L}x} \overset{\mathscr{F}}{\longleftrightarrow} \\ &\sum_{k=-\infty}^{\infty} c_{t|t}[k] \frac{2\sin((\omega - 2\pi/Lk)L/2)}{(\omega - 2\pi/Lk)} e^{-j(\omega - 2\pi/Lk)(\bar{L}+\underline{L})/2} \end{aligned} \tag{7.46}$$

so that

$$B_1 = \frac{2\sin((\omega - 2\pi/Lk)L/2)}{(\omega - 2\pi/Lk)} e^{-j(\omega - 2\pi/Lk)(\bar{L}+\underline{L})/2}. \tag{7.47}$$

Let us now move to the case where $n_x > 1$. We have

$$\begin{aligned} p_{t|t}(x) &\overset{\mathscr{F}}{\longleftrightarrow} P_{t|t}(\omega) \\ \sum_{k\in\mathcal{K}} c_{t|t}[k] e^{-j2\pi k./L.x} &\overset{\mathscr{F}}{\longleftrightarrow} (2\pi)^{n_x} \sum_{k\in\mathcal{K}} c_{t|t}[k]\delta(\omega - 2\pi k./L) \\ \prod_{i=1}^{n_x} \mathbf{1}_{[\underline{L}_i,\bar{L}_i]}(x) &\overset{\mathscr{F}}{\longleftrightarrow} \prod_{i=1}^{n_x} \frac{2\sin(\omega_i L_i/2)}{\omega_i} e^{-j\omega_i(\bar{L}_i+\underline{L}_i)/2} \end{aligned} \tag{7.48}$$

where $c_{t|t}[k]$ are the Fourier coefficients of the periodic replication of $P_{t|t}$ with period L so that $\Pi_{i=1}^{n_x}\mathbf{1}_{[\underline{L}_i,\bar{L}_i]}(x)\sum_k c_{t|t}[k]e^{-j2\pi k./L\cdot x} = p_{t|t}(x)$. Due to the product

property of the Fourier transforms ($f(x)g(x) \overset{\mathscr{F}}{\longleftrightarrow} \frac{1}{2\pi}F(j\omega) * G(j\omega)$) we get

$$\begin{aligned}&\prod_{i=1}^{n_x} \mathbf{1}_{[\underline{L}_i, \bar{L}_i]}(x) \sum_k c_{t|t}[k] e^{-j2\pi k_\cdot / L \cdot x} \overset{\mathscr{F}}{\longleftrightarrow} \\ &\sum_{k \in \mathcal{K}} c_{t|t}[k] \prod_{i=1}^{n_x} \frac{2\sin((\omega_i - 2\pi k_i/L_i)L_i/2)}{(\omega_i - 2\pi k_i/L_i)} e^{-j(\omega_i - 2\pi k_i/L_i)(\bar{L}_i + \underline{L}_i)/2}\end{aligned} \tag{7.49}$$

so that

$$B_1 = \prod_{i=1}^{n_x} \frac{2\sin((\omega_i - 2\pi k_i/L_i)L_i/2)}{(\omega_i - 2\pi k_i/L_i)} e^{-j(\omega_i - 2\pi k_i/L_i)(\bar{L}_i + \underline{L}_i)/2}. \tag{7.50}$$

V

Closing

Chapter 8

Conclusions and Recommendations

8.1 Conclusions

In this thesis, novel methods and algorithms have been developed to apply policy iteration to settings with limited data and partial information. These developments are grounded in Bayesian inference and address key challenges arising from partial observability, uncertainty in system dynamics, and the high cost of data collection in practical control problems. By combining advances in output feedback policy iteration, data-efficient policy evaluation, and state estimation, the thesis aimed to broaden the theoretical foundations and practical applicability of reinforcement learning-based control methods, aligned with the main objective of this thesis, as stated in Section 1.3:

> *Provide new policy iteration methods for settings with limited data and partial information, while providing convergence guarantees and realising data efficiency.*

The contributions of this thesis, developed to meet the research objective above, are organized into three categories:

1. *Development of policy iteration methods for output feedback systems (Part II – Chapters 2 and 3 – Contributions I and II).*
2. *Development of data-efficient policy evaluation methods (Part III – Chapters 4 and 5 – Contributions III and IV).*
3. *Development of state estimation methods (Part IV – Chapters 6 and 7 – Contributions V and VI).*

The main conclusions corresponding to each contribution of this thesis, as presented in Section 1.3, are given in the following sections.

8.1.1 Policy iteration for output feedback systems

Chapters 2 and 3 in Part II of this thesis focused on extending policy iteration to output feedback settings, where the full system state is not accessible for direct use in the control policy. Chapter 2 introduced a framework for discrete-state, finite-horizon systems, developing an output feedback policy iteration scheme that explicitly accounts for the non-Markovian nature of the system. Convergence guarantees were provided under standard assumptions, demonstrating that the proposed method converges monotonically to a locally optimal policy. Beyond monotonic convergence, we characterized the computational implications of different schedules and identified a unique (up to cyclic shift) schedule with minimal period that optimally balances forward and backward propagation of the state distribution and the state-action function.

Chapter 3 generalized these ideas to continuous-state systems, considering both finite-horizon and linear-quadratic settings. The proposed method was applied to finite-horizon linear-quadratic systems, enabling practical computation of locally optimal policies. The chapter also developed an infinite-horizon output feedback approach for linear-quadratic systems. Convergence of this approach was established for scalar systems. Furthermore, there was strong numerical evidence of convergence and optimality in higher-dimensional settings under mild assumptions.

These contributions collectively extend the theoretical foundations of policy iteration to partially observed systems and provide practical algorithms for both finite and infinite-horizon control problems, forming the core basis for the subsequent developments in data-efficient evaluation and state estimation.

8.1.2 Data-efficient policy evaluation

In Part III of this thesis, frameworks were developed to improve the data efficiency of policy evaluation within policy iteration. Chapter 4 introduced a Bayesian extension of least-squares temporal-difference learning through the proposed MAP-LSTD algorithm, which incorporates prior information on system dynamics into the cost-function estimation process. By interpreting policy evaluation as a maximum a posteriori estimation problem, this approach improves numerical conditioning, enables principled regularization, and yields recursive update rules suitable for online implementation. Theoretical results established that this method was equivalent to first estimating a model using maximum a posteriori estimation, and subsequently computing the cost function using model-based policy evaluation. Numerical examples demonstrated substantial gains in convergence speed and variance when compared to classical LSTD in low-data regimes.

Chapter 5 analysed the applicability of expected eligibility traces in LSTD. It was shown that the use of theoretical expected eligibility traces collapses to LSTD(0), independently of the trace parameter, whereas combining expected and ordinary traces through mixed eligibility traces was equivalent to standard LSTD(λ') with an effective parameter $\lambda' = \eta\lambda$. These equivalence results provide conceptual insight into how one-step TD, Monte-Carlo returns, and trace-based methods are related, and clarify how the choice of λ' governs the bias–variance trade-off in data-limited regimes. Another interpretation was that expected eligibility traces can be seen as a mechanism for extending the data efficiency of LSTD to nonlinear feature settings in which classical LSTD cannot be applied. Moreover, the methods LSET(λ) and LSET(η,λ) were presented, using the empirical mean of the eligibility traces to obtain the expected value. The empirical investigations on benchmark control problems further illustrated these relationships.

Collectively, the contributions in this part established a coherent theoretical and algorithmic foundation for data-efficient policy evaluation, which is essential for enabling policy iteration in practical scenarios where data acquisition is costly.

8.1.3 State estimation

Part IV in this thesis focused on state estimation for output feedback systems. Chapter 6 investigated pose-estimation problems using two Bayesian strategies. First, an active likelihood-estimation scheme based on Bayesian optimization was proposed to efficiently explore observation models when explicit sensor likelihoods are unavailable. Second, a deep-learning-based Bayes filter employing mixture models was developed to approximate belief updates directly from data. Together, the proposed methods form a scalable and adaptable framework for probabilistic state estimation. By combining Bayesian filtering with active likelihood estimation and deep-learning-based prediction models, the framework balances computational efficiency with estimation accuracy and provides a practical solution for industrial automation scenarios involving vision-based pose estimation. Simulation and experimental results demonstrated that both approaches can accurately and rapidly infer the object's yaw, while effectively bridging the gap between synthetic and real imagery.

Chapter 7 pursued Bayesian state estimation for systems with linear dynamics and arbitrary nonlinear output equations by reformulating the filtering problem in the frequency domain through Fourier transforms. This representation enables an exact propagation of probability densities using a countable set of Fourier coefficients, while naturally motivating an approximate variant in which high-frequency components are truncated to yield a finite-dimensional belief representation. The resulting approximation is computationally tractable, as the number of retained coefficients grows only linearly with time.

Collectively, the state-estimation methods developed in this thesis form building blocks for problems in which output feedback control alone is insufficient, and reliable state estimation becomes essential.

8.2 Recommendations

The developments in the field of policy iteration for output feedback systems have led to the following insights and recommendations for future research.

Extension of the proposed policy iteration methods to infinite-horizon setting: The policy iteration frameworks for memoryless output feedback control developed in Part II can be extended to infinite-horizon formulations. In Chapter 3, a first outline of the infinite-horizon setting is given; however, the proof of monotonic convergence is currently limited to scalar systems. Extending these results would significantly broaden the applicability of the proposed methods to realistic control systems. A potential challenge (especially for systems with finite state and action spaces) is that the method may converge to a periodic policy rather than a stationary one, which would require additional analysis to ensure practical implementability.

Approximate policy iteration for large-scale POMDPs: To handle problems involving high-dimensional systems, most notably those arising from continuous state and input spaces, future work should investigate approximate variants of the proposed output feedback policy iteration schemes. In particular, future research can explore the use of function-approximation architectures, such as linear parametrisations, neural networks, or kernel-based representations, to scale the methods to large or continuous state and observation spaces with nonlinear system dynamics.

Integration of Bayesian policy evaluation with policy improvement: While MAP-LSTD provides a principled Bayesian approach to value-function estimation, further research is needed to embed these techniques fully within a closed-loop policy iteration procedure, allowing uncertainty-aware evaluation and improvement to be performed jointly. In particular, integrating posterior uncertainty into the policy evaluation step could enable more robust decision-making under partial observability and limited data.

Generalization of MAP-LSTD to broader learning settings: Promising directions include extending MAP-LSTD to semi-gradient methods, nonlinear function approximation, and off-policy learning scenarios, thereby increasing its relevance for modern reinforcement-learning pipelines. Such generalizations can enable the method to handle high-dimensional and continuous state-action spaces while maintaining principled uncertainty estimates. Investigating stability, convergence properties, and computational scalability in these broader settings will be

crucial to ensure practical applicability in real-world reinforcement-learning tasks.

Scalability and generalizability in learned state estimators: The pose-estimation frameworks based on Bayesian optimization and deep learning should be further studied with respect to both scalability and generalizability, particularly in industrial applications where sensing, inference, and control must be tightly integrated. Future work can investigate efficient network architectures and approximate inference techniques to maintain real-time performance while extending applicability across different environments and operational conditions. Moreover, systematic evaluation of trade-offs between computational cost, estimation accuracy, and robustness under varying disturbances would provide practical guidelines for deploying learned state estimators in diverse, safety-critical, or resource-constrained settings.

Theoretical error analysis for Fourier-based Bayesian filters: For the approximate Fourier-domain filtering approach in Chapter 7, a rigorous analysis of estimation errors and truncation effects is required in order to derive performance guarantees and guidelines for selecting the number of retained Fourier coefficients. Such an analysis would clarify the trade-offs between computational efficiency and estimation accuracy, providing principled criteria for practical implementations.

Coupling state estimation with output feedback policy iteration: A key long-term research direction is the integration of the state-estimation methods developed in Part IV with the policy iteration frameworks of Parts II and III. Such a coupling would enable fully data-driven output feedback control schemes that jointly learn state beliefs, evaluate policies, and improve control performance under partial observability. Future work could explore unified algorithms that propagate uncertainty through the policy evaluation and improvement steps, investigate stability and convergence in the combined setting, and assess scalability to high-dimensional or continuous systems, ultimately bridging the gap between learning-based estimation and control in practical, real-world applications.

Bibliography

[1] Alspach, D. and Sorenson, H. (1972). Nonlinear Bayesian estimation using Gaussian sum approximations. *IEEE Transactions on Automatic Control*, 17(4):439–448.

[2] Anderson, B. and Moore, J. (1979). *Optimal Filtering*. Information and System Sciences series. Prentice-Hall.

[3] Anderson, T. W. and Goodman, L. A. (1957). Statistical inference about Markov chains. *The Annals of Mathematical Statistics*, 28(1):89–110.

[4] Arcieri, G., Hoelzl, C., Schwery, O., Straub, D., Papakonstantinou, K. G., and Chatzi, E. (2024). POMDP inference and robust solution via deep reinforcement learning: an application to railway optimal maintenance. *Machine Learning*, 113(10):7967–7995.

[5] Athans, M. and Chang, C. (1976). Adaptive estimation and parameter identification using multiple model estimation algorithm. Technical Report 1976-28, Lincoln Lab, Massachusetts Institute of Technology.

[6] Azizzadenesheli, K., Lazaric, A., and Anandkumar, A. (2016a). Open problem: Approximate planning of POMDPs in the class of memoryless policies. In *JMLR: Workshop and Conference Proceedings*, volume 49, pages 1–4.

[7] Azizzadenesheli, K., Lazaric, A., and Anandkumar, A. (2016b). Reinforcement learning of POMDPs using spectral methods. In *JMLR: Workshop and Conference Proceedings*, volume 49, pages 1–64.

[8] Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846.

[9] Baxter, J. and Bartlett, P. L. (2000). Reinforcement learning in POMDPs via direct gradient ascent. In *Proceedings of the 17th International Conference on Machine Learning*, pages 41–48.

[10] Bertsekas, D. P. (2011). Approximate policy iteration: A survey and some new methods. *Journal of Control Theory and Applications*, 9(3):310–335.

[11] Bertsekas, D. P. (2018). *Dynamic Programming and Optimal Control*. Athena Scientific, 4th edition.

[12] Bertsekas, D. P. and Yu, H. (2009). Projected equation methods for approximate solution of large linear systems. *Journal of Computational and Applied Mathematics*, 227(1):27–50.

[13] Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256.

[14] Blondel, V. and Tsitsiklis, J. N. (1997). NP-hardness of some linear control design problems. *SIAM Journal on Control and Optimization*, 35(6):2118–2127.

[15] Boyan, J. A. (2002). Technical update: Least-squares temporal difference learning. *Machine Learning*, 49:233–246.

[16] Bradtke, S. J. and Barto, A. G. (1996). Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22:33–57.

[17] Brunn, D., Sawo, F., and Hanebeck, U. D. (2006). Nonlinear multidimensional Bayesian estimation with Fourier densities. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 1303–1308.

[18] Cai, Y., Liu, X., Oikonomou, A., and Zhang, K. (2024). Provable partially observable reinforcement learning with privileged information. In *Advances in Neural Information Processing Systems*, volume 37, pages 63790–63857.

[19] Camacho, A., Chen, O., Sanner, S., and McIlraith, S. A. (2017). Decision-making with non-Markovian rewards: From LTL to automata-based reward shaping. In *Proceedings of the Multi-disciplinary Conference on Reinforcement Learning and Decision Making*, pages 279–283.

[20] Cohen, V. and Parmentier, A. (2023). Future memories are not needed for large classes of POMDPs. *Operations Research Letters*, 51(3):270–277.

[21] Coleman, D., Şucan, I. A., Chitta, S., and Correll, N. (2014). Reducing the barrier to entry of complex robotic software: a MoveIt! case study. *Journal of Software Engineering for Robotics*, 5(1):3–16.

[22] Şucan, I. A., Moll, M., and Kavraki, L. E. (2012). The open motion planning library. *IEEE Robotics & Automation Magazine*, 19(4):72–82.

[23] Dallaire, P., Besse, C., Ross, S., and Chaib-draa, B. (2009). Bayesian reinforcement learning in continuous POMDPs with gaussian processes. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2604–2609. IEEE.

[24] Dann, C., Neumann, G., and Peters, J. (2014). Policy evaluation with temporal differences: A survey and comparison. *Journal of Machine Learning Research*, 15(24):809–883.

[25] Dayan, P. (1993). Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624.

[26] Degrave, J., Felici, F., Buchli, J., Neunert, M., Tracey, B., Carpanese, F., Ewalds, T., Hafner, R., Abdolmaleki, A., de Las Casas, D., et al. (2022). Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419.

[27] Deisenroth, M. P. (2010). *Efficient Reinforcement Learning using Gaussian Processes*. PhD thesis, Karlsruhe Institute of Technology.

[28] Demir, O. and Özbay, H. (2020). Static output feedback stabilization of discrete time linear time invariant systems based on approximate dynamic programming. *Transactions of the Institute of Measurement and Control*, 42(16):3168–3182.

[29] Duan, J., Li, J., Chen, X., Zhao, K., Li, S. E., and Zhao, L. (2023). Optimization landscape of policy gradient methods for discrete-time static output feedback. *IEEE Transactions on Cybernetics*, 54(6):3588–3601.

[30] Duff, M. (2002). *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. PhD thesis, University of Massachusetts Amherst.

[31] Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., and Hester, T. (2021). Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9):2419–2468.

[32] Ermer, C. and Vandelinde, V. (1973). Output feedback gains for a linear-discrete stochastic control problem. *IEEE Transactions on Automatic Control*, 18(2):154–157.

[33] Fox, V., Hightower, J., Liao, L., Schulz, D., and Borriello, G. (2003). Bayesian filtering for location estimation. *IEEE Pervasive Computing*, 2(3):24–33.

[34] Gao, Q., Liu, J., Ju, Z., and Zhang, X. (2019). Dual-hand detection for human-robot interaction by a parallel network based on hand detection and body pose estimation. *IEEE Transactions on Industrial Electronics*, 66(12):9663–9672.

[35] Garnett, R. (2023). *Bayesian Optimization*. Cambridge University Press.

[36] Gauvain, J.-L. and Lee, C.-H. (1994). Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Transactions on Speech and Audio Processing*, 2(2):291–298.

[37] Geramifard, A., Bowling, M., Zinkevich, M., and Sutton, R. S. (2006). iLSTD: Eligibility traces and convergence analysis. In *Advances in Neural Information Processing Systems*, volume 19, pages 441–448.

[38] Ghavamzadeh, M., Mannor, S., Pineau, J., and Tamar, A. (2015). Bayesian reinforcement learning: A survey. *Foundations and Trends in Machine Learning*, 8(5-6):359–483.

[39] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[40] Gordon, N., Salmond, D., and Smith, A. (1993). Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140(2):107–113.

[41] Guez, A., Silver, D., and Dayan, P. (2012). Efficient Bayes-adaptive reinforcement learning using sample-based search. In *Advances in Neural Information Processing Systems*, volume 25, pages 1025–1033.

[42] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1861–1870.

[43] Hansen, E. A. (1997). An improved policy iteration algorithm for partially observable MDPs. In *Advances in Neural Information Processing Systems*, volume 11, pages 1015–1021.

[44] Hansen, E. A. (1998). Solving POMDPs by searching in policy space. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, pages 211–219.

[45] Hashemi, N. S., Aghdam, R., Ghiasi, A., and Fatemi, P. (2016). Template matching advances and applications in image analysis. *arXiv:1610.07231*.

[46] Hawkes, P. W. and Spence, J. C. H. (2007). *Science of Microscopy*. Springer.

[47] Hoerger, M., Kurniawati, H., Kroese, D., and Ye, N. (2024). A surprisingly simple continuous-action POMDP solver: Lazy cross-entropy search over policy trees. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(18):20134–20142.

[48] Hogg, R. V., McKean, J. W., and Craig, A. T. (2019). *Introduction to Mathematical Statistics*. Pearson, 8th edition.

[49] Howard, R. A. (1960). *Dynamic programming and markov processes*. John Wiley.

[50] Hu, B., Zhang, K., Li, N., Mesbahi, M., Fazel, M., and Başar, T. (2023). Toward a theoretical foundation of policy optimization for learning control policies. *Annual Review of Control, Robotics, and Autonomous Systems*, 6(1):123–158.

[51] Ishida, S. and Henriques, J. F. (2024). SOAP-RL: Sequential option advantage propagation for reinforcement learning in POMDP environments. *arXiv:2407.18913*.

[52] Jaakkola, T., Singh, S. P., and Jordan, M. I. (1994). Reinforcement learning algorithm for partially observable Markov decision problems. In *Advances in Neural Information Processing Systems*, volume 7, pages 345–352.

[53] Jasra, A., Singh, S. S., Martin, J. S., and McCoy, E. (2012). Filtering via approximate Bayesian computation. *Statistics and Computing*, 22:1223–1237.

[54] Jin, C., Kakade, S., Krishnamurthy, A., and Liu, Q. (2020). Sample-efficient reinforcement learning of undercomplete POMDPs. In *Advances in Neural Information Processing Systems*, volume 33, pages 18530–18539.

[55] Julier, S. J. and Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. In *Signal Processing, Sensor Fusion, and Target Recognition VI*, volume 3068, pages 182–193.

[56] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45.

[57] Konda, V. and Tsitsiklis, J. (1999). Actor-critic algorithms. *Advances in Neural Information Processing Systems*, 12.

[58] Krishnamurthy, A., Agarwal, A., and Langford, J. (2016). Contextual-MDPs for PAC reinforcement learning with rich observations. In *Proceedings of the Annual Conference on Learning Theory*, volume 49, pages 193–256.

[59] Kurz, G., Gilitschenski, I., and Hanebeck, U. D. (2013). Recursive nonlinear filtering for angular data based on circular distributions. In *2013 American Control Conference*, pages 5439–5445.

[60] Kurz, G., Gilitschenski, I., and Hanebeck, U. D. (2016). Recursive Bayesian filtering in circular state spaces. *IEEE Aerospace and Electronic Systems Magazine*, 31:70–87.

[61] Lagoudakis, M. G. and Parr, R. (2003). Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149.

[62] Lamperti, F., Lavoratori, K., and Tredicine, L. (2025). From globalization to reshoring? The role of Industry 4.0 in global supply chains across Europe. *Economics of Innovation and New Technology*, pages 1–23.

[63] Lauri, M., Hsu, D., and Pajarinen, J. (2023). Partially observable Markov decision processes in robotics: A survey. *IEEE Transactions on Robotics*, 39(1):21–40.

[64] LaValle, S. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Department of Computer Science, Iowa State University.

[65] Lewis, F. L., Vrabie, D., and Syrmos, V. L. (2012). *Optimal control*. John Wiley & Sons.

[66] Li, Y., Yin, B., and Xi, H. (2011). Finding optimal memoryless policies of POMDPs under the expected average reward criterion. *European Journal of Operational Research*, 211(3):556–567.

[67] Littman, M. L. (1993). An optimization-based categorization of reinforcement learning environments. In *From Animals to Animats*, volume 2, pages 262–270.

[68] Littman, M. L. (1994). Memoryless policies: Theoretical limitations and practical results. In *From Animals to Animats*, volume 3, pages 238–245.

[69] Littman, M. L., Cassandra, A. R., and Kaelbling, L. P. (1995). Efficient dynamic-programming updates in partially observable Markov decision processes. Technical Report 95-19, Department of Computer Science, Brown University.

[70] Loch, J. and Singh, S. P. (1998). Using eligibility traces to find the best memoryless policy in partially observable Markov decision processes. In *Proceedings of the 15th International Conference on Machine Learning*, pages 323–331.

[71] Macenski, S., Foote, T., Gerkey, B., Lalancette, C., and Woodall, W. (2022). Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66).

[72] Manum, H., Skogestad, S., and Jaschke, J. (2009). Convex initialization of the H_2-optimal static output feedback problem. In *2009 American Control Conference*, pages 1724–1729. IEEE.

[73] Mardia, K. V. and Jupp, P. E. (1999). *Directional Statistics*. Wiley.

[74] Marković, I. and Petrović, I. (2012). Bearing-only tracking with a mixture of von mises distributions. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 707–712.

[75] Meuleau, N., Kim, K.-E., Kaelbling, L. P., and Cassandra, A. R. (1999). Solving POMDPs by searching the space of finite policies. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 417–426.

[76] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1928–1937.

[77] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

[78] Mostafa, E.-S. M. (2012). A conjugate gradient method for discrete-time output feedback control design. *Journal of Computational Mathematics*, pages 279–297.

[79] Müller, J. and Montufar, G. (2022). The geometry of memoryless stochastic policy optimization in infinite-horizon POMDPs. In *Proceedings of the International Conference on Learning Representations*.

[80] Murray, R. F. and Morgenstern, Y. (2010). Cue combination on the circle and the sphere. *Journal of Vision*, 10(11):15.

[81] Nedić, A. and Bertsekas, D. P. (2003). Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems*, 13(1):79–110.

[82] Ng, K. W., Tian, G.-L., and Tang, M.-L. (2011). *Dirichlet and Related Distributions: Theory, Methods and Applications*. John Wiley & Sons.

[83] Papadimitriou, C. H. and Tsitsiklis, J. N. (1987). The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450.

[84] Parr, R., Li, L., Taylor, G., Painter-Wakefield, C., and Littman, M. L. (2008). An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 752–759.

[85] Parr, T., Pezzulo, G., and Friston, K. J. (2022). *Active inference: the free energy principle in mind, brain, and behavior*. MIT Press.

[86] Peng, S., Liu, Y., Huang, Q., Zhou, X., and Bao, H. (2019). PVNet: Pixel-wise voting network for 6DoF pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4561–4570.

[87] Perkins, T. J. (2002). Reinforcement learning for POMDPs based on action values and stochastic optimization. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 199–204.

[88] Perri, V., Petrović, L. V., and Scholtes, I. (2023). Bayesian inference of transition matrices from incomplete graph data with a topological prior. *EPJ Data Science*, 12(1):1–24.

[89] Pitis, S. (2018). Source traces for temporal difference learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pages 3952–3959.

[90] Porta, J. M., Vlassis, N., Spaan, M. T., and Poupart, P. (2006). Point-based value iteration for continuous POMDPs. *Journal of Machine Learning Research*, 7(83):2329–2367.

[91] Pyo, J., Choi, J., and Kuc, T. (2024). An object-centric hierarchical pose estimation method using semantic high-definition maps for general autonomous driving. *Sensors*, 24:5191.

[92] Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press.

[93] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788.

[94] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, volume 28, pages 91–99.

[95] Ronchi, V. (1964). Forty years of history of a grating interferometer. *Applied Optics*, 3(4):437–451.

[96] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241.

[97] Ross, S., Pineau, J., Chaib-draa, B., and Kreitmann, P. (2011). A Bayesian approach for learning and planning in partially observable Markov decision processes. *Journal of Machine Learning Research*, 12(48):1729–1770.

[98] Ross, S. M. (2010). *Introduction to Probability Models*. Academic Press, 10th edition.

[99] Rozek, B., Lee, J., Kokel, H., Katz, M., and Sohrabi, S. (2024). Partially observable hierarchical reinforcement learning with AI planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 23635–23636.

[100] Sawada, H., Sannomiya, T., Hosokawa, F., Nakamichi, T., Kaneyama, T., Tomita, T., Kondo, Y., Tanaka, T., Oshima, Y., Tanishiro, Y., and Takayanagi, K. (2008). Measurement method of aberration from Ronchigram by autocorrelation function. *Ultramicroscopy*, 108(11):1467–1475.

[101] Schnitzer, N., Sung, S. H., and Hovden, R. (2019). Introduction to the Ronchigram and its calculation with Ronchigram.com. *Microscopy Today*, 27(3):12–15.

[102] Schulman, J., Moritz, P., Levine, S., Jordan, M. I., and Abbeel, P. (2015). High-dimensional continuous control using generalized advantage estimation. *arXiv:1506.02438*.

[103] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

[104] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2018a). AlphaZero: Shedding new light on chess, shogi, and Go. https://deepmind.google/blog/alphazero-shedding-new-light-on-chess-shogi-and-go/.

[105] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2018b). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144.

[106] Singh, S. P., Jaakkola, T., and Jordan, M. I. (1994). Learning without state-estimation in partially observable Markovian decision processes. In *Proceedings of the 11th International Conference on Machine Learning*, pages 284–292.

[107] Sinha, A., Geist, M., and Mahajan, A. (2024). Periodic agent-state based Q-learning for POMDPs. In *Advances in Neural Information Processing Systems*, volume 37, pages 62123–62159.

[108] Sinha, A. and Mahajan, A. (2024). Agent-state based policies in POMDPs: Beyond belief-state mdps. In *Proceedings of the 63rd IEEE Conference on Decision and Control*, pages 6722–6735.

[109] Smallwood, R. D. and Sondik, E. J. (1973). The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088.

[110] Smidl, V. and Quinn, A. (2008). Variational Bayesian filtering. *IEEE Transactions on Signal Processing*, 56(10):5020–5030.

[111] Spaan, M. T. J. (2012). Partially observable Markov decision processes. In *Reinforcement Learning: State-of-the-Art*, pages 387–414. Springer.

[112] Steckelmacher, D., Roijers, D. M., Harutyunyan, A., Vrancx, P., Plisnier, H., and Nowé, A. (2018). Reinforcement learning in POMDPs with memoryless options and option-observation initiation sets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4099–4106.

[113] Subramanian, J., Sinha, A., Seraj, R., and Mahajan, A. (2022). Approximate information state for approximate planning and reinforcement learning in partially observed systems. *Journal of Machine Learning Research*, 23(1):1629–1636.

[114] Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44.

[115] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition.

[116] Sutton, R. S., McAllester, D., Singh, S. P., and Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12.

[117] Syrmos, V. L., Abdallah, C. T., Dorato, P., and Grigoriadis, K. (1997). Static output feedback—a survey. *Automatica*, 33(2):125–137.

[118] Tanimoto, S. L. (1981). Template matching in pyramids. *Computer Graphics and Image Processing*, 16:356–369.

[119] Thermo Fisher Scientific Inc (2026). Media gallery. https://newsroom.thermofisher.com/newsroom/media-gallery/default.aspx.

[120] US ITER (2023). iter. https://www.iter.org/iter-image-galleries/technical.

[121] van Hasselt, H., Madjiheurem, S., Hessel, M., Silver, D., Barreto, A., and Borsa, D. (2021). Expected eligibility traces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 9997–10005.

[122] Vlassis, N., Littman, M. L., and Barber, D. (2012). On the computational complexity of stochastic controller optimization in POMDPs. *ACM Transactions on Computation Theory*, 4(12):1–8.

[123] Wang, C., Xu, D., Zhu, Y., Martín-Martín, R., Lu, C., Fei-Fei, L., and Savarese, S. (2019). DenseFusion: 6D object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3343–3352.

[124] Williams, J. K. and Singh, S. (1998). Experimental results on learning stochastic memoryless policies for partially observable Markov decision processes. In *Advances in Neural Information Processing Systems*, volume 11, pages 1073–1080.

[125] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256.

[126] Wu, C., Chen, L., He, Z., and Jiang, J. (2021). Pseudo-Siamese graph matching network for textureless objects' 6-D pose estimation. *IEEE Transactions on Industrial Electronics*, 69(3):2718–2727.

[127] Xiang, Y., Schmidt, T., Narayanan, V., and Fox, D. (2017). PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. *arXiv:1711.00199*.

[128] Yu, H. (2012). Least squares temporal difference methods: An analysis under general conditions. *SIAM Journal on Control and Optimization*, 50(6):3310–3343.

[129] Yu, H. and Bertsekas, D. P. (2008). On near optimality of the set of finite-state controllers for average cost POMDP. *Mathematics of Operations Research*, 33(1):1–11.

[130] Yu, J.-T. (2020). An equivalent discrete-time output feedback linear quadratic regulator theory. In *2020 7th International Conference on Control, Decision and Information Technologies*, volume 1, pages 868–873.

[131] Yu, S., Zhai, D.-H., Yin, J., and Xia, Y. (2025). Category-level 6-D object pose estimation with learnable prior embeddings for robotic grasping. *IEEE Transactions on Industrial Electronics*, 72(11):11682–11694.

[132] Zamani, Z., Sanner, S., Poupart, P., and Kersting, K. (2012). Symbolic dynamic programming for continuous state and observation POMDPs. *Advances in Neural Information Processing Systems*, 25.

[133] Zhang, H., Liang, Z., Li, C., Zhong, H., Liu, L., Zhao, C., Wang, Y., and Wu, Q. J. (2021). A practical robotic grasping method by using 6-D pose estimation with protective correction. *IEEE Transactions on Industrial Electronics*, 69(4):3876–3886.

List of publications

Peer-reviewed journal articles

- R.A.C. van Zuijlen, D.J. Antunes, "*Least-squares temporal difference with expected eligibility traces*", Machine Learning Journal, 2025.
- J.S. van Hulst, R.A.C. van Zuijlen, N. Javaheri, M. Diephuis, D.J. Antunes, W.P.M.H. Heemels, "*Calibration of Electron Microscopes Through Deep Learning and Bayesian Optimization*", IEEE Access, 2025.
- R.A.C. van Zuijlen, D.J. Antunes, "*Memoryless Policy Iteration for Episodic POMDPs*", (*submitted*).
- S.J.A. van Esch, R.A.C. van Zuijlen, D.J. Antunes, "*A Bayesian approach to pose estimation using Gaussian processes and deep learning*", (*submitted*).

Peer-reviewed articles in conference proceedings

- R.A.C. van Zuijlen, D.J. Antunes, "*Maximum A Posteriori Least-Squares Temporal Difference*", American Control Conference, 2025.
- R.A.C. van Zuijlen, J.S. van Hulst, W.P.M.H. Heemels, D.J. Antunes, "*Bayesian Estimation for Linear Systems with Nonlinear Output and General Noise Distribution using Fourier Basis Functions*", IEEE Conference on Decision and Control, 2023.
- J.S. van Hulst, R.A.C. van Zuijlen, D.J. Antunes, W.P.M.H. Heemels, "*Estimation of Dynamic Gaussian Processes*", IEEE Conference on Decision and Control, 2023.
- R.A.C. van Zuijlen, F. Kupper, F.P.T. Willems, "*Energy Management for RCCI Engines with Electrically-Assisted Turbocharging*", 10th IFAC Symposium on Advances in Automotive Control, 2022.

- R.A.C. van Zuijlen, D.J. Antunes, "*Memoryless Policy Iteration for Continuous POMDPs: Application to Static Output Feedback LQG*", (*submitted*).
- R.A.C. van Zuijlen, D.J. Antunes, "*From Trajectory-Tracking to Path-Following through Rollout*", (*in preparation*).

Non peer-reviewed abstracts

- R.A.C. van Zuijlen, D.J. Antunes, "*Expected Eligibility Traces in LSTD Policy Evaluation*", Benelux Meeting on Systems and Control, 2025.
- R.A.C. van Zuijlen, D.J. Antunes, "*Including Prior Knowledge in Least-Squares Temporal Difference Learning*", Learning in Control and Robotics, 2024.
- R.A.C. van Zuijlen, W.P.M.H. Heemels, D.J. Antunes, "*MAP-LSTD: Including Prior Knowledge in Policy Evaluation*", Benelux Meeting on Systems and Control, 2024.
- J.S. van Hulst, R.A.C. van Zuijlen, D.J. Antunes, W.P.M.H. Heemels, "*Bayesian Optimization Applied to Calibration of Electron Microscope*", Benelux Meeting on Systems and Control, 2023.

Dankwoord

Na vier mooie jaren zit het PhD-avontuur erop! Wat begon als een sprong in het diepe, niet exact wetende waar ik aan was begonnen, eindigt met verschillende publicaties, fantastische conferentiebezoeken, een schat aan opgedane kennis, waardevolle ontmoetingen en samenwerkingen, en bovenal veel schik onderweg! Hierbij wil ik iedereen die de afgelopen jaren, op welke wijze dan ook, heeft bijgedragen hieraan ontzettend bedanken! Daarnaast wil ik graag een aantal mensen in het bijzonder bedanken.

First of all, Duarte, what would this thesis have been without you? I really enjoyed collaborating with you the last couple of years. Your dedicating, passion, and creativity helped me a lot to keep on pushing! And that was exactly what was needed, considering all the setbacks we had. A huge number of research topics were discussed during the meetings, and almost every single time we reached the same conclusion after a couple of weeks; it was not worthwhile to continue. Nevertheless, among all the topics, we still managed to find a couple of interesting ones to continue that led to this thesis, of which I'm really proud of! And in between all the research discussions, those moments where we briefly stepped away from research to talk about football, Benfica and Ajax, made the journey even more enjoyable.

Maurice, ik ben je zeer dankbaar voor je begeleiding en betrokkenheid in het begin van mijn promotietraject. Tijdens ons allereerste gesprek, toen ik nog aan het oriënteren was of ik überhaupt hier aan zou gaan beginnen, gaf jij mij direct heel veel vertrouwen dat het wel goed zou komen. Dat vertrouwen heeft er in grote mate aan bijgedragen dat ik de stap heb gezet om aan dit promotietraject te beginnen. Je benadrukte vanaf het begin dat ik vooral moest doen wat ik zelf interessant vond, iets wat voor mij van grote waarde is geweest. Dat zie ik ook terug in deze thesis, die grotendeels bestaat uit onderwerpen die ik zelf het meest interessant vind, wat het des te jammerder maakt dat dit avontuur nu "al" voorbij is.

Without the members of my Ph.D. committee, obtaining my doctorate and finishing this thesis would not have been possible. For this reason, I would like to thank Raphael Jungers, Bart de Schutter, Nirvana Meratnia, and Maarten

Schoukens for reading and reviewing my thesis and taking the time to participate in the committee.

I would like to thank everyone who was involved in the ASIMOV-project. The project led to interesting meetings and discussions with all the partners. In particular, I would like to thank ThermoFisher Scientific and especially Maurits and Narges for their time and effort to explain how the electron microscope worked and what challenges they faced.

Nancy en Lindsey, ik wil jullie hartelijk bedanken voor alle praktische ondersteuning gedurende mijn promotietraject. Jullie hulp bij het regelen van allerlei zaken, waaronder de organisatie van de ASIMOV General Assembly, heb ik zeer gewaardeerd.

Frank, tot de laatste fase van de Master had ik nooit overwogen om aan een PhD te beginnen. Jouw vertrouwen heeft dat balletje aan het rollen, en daar ben ik je nog altijd dankbaar voor.

Next, I'd like to thank everyone with whom I've collaborated with in the Dynamic Programming and Reinforcement Learning course. In particular, I would like to thank Menno for the pleasant collaboration. Besides that, I'd like to thank all the students who I guided. Stijn en WeFabricate, bedankt voor de fijne samenwerking, wat uiteindelijk tot een van de hoofdstukken in deze thesis heeft geleid!

De afgelopen jaren heb ik een heleboel leuke en gezellige collega's leren kennen, wat tot ontelbaar veel leerzame, memorabele momenten en herinneringen heeft gezorgd. Die momenten varieerden van gesprekken bij de koffieautomaat tot ervaringen aan de andere kant van de wereld. Tijdens conferenties heb ik niet alleen veel kunnen leren, maar ook bijzondere ervaringen opgedaan, zowel ver weg als dichter bij huis. Zo denk ik met veel plezier terug aan de reizen naar Singapore en Denver, maar ook aan de Benelux-meetings in Blankenberge en Egmond aan Zee, waar wetenschap en gezelligheid moeiteloos samengingen. Naast deze meer inhoudelijke momenten waren juist ook de informele momenten van grote waarde, met ruimte voor ontspanning en spontane activiteiten. Zelfs de rondjes wandelen (lees: rennen) tijdens de lunchpauze of de origami-momenten zijn dingen waar ik met een glimlach op terugkijk.

Daarnaast wil ik in het bijzonder mijn kantoorgenoten noemen, die een grote bijdrage hebben geleverd aan een prettige en gezellige werkomgeving. Allereerst Jilles, het was bijzonder om dit traject samen te beginnen en een groot deel ervan parallel te doorlopen. Roeland, dankzij jou heb ik ook nog wat opgestoken van schaken (al blijf ik ervan overtuigd dat het pionnetje 2 plekjes vooruit toch de beste zet was destijds, ookal was het niet helemaal volgens de regeltjes). Bas, het was altijd een waardige strijd om wie 's ochtends als eerste op kantoor was om het licht aan te maken, waarna de dag rustig kon beginnen. Lars, dankzij de Worldle-momenten hebben we de hele wereld gezien middels silhouetten van

achter een computerscherm, en landen bezocht waar ik van te voren nog nooit van had gehoord. After three years in the dark basements of Gemini, we moved to the Pendulum building, where the view outside the window stretched for a few kilometers instead of just 2 meters. I was happy to share the office with you Domagoj during the last year of the PhD.

Na een werkdag in het verre Eindhoven was het altijd weer tijd huiswaarts te keren richting het mooie Winssen. Daar ging het al snel verder op de Boshof met wedstrijden, vergaderingen en natuurlijk de nodige uurtjes in de kantine. Roda 4, bedankt voor de gezelligheid die daar altijd bij hoorde, wat steevast gepaard ging met het lappen van het welbekende tientje. Of we nou winnen of verliezen, dat maakt voor de schik die we hebben niet uit, want iedereen gaat onder luid applaus naar huis. Binnen het bestuur van Roda '28, waar ik inmiddels al een paar jaar deel van uitmaak, vond ik altijd een fijne afwisseling met de dagelijkse werkzaamheden. Ik zie ernaar uit om in de toekomst samen nog mooie projecten te realiseren. Daarnaast was er in de zomer het Jeugdvakantiewerk in Winssen, een week waarin de laptop en wiskunde plaatsmaakte voor de hamer en zaag. Veel dank aan alle vrijwilligers die deze week mogelijk maken en er een gezellige tijd van maken.

Naast werk en andere activiteiten wil ik ook mijn vrienden bedanken, die altijd een belangrijke steun en afleiding zijn geweest. Koen, Ties, Lucas, Dyon en Aniek, Bart en Madelief, Charlot, Jari: met jullie heb ik veel mooie momenten gedeeld. Van spelletjesavonden en weekendjes weg tot Formule 1 kijken, beachvolleybal of gewoon gezellig samen afspreken. Mark, Tim en Shreyas, ik ben blij dat ik jullie tijdens mijn studie heb leren kennen en hoop dat we nog lang contact zullen houden. Daniek, het is altijd erg fijn om met jou in gesprek te zijn over van alles en nog wat, al dan niet onder het genot van lekker eten. Dit allemaal is voor mij altijd van grote waarde geweest.

Tot slot wil ik mijn familie, met in het bijzonder ons pap, mam en Nicole bedanken. Jullie hebben er altijd voor gezorgd dat ik alle ruimte kreeg om mij volledig op mijn studie te kunnen richten en door te blijven gaan, ook tijdens mijn promotietraject. Het was voor mij vanzelfsprekend dat ik na een werkdag weer door kon naar bijvoorbeeld een vergadering van Roda, iets wat zonder jullie steun niet mogelijk was geweest. Daarnaast hebben jullie mij altijd meegegeven dat je af moet maken waar je aan begint, een les die mij door dit traject heen heeft geholpen. Daar ben ik jullie enorm dankbaar voor!

Roy van Zuijlen
Winssen, April 2026

Curriculum Vitae

Roy Albertus Cornelis van Zuijlen was born on June 2, 1997 in Nijmegen, the Netherlands. After finishing secondary education at the Mondial College in Nijmegen, he studied Automotive at the HAN University of Applied Science in Arnhem, the Netherlands. He received the Bachelor of Science degree in 2019, after which he continued his education at the Eindhoven University of Technology in Eindhoven, the Netherlands. He received the Master of Science degree in Automotive Technology in 2021. His master's thesis, entitled "Energy Management for RCCI-Engine with E-Turbo", focused on control strategies for heavy-duty engines equipped with advanced combustion techniques in combination with an electronically assisted turbocharger, and was supervised by Frank Kupper and Frank Willems.

In February 2022, Roy started his Ph.D. research in the Control Systems Technology section at the department of Mechanical Engineering at the Eindhoven University of Technology, under supervision of Duarte Antunes and Maurice Heemels. For his Ph.D. research, he focused on policy iteration methods for settings that only provide a limited amount of data samples, and systems where only partial information of the system's state is available. This research is part of the ASIMOV project, which is (partly) supported by the Netherlands Organisation for Applied Scientific Research TNO, the Dutch Ministry of Economic Affairs and Climate, and the German Federal Ministry of Education and Research. The main results of his research are included in this thesis.