TECHNISCHE UNIVERSITÄT WIEN

AUTOMATION & CONTROL INSTITUTE
INSTITUT FÜR AUTOMATISIERUNGS- & REGELUNGSTECHNIK

# Control of pulsed lasers with high repetition rate using reinforcement learning

**DIPLOMA THESIS**

Conducted in partial fulfillment of the requirements for the degree of a

Diplom-Ingenieur (Dipl.-Ing.)

supervised by

Univ.-Prof. Dr. techn. A. Kugi
L. Tarra, MSc

submitted at the

TU Wien
Faculty of Electrical Engineering and Information Technology
Automation and Control Institute

by
Bernhard Schimkowitsch
Matriculation number 11901992

Vienna, August 2024

# Abstract

This thesis investigates the stabilization of the dynamics of two pulsed laser system classes around a desired set-point using reinforcement learning. In particular, these two systems include a regenerative amplifier and laser pulses generated with cavity dumping. The choice of the learning algorithms is limited by computation time, so tabular-based reinforcement learning algorithms are selected. The linear-quadratic regulator problem serves as a basis for the reward function. The state space is partitioned into a finite number of states for the table setting, with a higher density of points around the desired set-point. Q-value iteration, a model-based reinforcement learning algorithm, as well as Monte Carlo Control, SARSA and Q-learning, all being model-free algorithms, are examined. Simulations show that Monte Carlo Control takes very long to learn an optimal policy. SARSA and Q-learning learn much faster and achieve a better performance of the closed loop, with Q-learning exhibiting the best overall performance as measured by variances of the output and input quantity of the plant. In addition, the influence of hyperparameters on the resulting control law is analyzed. In general, off-policy methods yield better performances compared to on-policy algorithms in all cases, indicating that off-policy methods might be better suited for stabilizing a desired set-point. To benchmark the reinforcement learning approach, the controllers are compared to an adaptive nonlinear controller which is based on a first-pinciples model and thus requires a priori system knowledge. Compared to this controller, the reinforcement learning algorithms need more samples to update the policy to unknown or drifting model parameters. Moreover, control laws obtained by reinforcement learning algorithms can handle small model-plant mismatches whereas adaptive control laws are able to cope with larger deviations from the nominal model. However, reinforcement learning algorithms are able to stabilize the nominal set-point without any prior system knowledge.

# Kurzzusammenfassung

Diese Arbeit untersucht die Stabilisierung der Dynamik von zwei gepulsten Lasersystem-Klassen um einen gewünschten Arbeitspunkt mit Hilfe von Reinforcement Learning. Diese beiden Systeme umfassen einen regenerativen Verstärker und ein System, das Pulse mittels Cavity Dumpings erzeugt. Nachdem die Rechenzeit begrenzt ist, bieten sich tabellenbasierte Reinforcement Learning-Algorithmen für das bestehende Problem an. Das linear-quadratische Regelungsproblem dient als Grundlage für die Belohnungsfunktion. Um tabellenbasierte Algorithmen anzuwenden, muss der Zustandsraum in eine endliche Anzahl von Zuständen unterteilt werden, welche eine höhere Dichte an Punkten um den gewünschten Arbeitspunkt aufweisen. Q-value iteration, ein modellbasierter, sowie Monte Carlo Control, SARSA und Q-learning, allesamt modellfreie Algorithmen, werden untersucht. Simulationen zeigen, dass Monte Carlo Control sehr lange benötigt, um eine optimale Strategie zu erlernen. Im Gegensatz dazu lernen SARSA und Q-learning viel schneller. Q-learning erreicht dabei die beste Leistungsfähigkeit, gemessen an der Varianz der Eingangs- und Ausgangsgröße der Regelstrecke im geschlossenen Regelkreis. Darüber hinaus wird der Einfluss von Hyperparametern auf das gelernte Regelgesetz analysiert. In allen untersuchten Fällen weisen Off-Policy-Methoden ein besseres Ergebnis auf als On-Policy-Methoden, was darauf hindeutet, dass Erstere besser für die Stabilisierung des Arbeitspunkts geeignet sind. Um die gelernten Regelgesetze zu bewerten, werden diese mit einem adaptiven nichtlinearen Regler verglichen, welcher auf grundlegendem Modellwissen basiert und daher eine vorherige Kenntnis des Systems erfordert. Im Vergleich zu diesem Regler benötigen Regler basierend auf Reinforcement Learning länger, um ihr Regelgesetz an sich verändernde Parameter der Strecke anzupassen. Des Weiteren können Regelgesetze basierend auf Reinforcement Learning mit kleinen Abweichungen zwischen Modell und Regelstrecke umgehen. Adaptive Regelgesetze eignen sich auch für größere Modellabweichungen. Dennoch sind die Algorithmen basierend auf Reinforcement Learning in der Lage, den nominellen Arbeitspunkt ohne vorher bekannte Systeminformation zu stabilisieren.

# Contents

# List of Figures

# List of Tables

# 1 Introduction

Nowadays, pulsed lasers are deployed ubiquitously. In material processing, they are used for drilling, cutting, welding, surface engineering [1] and material deposition [2]. Especially in the field of micro- and nanofabrication, fs/ps lasers are used. Due to their small heat affected zone, high precision engineering is ensured [1, p. 33]. Pulsed lasers are also frequently used in LiDAR (Light Detection and Ranging) systems. Short, high power pulses allow for long-distance measurements up to the kilometre range as pulse power is usually above background noise [3] which surpasses the range which would be feasible with an amplitude-modulated continuous-wave approach. Ultrafast spectroscopy investigates photon-triggered processes in atoms and nanostructures, using lasers with pulse widths of femtoseconds and even below [4]. Many more applications exist, e. g., in the medical sector for skin treatment [5], so the omnipresent use of pulsed lasers motivates a deeper look into some peculiar aspects of their operation.

In the next section, a brief overview of pulse generation methods for the two laser systems relevant to this thesis will be given. Next, pulse energy fluctuations are discussed which call for some stabilization techniques. Reinforcement learning is proposed to combat these fluctuations, so an outline of recent advancements in this area is provided. Finally, a structural overview of this thesis concludes this chapter.

## 1.1 Pulsed Laser Systems

Pulsed lasers systems can be roughly categorized into two groups: self-seeded laser systems which generate pulses of their own and laser amplifiers which require an additional seed pulse [6, Chapter 4]. Techniques in the former category include Mode-locking [6, Chapter 9] for the generation of ultrashort pulses in the ps and fs regime, and gain-switching [7, Chapter 8.5] for the self-seeded case. In the second category, single- and multiple-pass pulse amplifiers can be distinguished [6, Chapter 4.1]. Nevertheless, only Q-switching and regenerative amplifiers will be considered further in this thesis. In the following, these two methods will be shortly presented.

### 1.1.1 Q-switching

Q-switching is a technique to generate nanoseconds pulses with high power by changing cavity losses [8, Chapter 13.2]. The latter can be actively switched using rotating mirrors, electro-optic modulators like Pockels cells, or acoustooptic modulators (AOM) [9, p. 1006-1007]. During the first phase of a cycle, cavity losses are high, while the active medium is pumped, enabling large population inversions. Then, cavity losses are switched to a lower value such that round-trip gains surpass the losses. Spontaneous emissions cause a rapid growth of laser oscillation, which in turn depletes the population inversion below the lasing threshold. Thus, the pulse dies out quickly, leading to short high power pulses which may reach megawatt range and usually last a few nanoseconds to some tens of nanoseconds. The pulse width is typically in the range of the photon decay time [7, p. 319]. The high inversion requires a long lifetime in the upper laser level. Thus, solid-state lasers such as Nd:YAG and $CO_2$ gas lasers, which both feature long lifetimes, are applied frequently.

Cavity Dumping [6, Chapter 8.6] can further reduce the pulse width. Essentially, cavity dumping exploits the same idea of changing cavity losses in one cycle. However, switching is done between two extreme values of 0 % and 100 % reflection. During the first phase with huge cavity losses, the population inversion is accumulated. Then, the cavity is closed, leading to the pulse generation. When switching from 100 % back to 0 %, the optical energy stored in the cavity is dumped at once, leading to much shorter pulse widths compared to regular Q-switching. In addition, the pulse width is primarily determined by the cavity length, because almost all optical energy can leave the cavity within one round-trip. Moreover, when continuously pumped, higher switching frequencies can be achieved compared to Q-switching [6, p. 530-531].

### 1.1.2 Regenerative Amplifiers

Regenerative Amplifiers (RAs) are often used in the master oscillator-pulse amplifier (MOPA) scheme where very low-energy pulses from a master pulse source are amplified [6, Chapter 4.2]. RAs are normally pumped to a inversion level that is below the lasing threshold, so a seed pulse is required to initiate laser amplification. This seed pulse then travels several round-trips through the cavity, being amplified during each round-trip. When the pulse energy reaches its maximum due to gain saturation, a switching element (typically a Pockels cell) reflects the pulse to the output. In this sense, RAs also utilize cavity dumping.

### 1.1.3 Pulse energy fluctuations

One known issue for RAs are pulse energy fluctuations, i. e., that subsequent pulses exhibit different energy levels. This can be caused by pump or seed pulse fluctuations. However, also dynamic reasons can contribute to this issue. If repetition rates increase to the order of the inverse lifetime of the upper laser level, the output energy level might show so-called period-doubling bifurcations or even deterministic chaos because subsequent pulses may couple due to the limited time for the population inversion to recover after a pulse [10].

Methods have been studied to cope with this issue. For RAs, increasing the seed pulse energy [11, 12] leads to smaller gains and thus avoids bifurcations. However, this requires an additional amplifier in front of the RA in the amplifier chain. In [13], the pump power is increased to suppress bifurcations by saturating the gain medium, which, however impairs the energy efficiency of the laser.

Similar observations regarding bifurcations and deterministic chaos were made for actively [14] and passively [15] Q-switched laser systems. Pumping power, duty cycle and repetition rate seem to mainly contribute to bifurcations [16]. Methods exist to mitigate this phenomenon. For example, in [17], a Yb:YAG laser is stabilized using fluorescence feedback control where the population inversion is measured and controlled by adapting the current of a pump diode and thus changing the pump rate.

Recently, further advancements in the field of automatic control of RAs and Q-switched lasers have been made. In [18], a discrete-time model of the pulse-to-pulse dynamic for an RA is proposed in combination with an additional AOM to adjust the input seed pulse. A linear state feedback controller is proposed to stabilize the set-point locally by modifying the AOM losses. The approach is further pursued to advise a nonlinear controller for global stabilization of the set-point in [19].

In a similar fashion, a discrete-time model for the pulse-to-pulse dynamics of a Nd:YAG repetitively Q-switched laser is proposed in [20]. This model also incorporates stochastic noise terms introduced by spontaneous emission. In [21], this model is then used to stabilize the output pulse energy using the nonlinear controller from [19] with the switching time from the Pockels cell as a control input. To mitigate stochastic influences from spontaneous emissions, which might significantly influence the pulse-to-pulse dynamics, prelasing is advised and a linear correction term is added to the nonlinear control law.

The latter methods rely on model information to suppress bifurcations. However, the authors in [13] show that many model parameters such as absorption and emission cross-sections are subject to substantial drift, often caused by thermal dependencies. Control laws should therefore be designed very robustly to be able to cope with these parameter uncertainties. Alternatively, if model information is uncertain, data-based approaches such as reinforcement learning can be used to determine an appropriate control law. Thus, in the next section, a brief overview of this topic will be given.

## 1.2 Reinforcement Learning

Reinforcement learning (RL) is a computational approach to finding optimal actions in a possibly unknown and stochastic environment [22, Chapter 1.6]. RL is built on the notion of states. Thus, state-space models of plants can be directly transferred to the RL framework. Other core elements of RL include a policy, which can be either resembled by a deterministic state-dependent control law or by a probability distribution for each state, and a reward function. The typical framework for RL is drawn in Figure 1.1. An agent/the controller receives state signals from the environment and advises an action to take based on the current state. In addition, the policy of the agent is updated using information of the reward gained at that state.

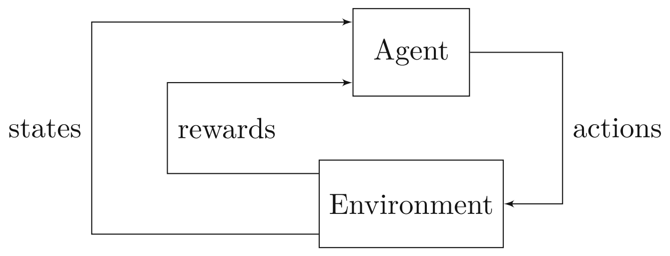

Figure 1.1: Standard RL framework [22, p. 48].

In recent years, some major advancements in the field of RL have been achieved. Deep Q Network (DQN) [23] expands the idea of Q-learning [24] to using a neural network as a function approximator which can handle larger state space dimensions if a convolutional neural network architecture is used. In [25], the policy gradient theorem is transferred to a deterministic version (deterministic policy gradient, DPG). This approach uses linear function approximators for the Q-function which are unbiased estimators and therefore frequently studied in literature. DPG shows fast learning, but is limited by the feasible state space dimension. [26] unifies the merits of DQN and DPG by harnessing the flexibility of neural networks as function approximators (deep DPG, DDPG). An issue with the deterministic policy update is the incorporation of exploration in the learning process. Exploration noise has to be artificially added to the action output. Soft Actor-Critic (SAC) [27] avoids this issue by adding an entropy term in the objective function. Thus, exploration is automatically accounted for and does not have to be manually adjusted which can lead to more efficient exploration over the state space.

Trust region policy optimization (TRPO) [28] tackles the problem of unstable policy gradient updates for large state-space dimensions by introducing a step size limitation based on the Kullback-Leibler divergence. Proximal policy optimization (PPO) [29] simplifies this constrained optimization problem for the policy update of TRPO and can therefore be implemented more easily.

While in the robotics area, RL has been used to solve many complex tasks like locomotion [30] and manipulation [31], RL starts gaining ground in the field of optics as well. In [32], the authors use Q-learning to modulate the laser power to control the laser welding quality. For the self-tuning of a fiber laser system in the presence of instability, the authors in [33] use DQN to adjust the orientation of wave plates and the fiber birefringence to achieve mode-locking. The authors in [34] use a DDPG-based RL algorithm to quickly recover mode-locking after vibration shocks for an ultrafast erbium-doped fiber laser system and train the agent to handle different temperatures.

All these methods operate in the milliseconds regime or above. It stands to question whether RL can be successfully applied to objectives where computation time is tightly restricted. Thus, this thesis covers the control of the pulse-to-pulse dynamics of pulsed laser systems using reinforcement learning. Chapter 2 investigates the pulse-to-pulse dynamics of a regenerative amplifier. After presenting the model of the pulse-to-pulse dynamics, a variety of RL algorithms, including Q-value iteration, Monte Carlo and temporal difference methods, are proposed to stabilize the set-point. These algorithms are then compared to a model-based non-linear controller in the presence of parameter uncertainties. Chapter 3 uses the same methods to control a pulsed laser system which utilizes cavity dumping. Finally, Chapter 4 wraps up the thesis by giving some final remarks.

# 2 Regenerative Amplifier

This chapter covers the control of a regenerative amplifier laser system, which is then adapted to Q-switched lasers in Chapter 3. In Section 2.1, the model of the discrete-time system is discussed. Next, a linear controller is used to stabilize the set-point in Section 2.2. Section 2.3 then investigates several reinforcement learning algorithms and their respective performances. To benchmark these results, Section 2.4 compares them to a nonlinear controller based on a dead-beat approach. Finally, the influence of parameter variations and hence the robustness of the different discussed controllers are studied in Section 2.5. Section 2.6 briefly summarizes the main results of this chapter.

## 2.1 Model

For the controller design and simulations in this chapter, a model of the pulse-to-pulse dynamics of a regenerative amplifier proposed in [18] is used. A seed pulse from a source passes through an acousto-optical modulator (AOM) element which acts as a control input. This device adapts the portion of the energy of the seed pulse which travels on into the linear cavity. Therefore, the control input is inherently constrained to the range $[u_{\min}, u_{\max}]$, where $u_{\min}$ and $u_{\max}$ represent the minimum and maximum seed pulse energy before the AOM element, respectively. In the linear cavity, the pulse is amplified during each round trip and is coupled out, eventually. Afterwards, the gain medium in the cavity recovers its population inversion during the remaining time until the next pulse is generated by the seed pulse source. Thus, the pulse-to-pulse dynamics can be modelled in two steps:

- the amplification of the seed pulse
- the recovery of the gain medium.

The model uses the gain $G$ which can be linked to the population inversion and the normalized fluence $H$ to describe the dynamical evolution of these quantities after each pass through the medium. For a linear cavity, the formulas given in [18] are slightly altered to fit the different geometry of the cavity. The single-pass function $\mathbf{f}_{\mathrm{SP}}(\cdot)$ is expanded to a dual-pass function

$$\begin{bmatrix} G_{k,n+1} \\ H_{k,n+1} \end{bmatrix} = \mathbf{f}_{\mathrm{DP}}\left(\begin{bmatrix} G_{k,n} \\ H_{k,n} \end{bmatrix}\right) = \mathbf{f}_{\mathrm{SP}}\left(\begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{\eta_{RC}} \end{bmatrix} \mathbf{f}_{\mathrm{SP}}\left(\begin{bmatrix} G_{k,n} \\ H_{k,n} \end{bmatrix}\right)\right) \tag{2.1}$$

where $G_{k,n}$ denotes the gain of the medium after $n$-th pass of the $k$-th pulse through the medium and $\eta_{RC}$ are the concentrated cavity losses. The multi-pass function changes to $\mathbf{f}_{\mathrm{MP}} = \mathbf{f}_{\mathrm{DP}}^{(N_{\mathrm{RT}})} = \mathbf{f}_{\mathrm{DP}} \circ \cdots \circ \mathbf{f}_{\mathrm{DP}}$, accordingly, where $N_{\mathrm{RT}}$ is the number of round-trips that the pulse travels through the cavity. The second step, the recovery of the gain medium, is unaffected by these changes and remains

$$G_{k+1,1} = f_{\text{pump}}\left(\begin{bmatrix}1 & 0\end{bmatrix}\mathbf{f}_{\text{MP}}\left(\begin{bmatrix}G_{k,1}\\ H_{k,1}\end{bmatrix}\right), p\right). \tag{2.2}$$

It should be noted at this point that the function $f_{\text{pump}}$ depends on the pumping power $p$. In this way, the gain before the next pulse $x_{k+1} = G_{k+1,1}$ can be expressed as a function of the gain before the current pulse $x_k = G_{k,1}$ and the current input fluence $u_k = H_{k,1}$. The normalized fluence of the pulse after amplification at the output $y_k = H_{k,N_{\text{RT}}}$ can serve as a output quantity for the plant. More details about the model, including the model parameters, are stated in Appendix A.1.

Summing up, the discrete-time model is very generally expressed as follows

$$x_{k+1} = f(x_k, u_k), \tag{2.3a}$$

$$y_k = h(x_k, u_k). \tag{2.3b}$$

The steady-state values, which will be denoted with subscript $s$, can be obtained by solving the fixed point equation

$$x_s = f(x_s, u_s) \tag{2.4}$$

for a given desired steady state control input $u_s$. However, as the gain is often not directly measurable, [18] suggests to use output measurements instead to model to pulse-to-pulse dynamics

$$y_{k+1} = h(f(h^{-1}(y_k, u_k), u_k), u_{k+1}) = \bar{f}(y_k, u_k, u_{k+1}). \tag{2.5}$$

This requires the inverse function of $h$ for a given $u$, which can be found by the implicit function theorem locally if $\frac{\partial h}{\partial x} \neq 0$. Figure 2.1 shows the functions $f(x, u_s)$ and $h(x, u_s)$ for the set-point defined by $p = 297$ W, $u_s = 1.56482 \cdot 10^{-8}$ and the parameters from Table A.1. In particular, it stands out that $h(x, u_s)$ is not injective. This implies that the value for $x$ can not be uniquely determined on a global scale by using this output quantity.

An alternative output quantity which fulfils the latter property is the so-called build-up. It is given as

$$b_k = \sum_{n=1}^{N_{\text{RT}}} H_{k,n} = \tilde{h}(x_k, u_k). \tag{2.6}$$

The build-up increases with each round-trip in the cavity and therefore represents the energy build-up of the light pulse during amplification. It can be measured using a photo-diode in combination with a low-pass filter. The function $\tilde{h}(x, u_s)$ is shown in the third row in Figure 2.1. This function is strictly increasing with $x$ and therefore allows for a one-to-one mapping between the build-up and the gain. The modified pulse-to-pulse dynamics using the build-up as output quantity is given as

$$b_{k+1} = \tilde{h}(f(\tilde{h}^{-1}(b_k, u_k), u_k), u_{k+1}) = \tilde{f}(b_k, u_k, u_{k+1}). \tag{2.7}$$

The pulse-to-pulse dynamic for the selected set-point is unstable and shows bifurcations, as will be demonstrated in simulations later. Thus, the next sections develop controllers to stabilize the set-point.

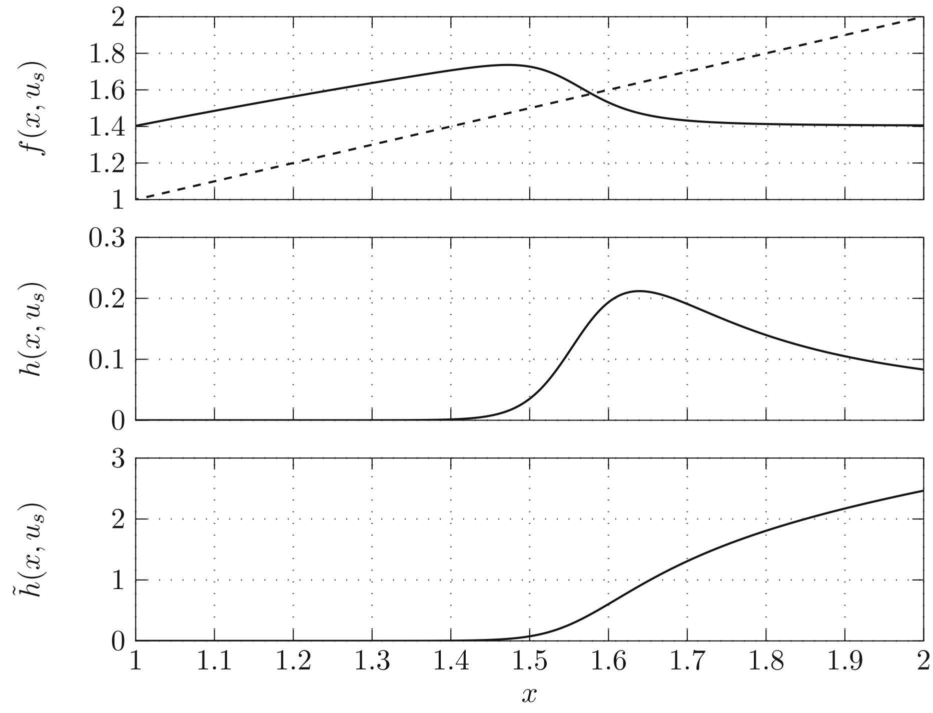

Figure 2.1: Pulse-to-pulse dynamics of a regenerative amplifier at a selected set-point with output energy (second row) and build-up (third row).

## 2.2 Linear state feedback controller

In this section, the theory for the linear quadratic regulator (LQR) problem will be revisited briefly as it lays the foundations for the reinforcement learning framework.

The basis for the analysis is (2.7) which can be expressed with a new input $\nu_k = u_{k+1}$ and augmented state $\mathbf{z} = \begin{bmatrix} b & u \end{bmatrix}^{\mathrm{T}}$ as

$$\mathbf{z}_{k+1} = \begin{bmatrix} \tilde{f}(b_k, u_k, \nu_k) \\ \nu_k \end{bmatrix} = \mathbf{f}(\mathbf{z}_k, \nu_k). \tag{2.8}$$

The LQR problem is to find a linear state feedback $\Delta\nu_k = \mathbf{k}^{\mathrm{T}}\Delta\mathbf{z}_k$ with $\Delta\mathbf{z}_k = \mathbf{z}_k - \mathbf{z}_s$ and gain vector $\mathbf{k}$ to minimize the cost function of the linearized system

$$J(\mathbf{z}_k) = \sum_{n=k}^{\infty}\left(\Delta\mathbf{z}_n^{\mathrm{T}}\mathbf{Q}\Delta\mathbf{z}_n + R\Delta\nu_n^2\right) = \sum_{n=k}^{\infty}\left(Q_{00}\Delta b_n^2 + 2Q_{01}\Delta b_n\Delta u_n + Q_{11}\Delta u_n^2 + R\Delta\nu_n^2\right), \tag{2.9}$$

where the latter equality sign holds if one assumes a symmetric matrix $\mathbf{Q} = \mathbf{Q}^{\mathrm{T}} = [Q_{ij}]$. $R$ is the weighting factor for the control input $\nu$. In [18], a tanh-smoothing of the linear state feedback law is incorporated to account for the input constraints. This will also be used in the simulations.

## 2.3 Reinforcement learning

Many of the nowadays popular RL algorithms like DDPG, PPO and SAC are either actor-critic algorithms or split up the RL problem in a part involving the estimation of state values/advantage function and optimizing the policy based on this function. The critic is usually trained to learn a state or a state-action or advantage function. The actor, on the other hand, tries to optimize its policy parameters by the values from the critic and the state utilizing some variant of the policy gradient theorem. Figure 2.2 shows a generic information flow for the actor-critic framework.

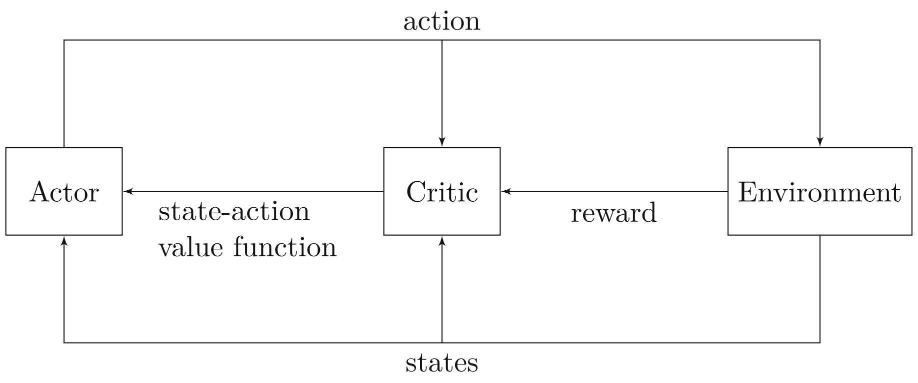

Figure 2.2: Generic information flow of the actor-critic framework.

Both actor and critic are often implemented as neural networks. They normally include several hidden layers and are therefore computationally expensive to optimize. This does usually not constitute a problem as many applications do not require real-time operation. For other applications that require fast computation, especially in the robotics area, model parameters are typically very well-known or can be determined from experiments. Off-line RL offers a practical solution for these applications as only the actor has to be implemented in real-time after learning is completed. Furthermore, no gradient calculation is involved after learning, so the sample time is mainly limited by the forward pass through the actor network.

For the use case investigated in this work, i. e. for controlling the light pulse energy of a regenerative amplifier and a laser system based on cavity dumping, off-line RL is not applicable. The main reason for that is that many model parameters are not known accurately or are subject to substantial drift, caused by, amongst others, thermal dependencies. Thus, the RL algorithms presented in this work are selected to fit in the short computation window between two pulses. Repetition rates for Q-switched laser systems can reach the MHz range [35], so the computation time should ideally meet this requirement.

### 2.3.1 Reward function

In the following, the reward function that is used for the RL framework will be developed, starting from the LQR problem in Section 2.2.

The cost function for the LQR problem was given in (2.9). As a first step, the cost function of (2.9) can be rewritten as

$$J(\mathbf{z}_k) = \sum_{n=k}^{\infty}\left(\Delta\mathbf{z}_n^{\mathrm{T}}\mathbf{Q}\Delta\mathbf{z}_n + R\Delta\nu_n^2\right) = \sum_{n=k}^{\infty} c(\mathbf{z}_n, \nu_n) \tag{2.10a}$$

with

$$c(\mathbf{z}_n, \nu_n) = \Delta\mathbf{z}_n^{\mathrm{T}}\mathbf{Q}\Delta\mathbf{z}_n + R\Delta\nu_n^2. \tag{2.10b}$$

It has to be emphasized that the cost function depends on the current state deviation from its set-point $\Delta\mathbf{z}_k$. While from a control engineering perspective, this fact is convenient to show stability of the closed loop using Lyapunov's direct method in a model predictive control setting [36, Chapter 4], it is not so well-suited in the RL framework. In RL, the idea is to reward the agent based on what action it took. This is reflected by the state the environment lands in after control input $\nu_k$ was applied to the plant which corresponds to state $\mathbf{z}_{k+1}$. So, it is more intuitive to use a slightly modified cost function

$$J_d(\mathbf{z}_k) = \sum_{n=k}^{\infty} c_d(\mathbf{z}_n, \nu_n) \tag{2.11a}$$

with

$$c_d(\mathbf{z}_n, \nu_n) = Q_{00}\Delta b_{n+1}^2 + 2Q_{01}\Delta b_{n+1}\Delta u_n + Q_{11}\Delta u_n^2 + R\Delta\nu_n^2, \tag{2.11b}$$

assuming a symmetric matrix $\mathbf{Q} = \mathbf{Q}^{\mathrm{T}}$. Simulations show that without this index shift for the build-up, learning can be decelerated significantly. In this case, suboptimal actions become visible to the agent only with unit delay. In contrast, the agent receives information on how good the last action was immediately with (2.11b), speeding up the learning process.

Another key difference between cost functions in control theory and reward functions in RL includes that in the RL framework, reward functions are typically maximized, whereas the cost function in (2.9) is minimized. This can be easily fixed by assigning each term a negative sign. Finally, RL rewards are typically equipped with a discount factor $0 < \gamma < 1$ which helps the algorithms to converge in the infinite-horizon case.

Summing up, the total reward for one state is given as

$$R_m(\mathbf{z}_k) = \sum_{n=k}^{\infty} \gamma^{n-k} r_m(\mathbf{z}_n, \nu_n) \tag{2.12a}$$

with (2.11b) and

$$r_m(\mathbf{z}_n, \nu_n) = -c_d(\mathbf{z}_n, \nu_n). \tag{2.12b}$$

(2.12) is still not completely model-free as it consists of expressions involving $\Delta\mathbf{z}_n = \mathbf{z}_n - \mathbf{z}_s$, in particular $\Delta b_n = b_n - b_s$. In general, the value for the steady-state build-up $b_s$ is not given in the model-free setting. A way to circumvent this issue is by replacing the difference between the current value of the build-up and the steady state value with the difference between two consecutive measurements of the build-up, i. e.

$$R_f(\mathbf{z}_k) = \sum_{n=k}^{\infty} \gamma^{n-k} r_f(\mathbf{z}_n, \nu_n) \tag{2.13a}$$

$$r_f(\mathbf{z}_n, \nu_n) = -Q_{00}(b_{n+1} - b_n)^2 - 2Q_{01}(b_{n+1} - b_n)\Delta u_n - Q_{11}\Delta u_n^2 - R\Delta\nu_n^2. \tag{2.13b}$$

From (2.13b) and the second line of (2.8), it becomes clear that both $Q_{11}$ and $R$ assign weight to the control input, only delayed by one sample step. (2.13) can be used in a model-free setting if no steady state build-up is (accurately) known. One interpretation of this formulation would be that the build-up signal is sent through a discrete-time differentiator before entering the reward function.

### 2.3.2 State space discretization

As already mentioned, this work focusses primarily on RL algorithms which require little computation time. Algorithms which fulfil this property, such as Q-learning, SARSA or Monte Carlo methods [22, Chapters 5, 6], all share a tabular-based architecture. This is because memory-accesses to values of states (or state-action-pairs) can be done swiftly and update laws can be calculated easily without the need for a gradient.

On the other hand, a tabular-based structure requires a finite number of states. However, the control problem is given in two continuous dimensions for the state and one continuous dimension for the control input. Hence, some sort of state-space discretization must be used to transform the original problem into a tabular-based setting with a finite number of discrete states.

One intuitive way to discretize states and actions is to tile up the state space. Equal spacings between tiling borders are a straight-forward way to partition the state space.

Unfortunately, using this method typically requires a large number of tiles in each axis in order to yield an acceptable control result. Thus, a different strategy is pursued to select tiling sizes.

As the goal is to stabilize a given set-point, many measurements will occur near this equilibrium point. Instances that are far away from the nominal set-point will be seen infrequently during normal operation, in particular only during the initial transient phase. Therefore, it is preferable to have many (discrete) states in close proximity to the nominal set-point and a smaller density of states far away.

In the following, a tiling pattern that implements this idea will be constructed using a linear spacing mapped by a third order polynomial. To calculate the coefficients of this polynomial, some boundaries for the region of interest $\xi_{\min}, \xi_{\max}$ around the steady state point $\xi_s \in [\xi_{\min}, \xi_{\max}]$ have to be selected. These boundaries define the maximum and minimum value which are chosen as a center of a tile and should cover a wide range of possible values that either the build-up or the control input can assume. In this context, $\xi$ represents a generic quantity which can refer to either the build-up $b$ or the control input $u$ or $\nu$. From the boundaries $[\xi_{\min}, \xi_{\max}]$, the mapping $p_\xi : [0,1] \to [\xi_{\min}, \xi_{\max}]$

$$p_\xi(\zeta) = \begin{cases} -8(\xi_{\min} - \xi_s)(\zeta - \frac{1}{2})^3 + \xi_s & \text{if } \zeta < \frac{1}{2}, \\ 8(\xi_{\max} - \xi_s)(\zeta - \frac{1}{2})^3 + \xi_s & \text{else} \end{cases} \tag{2.14}$$

transforms the linear spacing $\zeta_i = \frac{i}{N_\xi - 1}, i = 0, 1, ..., N_\xi - 1$, with $N_\xi \in \mathbb{N}_+$ being the number of centres, into the non-linear spacing of the centre of the tiles $\check{\xi}_i = p(\zeta_i)$. The accent $(\check{\cdot})$ is used to indicate discrete values. Finally, the continuous state is rounded to the nearest center

$$\check{\xi} = \mathrm{round}(\xi) \tag{2.15a}$$

with the rounding operator

$$\mathrm{round}(\xi) = \check{\xi}_i,\ i = \arg \min_{i=0,\dots,N_\xi-1} |\xi - \check{\xi}_i|. \tag{2.15b}$$

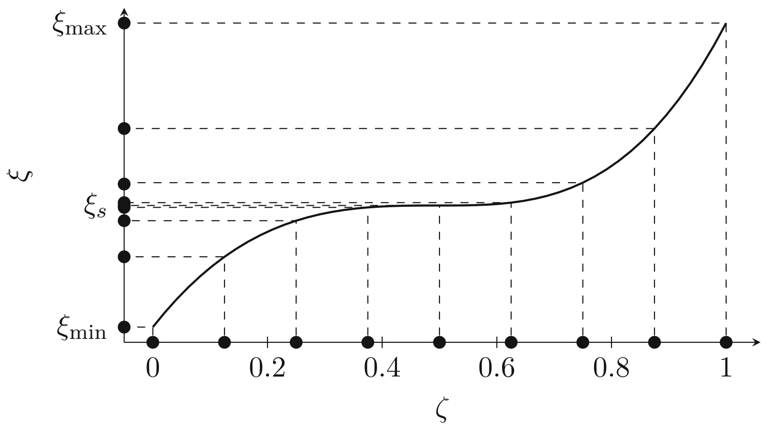

Figure 2.3: Calculation of center points.

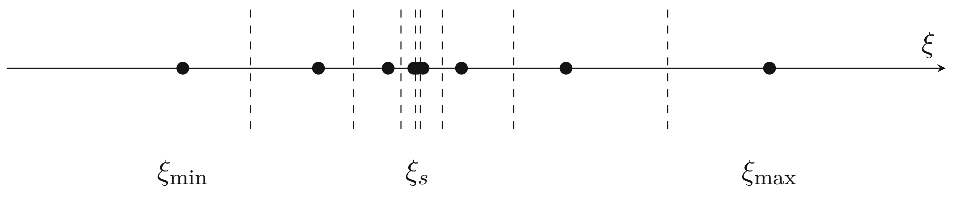

Figure 2.4: Decision boundaries.

Figure 2.3 shows the generation of $N_\xi = 9$ center points for a generic quantity $\xi$. As visible, the extremely flat curve around $\zeta = 0.5$, which is associated with the steady state, generates many discrete states in this region, while for large $|\xi - \xi_s|$, only few states are allocated. In Figure 2.4, the decision boundaries of each finite state are displayed. It has to be noted that the tile centres (drawn as dots) are not in the middle of the tile. The decision boundaries (drawn dashed) divide the distance between two centres in half. The left- and right-most states contain an infinitely large decision region as all values above/below are assigned to one of the extreme values $\xi_{\max}$ or $\xi_{\min}$.

The two-dimensional state-space investigated in this chapter consists of a value for the build-up and a value for the last control input. Therefore, the discretization proposed in this section can be applied to both dimensions with a number of discrete values for the build-up $N_b$ and a number for the discrete states for the last control input $N_u$. An example is shown in Figure 2.5, with $N_b = 7$ and $N_u = 5$.

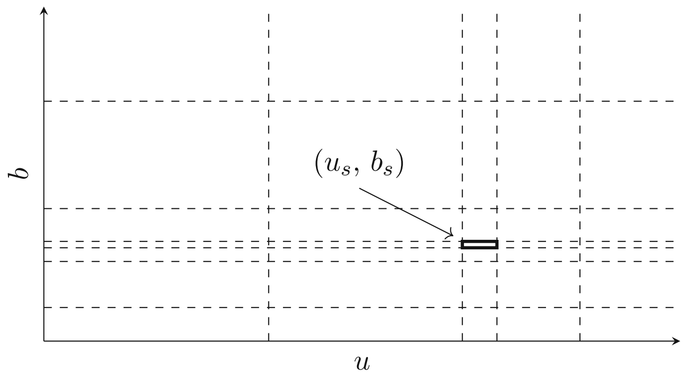

Figure 2.5: Two-dimensional discretized state space.

Because $\nu_k = u_{k+1}$, the state-space discretization automatically leads to a discretization of actions as well. In total, the state space has a finite number of $N_b \cdot N_u$ states, while the space of state-actions has a number of $N_b \cdot N_u^2$ possible combinations.

### 2.3.3 Markov Decision Process

This section formalizes all relevant details of the model, reward and discretization into the RL setting using Markov Decision Processes. A Markov Decision Process (MDP) is a tuple containing a set of states $\mathcal{Z}$, a set of actions $\mathcal{U}$, a transition probability between states $p(\check{\mathbf{z}}_{k+1}|\check{\mathbf{z}}_k, \check{\nu}_k)$, a reward function $R_{\text{MDP}}(\check{\mathbf{z}}_k)$ and a discount factor $\gamma$ [37, Chapter

2]. The transition probability reflects the Markov property that the next state $\check{\mathbf{z}}_{k+1}$ only depends on the current state $\check{\mathbf{z}}_k$ and the current control input $\check{\nu}_k$ and is independent of past states $\check{\mathbf{z}}_{k-1}, \check{\mathbf{z}}_{k-2}, \dots$ .

For the given application, the set of states $\mathcal{Z} = \mathcal{B} \times \mathcal{U}$ is given as the Cartesian product of the set of values for the build-up $\mathcal{B} = \{\check{b}_i, i = 1, ..., N_b\}$ and the set of discrete control inputs $\mathcal{U} = \{\check{u}_i, i = 1, ..., N_u\}$. The latter is the set of actions, simultaneously. As the pulse-to-pulse dynamics of the regenerative amplifier is given as a deterministic model (2.8), the transition probability

$$p(\check{\mathbf{z}}_{k+1}|\check{\mathbf{z}}_k, \check{\nu}_k) = \begin{cases} 1 & \text{if } \check{\mathbf{z}}_{k+1} = \operatorname{round}(\mathbf{f}(\check{\mathbf{z}}_k, \check{\nu}_k)) \\ 0 & \text{else} \end{cases} \tag{2.16}$$

only exhibits a non-zero probability where the nominal pulse-to-pulse dynamics fits best according to the rounding operation. This implies that there is no process or measurement noise apart from the quantization noise which stems from the discretization. The reward function $R_{\mathrm{MDP}}(\check{\mathbf{z}}_k)$ is either based on the model-based reward function in (2.12) or the model-free reward function in (2.13).

Finally, the discount factor is chosen to be $\gamma = 0.99$. In the infinite horizon setting, a discount factor $\gamma < 1$ is a typical assumption for convergence proofs of several dynamic programming approaches such as value iteration and policy iteration [37, p. 143]. A low value for $\gamma$ results in faster convergence speed and fewer iterations required by the algorithms to reach a desired accuracy.

However, a convergent discounted reward does not necessarily imply a stable dynamic system. In [38], the authors provide stability guarantees for the closed loop if the discount factor $\gamma$ of the cost/reward function is in the range between some lower bound that is dependent on the plant and 1. From a control engineering perspective, the discount factor should therefore be as close to 1 as possible. The selected value of $\gamma = 0.99$ tries to balance a fast convergence speed of the RL algorithms while still maintaining stable pulse-to-pulse dynamics.

With all the necessary preliminaries at hand, the goal now is to find a control law $\check{\nu} = k(\check{\mathbf{z}})$ for every state $\check{\mathbf{z}} \in \mathcal{Z}$. The following sections will treat model-based and model-free RL algorithms which try to fulfil this task.

### 2.3.4 Q-value iteration

This section covers a model-based RL algorithm known as Q-value iteration, a variant of the value iteration algorithms [37]. These algorithms are based on dynamical programming which splits up an optimization problem into the current step and all future steps exploiting Bellman's principle of optimality [39]. For the deterministic case investigated here, the Bellman equation

$$Q^*(\check{\mathbf{z}}_k, \check{\nu}_k) = r_m(\check{\mathbf{z}}_k, \check{\nu}_k) + \gamma \max_{\check{\nu}_{k+1}} Q^*(\operatorname{round}(\mathbf{f}(\check{\mathbf{z}}_k, \nu_k)), \check{\nu}_{k+1}) \tag{2.17}$$

for the optimal state-action values $Q^*(\check{\mathbf{z}}_k, \check{\nu}_k)$ is the central approach to solving for the optimal control law [22, Chapter 3]. The optimal state-action values can be interpreted as the future discounted reward when starting in state $\check{\mathbf{z}}_k$ and applying control input $\check{\nu}_k$ to the plant. Afterwards, only optimal control actions are selected, i. e.

$$Q^*(\check{\mathbf{z}}_k, \check{\nu}_k) = r_m(\check{\mathbf{z}}_k, \check{\nu}_k) + \sum_{n=k+1}^{\infty} \gamma^{n-k} r_m(\check{\mathbf{z}}_n, \check{\nu}_n^*). \tag{2.18}$$

Hence, the optimal control law for the current state $\check{\mathbf{z}}_k$,

$$\check{\nu}_k^* = \arg\max_{\check{\nu}_k} Q^*(\check{\mathbf{z}}_k, \check{\nu}_k), \tag{2.19}$$

can be extracted by choosing the control input that maximizes the state-action value [22, Chapter 3].

Q-value iteration is an algorithm to obtain these state-action values. It iteratively assigns new state-action values

$$Q_{l+1}(\check{\mathbf{z}}_k, \check{\nu}_k) \leftarrow r_m(\check{\mathbf{z}}_k, \check{\nu}_k) + \gamma \max_{\check{\nu}_{k+1}} Q_l(\mathrm{round}(\mathbf{f}(\check{\mathbf{z}}_k, \nu_k)), \check{\nu}_{k+1}) \tag{2.20}$$

with $Q_l(\cdot)$ being the estimate for the $Q$-function after iteration $l$. This assignment is performed for all state-action combinations $(\check{\mathbf{z}}_k, \check{\nu}_k)$ during one iteration. As Q-value iteration is a variant of the larger set of value iteration algorithms, it shares its convergence properties [37, Theorem 6.3.1]. As all necessary assumptions for Theorem 6.3.1 are fulfilled, Q-value iteration is guaranteed to converge for the defined MDP.

However, the control law obtained by Q-value iteration might not stabilize the laser system even if the algorithm itself converges. The reason for this lies (in addition to the discounting issue) in the quantization process and the transition probability in (2.16).

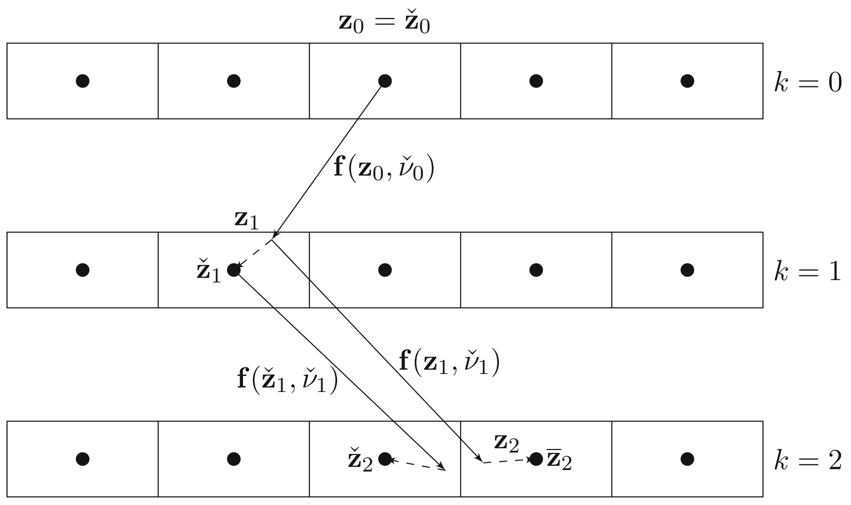
$\mathbf{z}_0 = \check{\mathbf{z}}_0$
$k = 0$
$\mathbf{f}(\mathbf{z}_0, \check{\nu}_0)$
$\mathbf{z}_1$
$\check{\mathbf{z}}_1$
$k = 1$
$\mathbf{f}(\check{\mathbf{z}}_1, \check{\nu}_1)$
$\mathbf{f}(\mathbf{z}_1, \check{\nu}_1)$
$\check{\mathbf{z}}_2$
$\mathbf{z}_2$
$\bar{\mathbf{z}}_2$
$k = 2$

Figure 2.6: Quantization issue.

The rounding operation assigns a continuous build-up to a state in $\mathcal{B}$. This alone would not pose a problem. However, it might become one if a sequence of more than one action is concatenated. For the following considerations, it will be assumed that the real system starts (by chance) exactly from a center of a discretized tile, i. e. $\mathbf{z}_0 \in \mathcal{Z}$. Consequently, the discretized version $\check{\mathbf{z}}_0 = \mathbf{z}_0$ matches perfectly with the state of the plant. A graphical illustration of this issue is drawn in Figure 2.6.

After the first transition, when an arbitrary control input $\check{\nu}_0$ has been applied, the state of the real plant is $\mathbf{z}_1 = \mathbf{f}(\mathbf{z}_0, \check{\nu}_0)$ whereas the state visible to the MDP (and by extension to the Q-value iteration algorithm) is $\check{\mathbf{z}}_1 = \text{round}(\mathbf{z}_1)$. After the second transition, the quantization issue becomes evident. The real system arrives at state $\mathbf{z}_2 = \mathbf{f}(\mathbf{z}_1, \check{\nu}_1)$. The associated rounded state should be $\overline{\mathbf{z}}_2 = \text{round}(\mathbf{z}_2)$. Unfortunately, this does not always coincide with the nominal transition $\check{\mathbf{z}}_2 = \text{round}(\mathbf{f}(\check{\mathbf{z}}_1, \check{\nu}_1))$ assumed in the MDP. In essence, the problem arises from the fact that

$$\text{round}\{\mathbf{f}(\mathbf{f}(\check{\mathbf{z}}_k, \check{\nu}_k), \check{\nu}_{k+1})\} \neq \text{round}\{\mathbf{f}(\text{round}[\mathbf{f}(\check{\mathbf{z}}_k, \check{\nu}_k)], \check{\nu}_{k+1})\}. \tag{2.21}$$

In particular, the transition $(\check{\mathbf{z}}_1, \check{\nu}_1) \to \overline{\mathbf{z}}_2$ shown in Figure 2.6 is associated with 0 probability (forbidden transition) according to (2.16). In general, this means that neighbouring states of $\check{\mathbf{z}}_{k+1}$ will have a non-zero transition probability. This transition probability depends on the initial (continuous) distribution where the real system starts within the discretized tile $p(\mathbf{z}_k|\check{\mathbf{z}}_k)$. If this probability were known, one could calculate a modified transition probability

$$\tilde{p}(\check{\mathbf{z}}_{k+1}|\check{\mathbf{z}}_k, \check{\nu}_k) = \int_{\check{\mathcal{Z}}} p(\check{\mathbf{z}}_{k+1}|\mathbf{z}_k, \check{\nu}_k) p(\mathbf{z}_k|\check{\mathbf{z}}_k) \mathrm{d}\mathbf{z}_k \tag{2.22}$$

with $\check{\mathcal{Z}} = \{\mathbf{z}_k \in \mathbb{R}^2 | \text{round}(\mathbf{z}_k) = \check{\mathbf{z}}_k\}$ and the transition probability

$$p(\check{\mathbf{z}}_{k+1}|\mathbf{z}_k, \check{\nu}_k) = \begin{cases} 1 & \text{if } \check{\mathbf{z}}_{k+1} = \text{round}(\mathbf{f}(\mathbf{z}_k, \check{\nu}_k)) \\ 0 & \text{else} \end{cases}. \tag{2.23}$$

From the view of the MDP, the deterministic pulse-to-pulse dynamics (2.8) becomes stochastic due to quantization. This could be accounted for by solving the stochastic version

$$\begin{aligned} Q^*(\check{\mathbf{z}}_k, \check{\nu}_k) &= \mathbb{E}_{\check{\mathbf{z}}_{k+1} \sim \tilde{p}(\check{\mathbf{z}}_{k+1}|\check{\mathbf{z}}_k, \check{\nu}_k)} \left( r_m(\check{\mathbf{z}}_k, \check{\nu}_k) + \gamma \max_{\check{\nu}_{k+1}} Q^*(\check{\mathbf{z}}_{k+1}, \check{\nu}_{k+1}) \right) \\ &= \sum_{\check{\mathbf{z}}_{k+1} \in \mathcal{Z}} \tilde{p}(\check{\mathbf{z}}_{k+1}|\check{\mathbf{z}}_k, \check{\nu}_k) \left( r_m(\check{\mathbf{z}}_k, \check{\nu}_k) + \gamma \max_{\check{\nu}_{k+1}} Q^*(\check{\mathbf{z}}_{k+1}, \check{\nu}_{k+1}) \right) \end{aligned} \tag{2.24}$$

of the Bellman equation [22, Chapter 3].

However, when looking at (2.21), it is evident that the probability $p(\mathbf{z}_k|\check{\mathbf{z}}_k)$ will not only depend on the last state because $\overline{\mathbf{z}}_2 \neq \check{\mathbf{z}}_2$ is caused by the succession of two transitions. Strictly speaking, the transition probability does not only depend on the last state, but rather on all previous states in combination with the control inputs selected, so

$$p(\mathbf{z}_k|\check{\mathbf{z}}_k) = p(\mathbf{z}_k|\check{\mathbf{z}}_k, \check{\mathbf{z}}_{k-1}, \check{\nu}_{k-1}, \check{\mathbf{z}}_{k-2}, \check{\nu}_{k-2}, \ldots) \tag{2.25}$$

and therefore violates the Markov property. Thus, the modeling of the transition probability in (2.16) does not perfectly reflect the actual transitions which will take place.

Simulations show that this issue can become problematic if states are quantized coarsely, i. e. for small $N_b$ and $N_u$. In this case, quantization errors can be large, leading to frequent violations of the nominal transitions as in (2.21) and transient responses of the closed loop may show limit cycles due to the mismatch between the MDP model and the actual pulse-to-pulse dynamics. If $N_b$ and $N_u$ are selected sufficiently large, these undesirable effects can be avoided because quantization errors remain negligible.

### 2.3.5 Monte Carlo Control

Starting from this section, model-free RL algorithms are applied, i. e. algorithms which have no knowledge about the system and try to learn an optimal policy solely based on interactions with the environment. For the tabular setting, the RL algorithms are typically value-based, so the aim is to retrieve the policy from the state-action values.

These algorithms rely on generalized policy iteration, an alternating sequence of policy evaluation (calculating the state-action value for a policy) and policy improvement (finding a policy that increases the state values).

In this section, Monte Carlo (MC) control methods in RL will be investigated. Although several variants exist, such as Exploring Starts (MCES), or on- and off-policy algorithms [22, Chapter 5], they all share the same property. They are necessarily limited to episodic settings because the returns which are required to learn the state-action values are computed from the episode end backwards. One sample for the future reward is given as

$$\hat{Q}(\check{\mathbf{z}}_k, \check{\nu}_k) = \sum_{n=k}^{N} \gamma^{n-k} r_f(\check{\mathbf{z}}_n, \check{\nu}_n) \tag{2.26}$$

with the episode length $N$. Samples of the same state-action pair are averaged over the learning process. One update step

$$Q(\check{\mathbf{z}}_k, \check{\nu}_k) \leftarrow Q(\check{\mathbf{z}}_k, \check{\nu}_k) + \alpha_K(\hat{Q}(\check{\mathbf{z}}_k, \check{\nu}_k) - Q(\check{\mathbf{z}}_k, \check{\nu}_k)) \tag{2.27}$$

modifies the current estimate of the state-value function $Q(\check{\mathbf{z}}_k, \check{\nu}_k)$ using the generated sample $\hat{Q}(\check{\mathbf{z}}_k, \check{\nu}_k)$ with a time-dependent step size or learning rate parameter $\alpha_K$. For a true average over the samples, the step size parameter $\alpha_K = 1/N_K(\check{\mathbf{z}}_k, \check{\nu}_k)$ must be chosen, with $N_K(\check{\mathbf{z}}_k, \check{\nu}_k)$ being the total number of visits to the state-action pair over the whole learning process up until now. While $k$ denotes local time in an episode, $K$ represents global time which spans all episodes over the whole learning process.

Thus, MC methods do not use dynamical programming. Different states are not connected by the update law directly which can lead to longer learning times because each state-action pair is learnt separately, ignoring the inherent connections between states imposed by the Markov property. On the other hand, if the Markov property is violated, Monte Carlo approaches might still be applicable while other methods which exploit bootstrapping are not.

In MC control, policy evaluation and policy improvement steps are done on an episode-by-episode basis. For the current policy, one episode is generated and samples for the state-action values are calculated using (2.26). There exist two ways to deal with the issue if one state occurs more than once in an episode. First-visit methods will only calculate a sample according to (2.26) the first time the state is encountered in an episode. Every-visit methods will generate a sample for every time a state is visited in an episode. It can be shown that in the limit, as the number of samples tends to infinity, both methods will deliver the correct expectation for the state-value function. However, in the finite case, the estimate for the state-value functions of every-visit methods is biased, while the estimate for first-visit methods is not [40, Chapter 5.2.1]. On the other hand, the mean squared error (bias$^2$ + variance) of the every-visit method is lower compared to first-visit methods for single episodes [41]. Another aspect is computational effort. First-visit methods are computationally more expensive as the algorithm needs to check whether a state has already been visited during the current episode before generating a sample.

A central concept in RL is the exploration/exploitation trade-off [22, Chapter 1]. Every possible state should be explored to find an optimal sequence because otherwise, one could not be guaranteed that there does not exist a path that generates an even larger reward than the currently known optimal policy. Exploitation, on the other hand, means to select the best known policy which is crucial to finding sequences of control actions which maximize the rewards and fulfil the control task optimally.

MCES explores by selecting initial states at random at the beginning of each episode such that every state will be visited. Although there is yet no convergence proof for a general MDP to converge to the optimal policy using MCES [22, Chapter 5.3], the authors in [42] provide a convergence proof for certain classes of MDPs.

On-policy MC methods treat exploration differently. These methods learn a stochastic policy. Popular methods to create such stochastic policies in the tabular setting are $\varepsilon$-greedy [22, Chapter 5.4] and softmax [43, Chapter 4.1]. The $\varepsilon$-greedy policy with the probability distribution

$$p(\check{\nu}_i|\mathbf{z}) = \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{N_u} & \text{if } \check{\nu}_i = \arg\max_{\check{\nu}} Q(\mathbf{z}, \check{\nu}) \\ \frac{\varepsilon}{N_u} & \text{else} \end{cases}, \quad i = 0, 1, ..., N_u - 1 \tag{2.28}$$

selects the currently best known action with high probability $1 - \varepsilon$. With low probability $\varepsilon$, an action is selected at random from a uniform distribution. In essence, this policy can be implemented by drawing two uniform random variables or drawing one uniform random variable and performing one maximization operation. By contrast, the softmax policy

$$p(\check{\nu}_i|\mathbf{z}) = \frac{\exp(Q(\mathbf{z}, \check{\nu}_i))}{\sum_{j=0}^{N_u-1} \exp(Q(\mathbf{z}, \check{\nu}_j))}, \quad i = 0, 1, ..., N_u - 1 \tag{2.29}$$

would include $N_u$ time-consuming evaluations of the exponential function to calculate the distribution.

The disadvantage of on-policy methods is that they learn a stochastic policy which might not be desirable in a deterministic setting where a deterministic policy could perform better. Off-policy methods operate with two different policies: a target policy which is optimized and a behavioral policy that drives exploration. When generating roll-outs using the behavioral policy, this leads to an incorrect expectation value of the future rewards. This can be corrected using importance sampling. Importance sampling is a technique to update the sample generated by the behavioral policy to fit the expected value of the target policy using the importance sampling ratio

$$\rho_{k|N} = \prod_{n=k}^{N-1} \frac{p(\check{\nu}_n|\check{\mathbf{z}}_n)}{p_b(\check{\nu}_n|\check{\mathbf{z}}_n)} \tag{2.30}$$

where $p_b(\check{\nu}_k|\check{\mathbf{z}}_k)$ denotes the probability of the behavioral policy to select control action $\check{\nu}_k$ in state $\check{\mathbf{z}}_k$ and $p(\check{\nu}_k|\check{\mathbf{z}}_k)$ the corresponding probability of the target policy. With this correction step, a MC off-policy variant can be implemented with a new estimate for the state-action value [22, Chapter 5.5]

$$\hat{Q}(\check{\mathbf{z}}_k, \check{\nu}_k) = \sum_{n=k}^{N} \gamma^{n-k} \rho_{k|N} r_f(\check{\mathbf{z}}_n, \check{\nu}_n). \tag{2.31}$$

However, if the target policy is deterministic, then any exploration at time $k$ will lead to $p(\check{\nu}_k|\check{\mathbf{z}}_k) = 0$ and $\rho_{n|N} = 0$, $\forall n \leq k$ due to the product structure of (2.30). The update step (2.27) is then omitted if $\rho_{n|N} = 0$, implying that the learning will only occur from the episode end backwards. Unweighted importance sampling can be distinguished from weighted importance sampling. While the former is unbiased, the latter is not [44]. However, both are consistent estimators of the true state-value function.

With regard to the control objective of this work, to stabilize a set-point of a pulsed laser system, MC methods are not directly applicable. They require episodic settings while the control task is a priori not time-limited. Nevertheless, it is possible to group every $N$ measurements together to form an episode. This will, of course, lead to incorrect estimates of state-action values at the end of each episode. If an episode does not end at the steady state, state-action values will be overestimated. This phenomenon is caused by the missing negative rewards corresponding to the intermediate states which are required to drive the system to the steady state.

Another huge drawback for MC methods in this context is that the algorithms postpone computation effort to the end of the episode, when all rewards for each state-action pair of the episode have to be calculated at once to update the state-action values accordingly. These calculations must be completed within one time step as the policy improvement step should be finished before the next episode starts and samples from the new policy are drawn.

### 2.3.6 SARSA

To evade the massive computational effort at the end of each episode, the next sections will tend to algorithms which are based on Temporal Difference (TD) learning. SARSA, short for State-Action-Reward-State-Action, was originally proposed in [45]. It is an on-policy RL algorithm which can be implemented on-line easily. SARSA uses for the sample generation

$$\hat{Q}(\check{\mathbf{z}}_k, \check{\nu}_k) = r_f(\check{\mathbf{z}}_k, \check{\nu}_k) + \gamma Q(\check{\mathbf{z}}_{k+1}, \check{\nu}_{k+1}) \tag{2.32}$$

an estimate for the future state-action values $Q(\check{\mathbf{z}}_{k+1}, \check{\nu}_{k+1})$. The update law is then given as

$$\begin{aligned} Q(\check{\mathbf{z}}_k, \check{\nu}_k) &\leftarrow Q(\check{\mathbf{z}}_k, \check{\nu}_k) + \alpha_K(\hat{Q}(\check{\mathbf{z}}_k, \check{\nu}_k) - Q(\check{\mathbf{z}}_k, \check{\nu}_k)) \\ &= Q(\check{\mathbf{z}}_k, \check{\nu}_k) + \alpha_K(r_f(\check{\mathbf{z}}_k, \check{\nu}_k) + \gamma Q(\check{\mathbf{z}}_{k+1}, \check{\nu}_{k+1}) - Q(\check{\mathbf{z}}_k, \check{\nu}_k)) \end{aligned} \tag{2.33}$$

which is executed every step. This approach distributes computation effort uniformly over all samples which is a huge advantage compared to MC approaches. In addition, SARSA is based on the Bellman equation deployed in dynamical programming. So, it utilizes the Markov property which connects individual state-action values together. Learning might therefore speed up in comparison with MC methods.

SARSA is called on-policy because the state-action values are updated using samples which are generated by the policy that is learnt itself. $Q(\check{\mathbf{z}}_{k+1}, \check{\nu}_{k+1})$ depends on the action $\check{\nu}_{k+1}$ that is selected by the policy [22, Chapter 6.4].

SARSA is known to converge in the tabular setting for decaying $\varepsilon$-greedy policies if the variance of the reward function is bounded and the step size parameter $\alpha_K$ adheres to [46, Theorem 1]

$$0 \leq \alpha_K \leq 1, \quad \sum_{K=0}^{\infty} \alpha_K \to \infty, \quad \sum_{K=0}^{\infty} \alpha_K^2 < \infty. \tag{2.34}$$

In addition, the state-action value of a specific state-action combination should only be updated if this state-action combination is actually observed and all state-action combinations should be visited infinitely many times.

Because SARSA is an on-policy algorithm, choosing a method for exploration, e. g. $\varepsilon$-greedy or softmax, strongly influences the algorithm's performance. Exploration also adds to variance in the update steps. If variance is mainly introduced by exploration in a deterministic environment, expected SARSA with the sample estimate

$$\hat{Q}(\check{\mathbf{z}}_k, \check{\nu}_k) = r_f(\check{\mathbf{z}}_k, \check{\nu}_k) + \gamma \sum_{\check{\nu}_{k+1} \in \mathcal{U}} p(\check{\nu}_{k+1} | \check{\mathbf{z}}_{k+1}) Q(\check{\mathbf{z}}_{k+1}, \check{\nu}_{k+1}) \tag{2.35}$$

with $p(\check{\nu}_{k+1}|\check{\mathbf{z}}_{k+1})$ from (2.28) is likely to improve its policy with fewer interaction steps compared to SARSA [47] as the information about the stochasticity brought in by the policy is incorporated in the update step. This comes at the cost of the summation over all control actions in one update step and is hence less relevant for the objective of this work where computation time is limited.

The basic SARSA version, on the other hand, can be implemented very efficiently. First, the new measurement $b_{k+1}$ must be rounded to a valid center $\check{b}_{k+1} \in \mathcal{B}$. Assuming these centres are stored in a sorted array, then finding the closest value takes a maximum of $\log_2(N_b) + 1$ memory accesses and comparisons. Next, it has to be determined whether to use the best action with the highest state-action value or a random action, i. e. a random number must be drawn from a uniform distribution. Linear Feedback Shift Registers (LFSRs) [48, Chapter 2] provide an efficient way to do so. Unfortunately, finding the best action requires a memory sweep over all state-action values $Q(\check{\mathbf{z}}_{k+1}, \cdot)$. Lastly, the update step, including the calculation of the reward function, takes eight multiplications and six additions. It is not necessary to load $Q(\check{\mathbf{z}}_k, \check{\nu}_k)$ from memory because this value can be buffered from the last update step. The bottleneck of this implementation is the memory sweep for determining the optimal control action. If memory is available abundantly but timing is critical, one could implement a second memory that holds the maximum state-action values and corresponding indices where the maximum is located for each state. Then, only during the writing back phase, it needs to be checked whether the new state-action value is larger than the old maximum, reducing the $N_u + 1$ memory accesses to three memory accesses (one to load the old maximum to be compared to the new one, one to write the new index and one to write the new maximum value in case a new maximum is found).

### 2.3.7 Q-learning

Q-learning [24] is the off-policy variant of TD learning. Its samples are generated by

$$\hat{Q}(\check{\mathbf{z}}_k, \check{\nu}_k) = r_f(\check{\mathbf{z}}_k, \check{\nu}_k) + \gamma \max_{\check{\nu}_{k+1}} Q(\check{\mathbf{z}}_{k+1}, \check{\nu}_{k+1}) \tag{2.36}$$

with the corresponding update law

$$\begin{aligned} Q(\check{\mathbf{z}}_k, \check{\nu}_k) &\leftarrow Q(\check{\mathbf{z}}_k, \check{\nu}_k) + \alpha_K(\hat{Q}(\check{\mathbf{z}}_k, \check{\nu}_k) - Q(\check{\mathbf{z}}_k, \check{\nu}_k)) \\ &= Q(\check{\mathbf{z}}_k, \check{\nu}_k) + \alpha_K(r_f(\check{\mathbf{z}}_k, \check{\nu}_k) + \gamma \max_{\check{\nu}_{k+1}} Q(\check{\mathbf{z}}_{k+1}, \check{\nu}_{k+1}) - Q(\check{\mathbf{z}}_k, \check{\nu}_k)). \end{aligned} \tag{2.37}$$

In contrast to SARSA, Q-learning always learns from the best known policy that maximizes the future discounted rewards $\max_{\check{\nu}_{k+1}} Q(\check{\mathbf{z}}_{k+1}, \check{\nu}_{k+1})$, regardless of what control action $\check{\nu}_{k+1}$ is actually taken. Hence, Q-learning is off-policy.

The convergence proof for Q-learning [24, 49] involves the same assumptions for the step size as in (2.34). However, while the convergence proof for SARSA makes additional assumptions regarding the exploration policy, this is not required for Q-learning as it is an off-policy algorithm. Nevertheless, each state-action must still be visited infinitely many times.

The maximization operation in the update law (2.36) has some drawbacks. In [50], the author shows that in some situations, Q-learning can lead to overestimation of the state-action values due to the maximization operation, slowing down the learning process immensely. Therefore, he proposes Double Q-learning, an algorithm which utilizes a second set of state-action values for each state-action pair. During each step, only one of the two sets is updated using the best action selected by the other set of state-action values. Simulation results show that this can reduce the overestimation issue with Q-learning.

From an implementation perspective, Q-learning is equivalent with SARSA in terms of computational effort. This is counter-intuitive as the update law of Q-learning involves a maximization operation while the SARSA update law does not. However, SARSA implicitly also requires the same maximization step upon determining the next control action $\check{\nu}_{k+1} = \arg\max_{\check{\nu}_{k+1}} Q(\check{\mathbf{z}}_{k+1}, \check{\nu}_{k+1})$ when using $\varepsilon$-greedy exploration. The maximization step does not have to be executed if a random action is selected, but the algorithm needs to fit into the short computation window even for the worst-case scenario. Thus, whenever SARSA with $\varepsilon$-greedy exploration meets the timing requirements, Q-learning is also applicable.

### 2.3.8 Simulation Results

In this section, the main results will be presented. First, the learning process of the model-free RL algorithms will be analyzed, followed by the model-based RL algorithm. Then, the learned control laws of the different algorithms are compared to the linear controller. Finally, the performance is evaluated for the nominal plant.

### Monte Carlo

In Figure 2.7, the undiscounted reward

$$\sum_{n=1}^{N} r_f(\check{\mathbf{z}}_n, \check{\nu}_n) \tag{2.38}$$

for each episode is displayed for both the first-visit and every-visit on-policy MC method. $K$ denotes the global time, increasing with every interaction with the unknown control plant. Every episode contains $N = 20$ samples. The state-space discretization parameters are $N_b = 50$ and $N_u = 30$, while the exploration rate $\varepsilon$ is reduced linearly over the learning process, starting with an initial value of 0.1. The solid curve illustrates the local median over 1,000 episodes. While both median rewards increase over the learning process, the first-visit method clearly outperforms the every-visit method. An explanation for this might lie in the transient behaviour during learning which is drawn in Figure 2.8. There, the build-up and the control input, normalized to the maximum input $u_{\max}$, is displayed for 20 samples for both every-visit and first-visit controllers. Note that because of $u_{k+1} = \nu_k$, both control actions and states are drawn in the lower plot.

As the uncontrolled plant shows bifurcations, where output levels alternate between two states, the controller experiences these state combinations frequently during training. Thus, same states are likely to occur during one episode, generating multiple samples for a given state-action pair which in turn cause a biased estimate for the episode. This hinders effective learning. While the every-visit controller clearly fails to stabilize the set-point, drawn dashed, the first-visit controller is able to bring the system to close proximity of the nominal set-point after $k = 12$ cycles. However, fluctuations in both build-up and control input remain thereafter, yielding an unacceptable performance.

As MC on-policy methods do not provide satisfying result, off-policy methods are considered next. Contrary to on-policy methods, where the first-visit method is superior to every-visit, the opposite effect can be observed for off-policy methods. Simulations show that off-policy first-visit methods do not learn efficiently due to two competing mechanisms. While off-policy methods learn primarily from episode ends backwards due to importance sampling, as discussed in Section 2.3.5, first-visit methods favor episode starts. So, importance sampling cuts off episode starts, while first-visit methods neglect samples drawn late in an episode as they will likely appear multiple times, leaving hardly any samples to learn from. Figure 2.9 shows the state-action space coverage rate after the learning process for first-visit and every-visit off-policy MC methods for both unweighted and weighted importance sampling. The state-action space coverage percentage is defined as the fraction of the state-action space from which at least one sample has been generated during the learning process. The coverage for first-visit methods is below 10 % for both the unweighted and weighted case, while more than three quarters of the state-action space have been encountered by the every-visit methods. Thus, choosing the first-visit/off-policy combination overwhelmingly inhibits learning and is not considered any further.

Figure 2.10 shows the episode rewards for off-policy every-visit MC methods. In this graph, weighted importance sampling is compared to the unweighted version. The weighted importance sampling algorithm shows a better performance as it is able to reach a mean episode reward of about $-5$ after training, while the unweighted algorithm reaches a mean

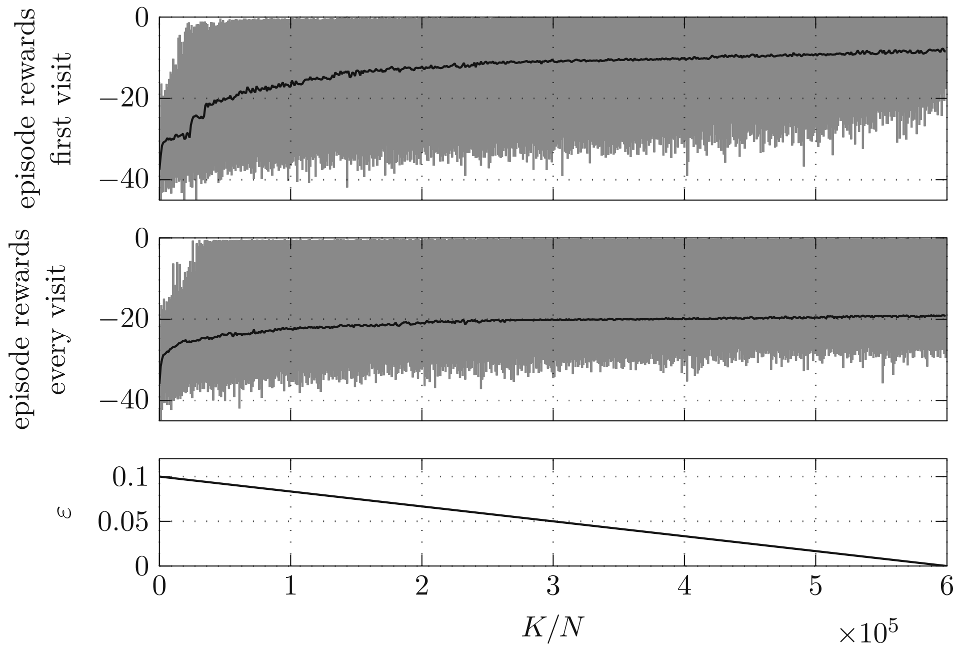

Figure 2.7: Episode rewards for on-policy MC simulation with linearly decreasing exploration, comparing first-visit method with every-visit method.

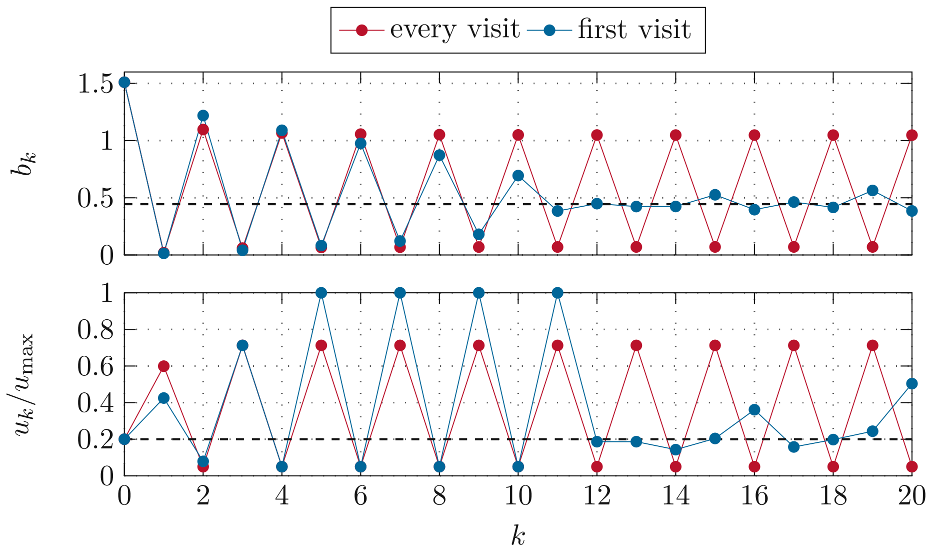

Figure 2.8: Transient closed-loop response with first-visit and every-visit on-policy MC controller, $x_0 = 1.1x_s, u_0 = u_s$.

value of about $-10$. In Figure 2.9, the weighted algorithm is shown to explore about 5 % more of the state-action space which might be a reason for the better performance. This agrees with common results that weighted importance sampling is preferred in practice [22, p. 105].

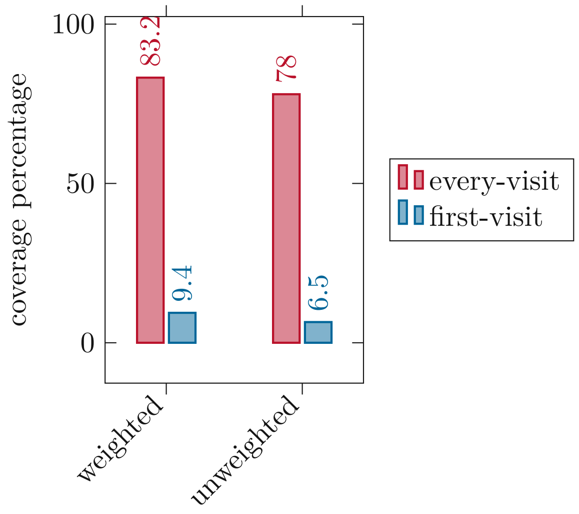

Figure 2.9: State-action space coverage for off-policy MC methods.

In Figure 2.11, the transient response of the closed loop is illustrated for both the unweighted and weighted importance sampling controller. The weighted importance sampling controller is able to bring the system to the steady-state point in about 15 cycles, which is shorter than the other controller. However, the system does not remain in the equilibrium point, as small spikes in both control input and build-up occur around $k = 38$ and $k = 47$. While this is a significant improvement to the on-policy methods, the results are still not satisfactory.

Thus, the next sections will tend to temporal difference algorithms.

### SARSA and Q-learning

Figure 2.12 shows the mean sample rewards, averaged over 500 samples, for the learning process of SARSA and Q-learning. The step size $\alpha_K = 1/N_K(\check{\mathbf{z}}_k, \check{\nu}_k)$ is in accordance with (2.34). The learning speed seems to be decoupled from the choice of $\gamma$ if $\gamma$ is selected in a range between 0.9 and 1 as the reward graphs for different discount factors overlap almost perfectly. SARSA and Q-learning both learn nearly identically fast in terms of the mean reward achieved. Thus, the discount factor $\gamma$ will be set to 0.99 throughout the remainder of this chapter if not stated otherwise.

Note that the learning process in Figure 2.12 is plotted in global time $K$, while in Figures 2.7 and 2.10, the learning process is plotted over $K/N$. Hence, SARSA and Q-learning need $N = 20$ times fewer training samples compared to MC methods to achieve a better result as measured by the rewards. This demonstrates the enormous advantage of dynamical programming and exploiting the inherent connection between states over

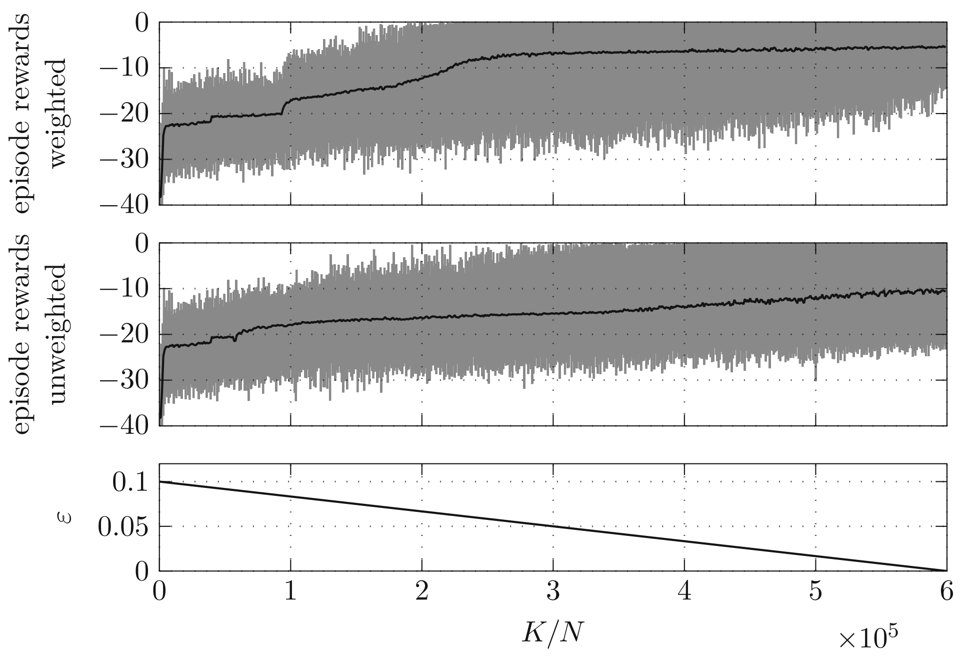

Figure 2.10: Episode rewards for off-policy MC simulation with linearly decreasing exploration, comparing weighted and unweighted importance sampling.

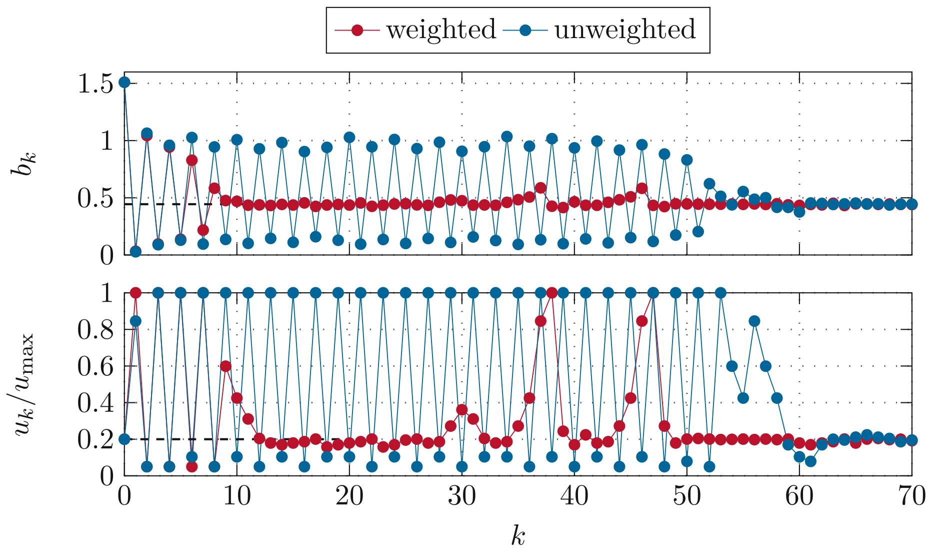

Figure 2.11: Transient closed-loop response with unweighted and weighted importance sampling MC controller, $x_0 = 1.1x_s, u_0 = u_s$.

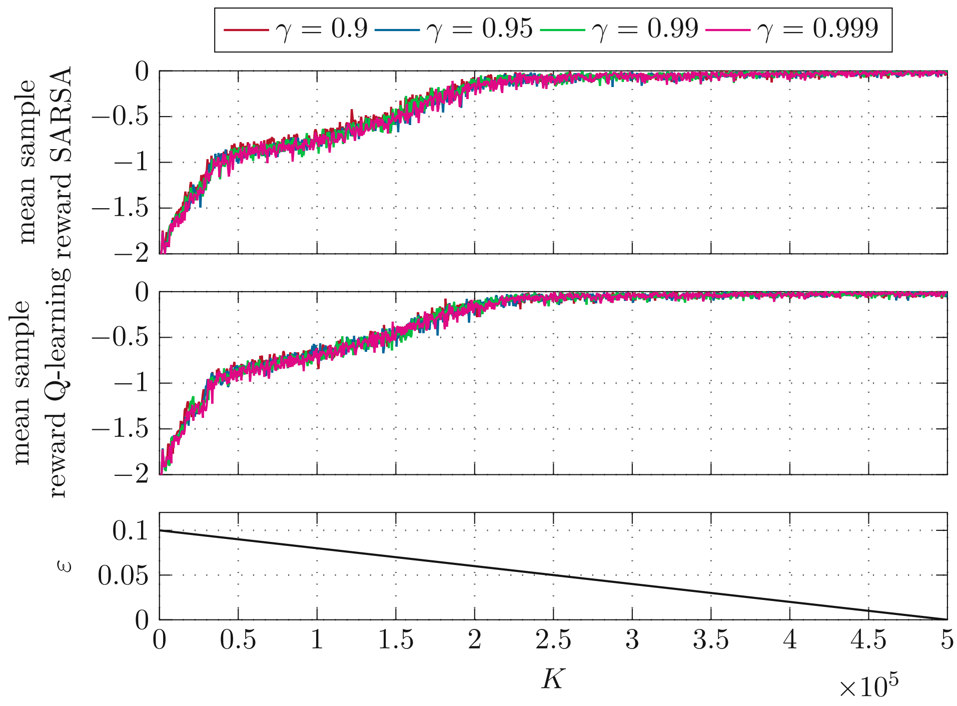

Figure 2.12: Episode rewards for SARSA and Q-learning simulation with linearly decreasing exploration, comparing discount factors.

MC methods for the given application.

However, while SARSA and Q-learning show very similar results when looking at reward functions only, they still perform very differently when investigating transient processes. Figure 2.13 shows the transient response of the closed loop for both the Q-learning and SARSA controller. While both controllers are able to bring the plant close to the steady state, only Q-learning manages to keep the system at the equilibrium point. For SARSA, every now and then, the transient curves show some spikes, e. g. at $k = 38$ or around $k = 54$. One explanation for this behaviour might be that SARSA is on-policy while Q-learning is off-policy. Therefore, SARSA learns the optimal policy among stochastic $\varepsilon$-greedy policies. Due to exploration required to find better actions, random actions are performed occasionally which are accounted for in the SARSA update step (2.33). Thus, SARSA prepares for a certain degree of randomness in its operation and accounts for this in the policy. If this stochastic policy is then turned into a deterministic policy, i. e. $\check{\nu}_k = \arg\max_{\check{\nu}_k} Q(\check{\mathbf{z}}_k, \check{\nu}_k)$ for frozen state-action values $Q(\cdot)$, these absent random actions will reduce performance. On the other hand, Q-learning is not confined to stochastic policies only because Q-learning - as an off-policy algorithm - can distinguish between the target policy, which can be deterministic, and the behavioral policy used to explore the state-action space. Thus, the conversion of the target policy to a deterministic version after training does not influence the algorithm's performance as much as for SARSA.

The issue of turning a stochastic policy into a deterministic one is illustrated in Figure 2.14. The initial state is selected to be the steady state. Then, the state-action values, which are retrieved after learning of Figure 2.12 is completed, are frozen and an $\varepsilon$-greedy policy is applied to the plant over the course of $N_{\text{tot}} = 20,000$ samples, recording the variance of the build-up $\sigma^2(b_k)$, the variance of the control input $\sigma^2(u_k)$ and the absolute mean reward per sample $1/N_{\text{tot}} \sum_{l=0}^{N_{\text{tot}}} |r_f(\mathbf{z}_l, \nu_l)|$ that Q-learning and SARSA receive, respectively. The absolute mean reward per sample is shown to allow for a logarithmic scale, only flipping the sign of the reward function as the latter is non-positive. In Figure 2.14, low absolute rewards are therefore desirable.

For $\varepsilon \gtrsim 0.04$, the policies of Q-learning and SARSA yield similar results, showing that SARSA's policy delivers close-to-optimal results in the stochastic setting. For small values of $\varepsilon$, however, Q-learning is able to reduce variance and increase rewards significantly, while the conversion of SARSA's stochastic policy into a deterministic one does not lead to such large improvements.

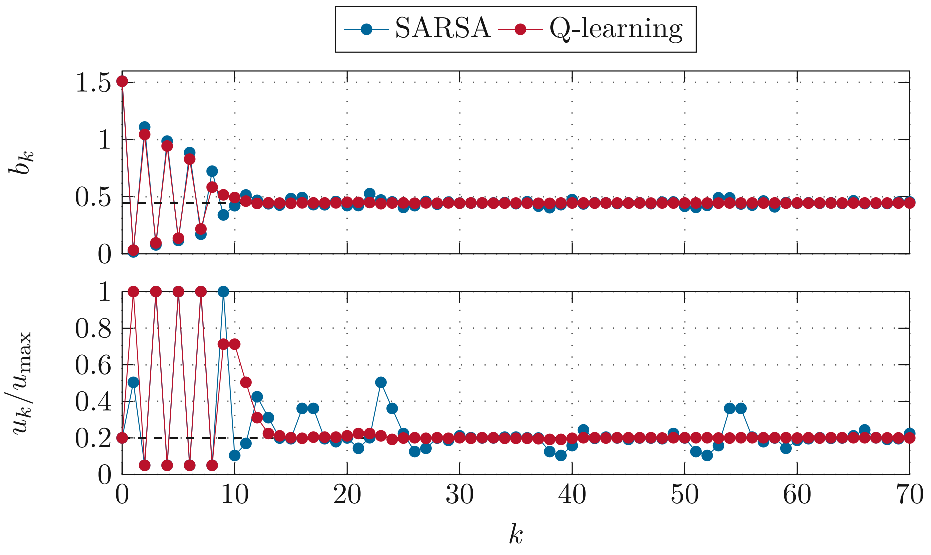

Figure 2.13: Transient closed-loop response with SARSA and Q-learning controller, $x_0 = 1.1x_s, u_0 = u_s$.

## Hyperparameter analysis

If plant parameters drift slowly over time, the approach in Figure 2.12 with step size $\alpha_K = 1/N_K(\check{\mathbf{z}}_k, \check{\nu}_k)$ and decreasing exploration rate $\varepsilon$ is not suitable as this will cause the learning to come to a halt after $\varepsilon$ is reduced to 0. Thus, this section investigates the results if the step size and exploration rate are chosen differently compared to the nominal scenario.

First, different choices for the exploration rate $\varepsilon$ will be discussed. Figure 2.15 shows the mean sample reward over the learning process for different $\varepsilon$ strategies. The nominal case, where $\varepsilon$ is linearly reduced from 0.1 to 0 over the learning process, is compared to an exponential reduction, where $\varepsilon = 0.995^{K/500}$, and two constant values $\varepsilon = 0.1$ and $\varepsilon = 0.05$. For both SARSA and Q-learning, the constant value 0.1 with the largest total exploration over the learning process shows the fastest improvements as measured by the

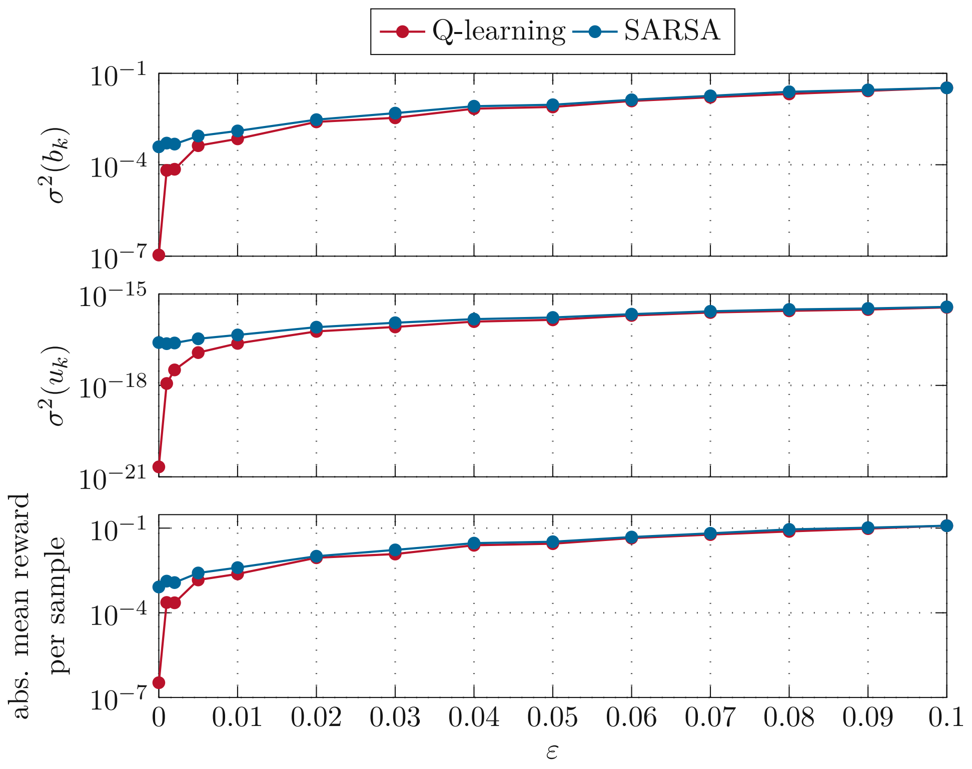

Figure 2.14: Variances of build-up and control input of SARSA and Q-learning for varying exploration rate $\varepsilon$ over 20,000 samples.

mean reward function. Likewise, the exponential reduction method learns the slowest, as the learning rate is reduced very quickly. Choosing a method for $\varepsilon$ also impacts the state-action space coverage drawn in Figure 2.16. In accordance with the learning speed, the constant exploration $\varepsilon = 0.1$ reaches the highest state-action space coverage after the learning process is finished, while the exponential reduction method visits the fewest state-action combinations at least once.

However, when comparing the performance of the closed loop with respect to variance of build-up and control input and the absolute mean rewards in Figure 2.17, the opposite results are obtained. The variances and rewards are recorded over $N_{\text{tot}} = 20,000$ samples for frozen state-action values and a deterministic control law, i. e. $\check{\nu}_k = \arg\max_{\check{\nu}_k} Q(\check{\mathbf{z}}_k, \check{\nu}_k)$.

The exponential reduction method reaches the smallest build-up and control variances and the largest rewards (smallest absolute rewards/smallest cost) over 20,000 samples, thus being able to stabilize the plant most efficiently around the desired set-point. The only exception to this is the control input variance for Q-learning where the exponential reduction method almost balances with the linear reduction method. On the other hand, the method with the largest exploration, i. e. constant $\varepsilon = 0.1$, performs the poorest in all cases. This illustrates the exploration/exploitation trade-off in RL. While methods that explore a lot reach large portions of the state-action space, they cannot find such good control strategies because of their lack of exploration of the currently best-known policy. Control strategies which do not explore as much have the opportunity to optimize this best-known policy more frequently, thus leading to better control results.

While the methods where $\varepsilon$ is reduced to 0, eventually, provide the most satisfying results with regard to the control objective, they will not handle slow parameter drifts as mentioned earlier. In this case, constant exploration might be a more suitable approach. Figure 2.17 shows that the method with 0.05 constant exploration achieves results comparable to the method of linear reduction and is hence a candidate to be used in on-line operation.

Next, the influence of the step size parameter $\alpha_K$ will be investigated. Figure 2.18 shows the mean rewards over 500 samples during the learning process for different choices of $\alpha_K$ and a constant exploration rate $\varepsilon = 0.05$. In particular, the nominal behaviour with $\alpha_K = 1/N_K(\check{\mathbf{z}}_k, \check{\nu}_k)$ is compared to a linear decreasing step size with initial values 0.1 and constant step sizes 0.05 and 0.1. As these variations do no satisfy (2.34), convergence is not guaranteed. This might be a reason why methods with constant or linearly decreasing step size exhibit large jumps in the mean rewards, especially during the initial phase.

Compared to Figure 2.15, training time is doubled in order to account for the longer learning time required to converge to the final mean rewards. The main reason for the slower learning time of the constant and linear decreasing methods is the lower initial step size. The first ten encounters of each state-action pair are assigned a step size $\alpha_K = 1/N_K(\check{\mathbf{z}}_k, \check{\nu}_k) \geq 0.1$ for the nominal case. Hence, initial encounters are weighted more compared to the constant step sizes, leading to a fast improvement at the start. In contrast, the other methods only learn with the small step size at the beginning which affects state-action values only very little, so more training samples are required for similar results.

In Figure 2.19, the state-action space coverage of these methods with different $\alpha_K$ strategies is shown. Compared to varying the exploration rate $\varepsilon$, changing the method of the step size influences the coverage less. Especially SARSA is hardly affected by the

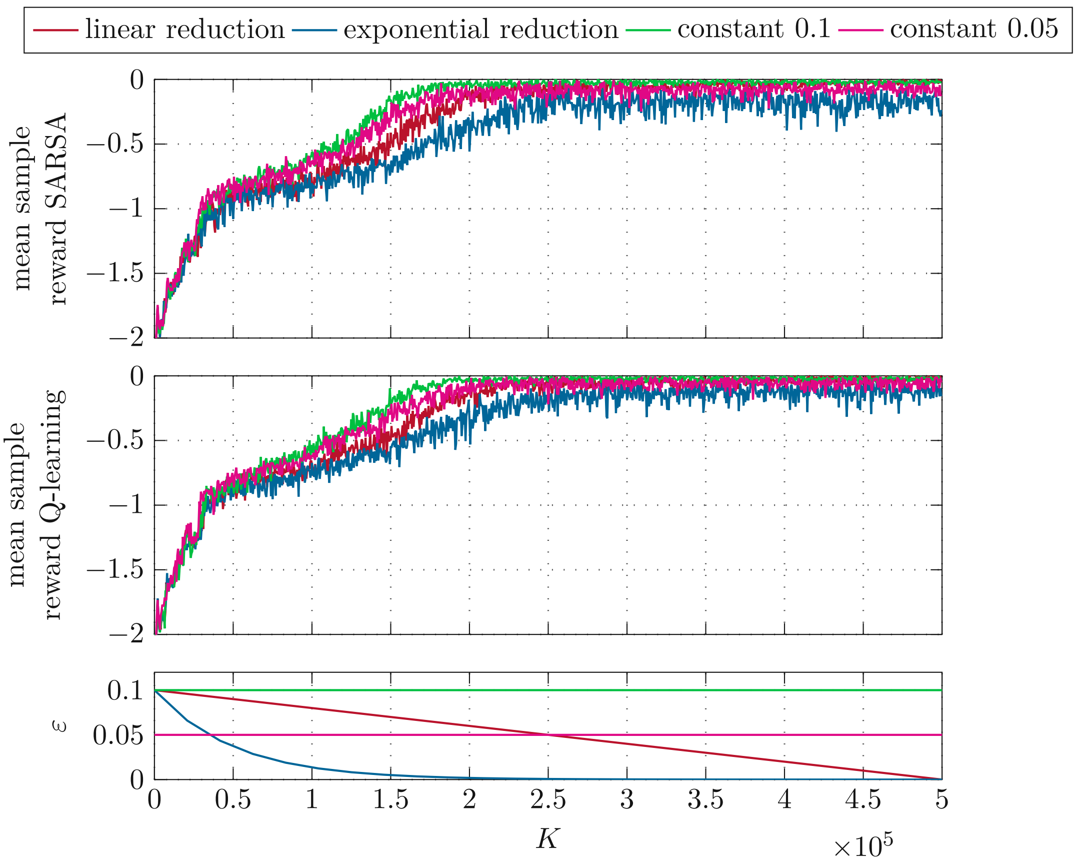

Figure 2.15: Influence of learning rate $\varepsilon$ on learning speed.

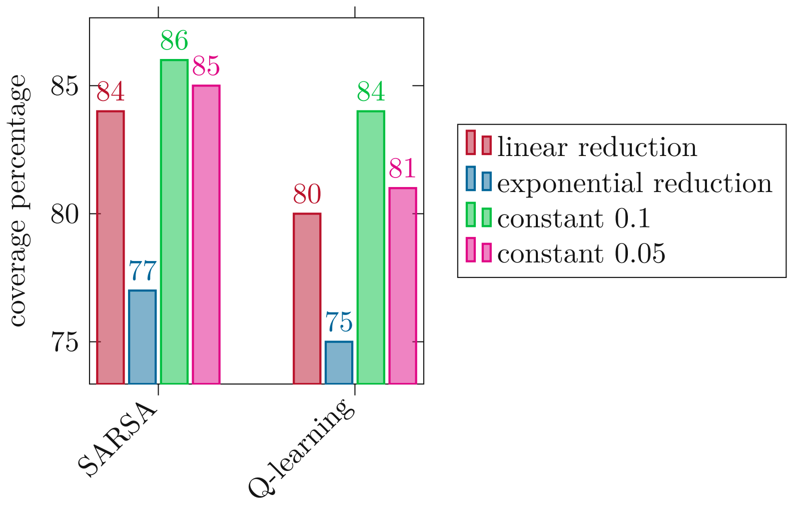

Figure 2.16: Influence of learning rate $\varepsilon$ on state-action space coverage.

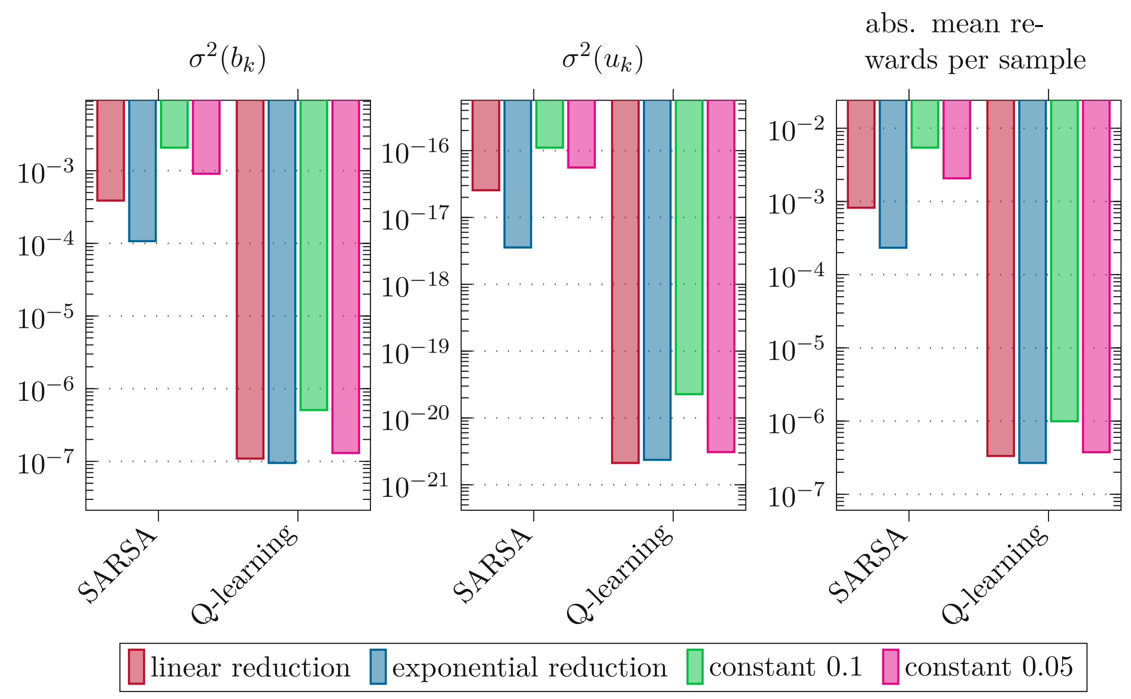

Figure 2.17: Influence of learning rate $\varepsilon$ on variance and rewards.

choice of the step size. In both cases, the coverage rate is higher compared to Figure 2.16 due to the double of amount of samples.

In Figure 2.20, the variances and mean rewards of all methods are shown for the same procedure as for Figures 2.14 and 2.17. While constant step sizes perform not so well for SARSA as they exhibit larger variances and costs compared to their decreasing counterparts, the results for Q-learning are largely independent of the choice of the step size.

Thus, a viable candidate for on-line RL is Q-learning with constant exploration rate $\varepsilon = 0.05$ and step size $\alpha_K = 0.1$.

### Q-value iteration

After several model-free RL algorithms have been studied, the focus will now shift to Q-value iteration (QVI), a model-based RL algorithm. As already mentioned in Section 2.3.4, quantization is an issue for QVI as information that is lost due to rounding cannot be recovered. Figure 2.21 shows the transient response of the closed loop controlled by two QVI algorithms with different quantization steps after the value iteration algorithm has converged. Both controller design and simulation are carried out on the nominal model without any noise or model-plant mismatch. Of all RL methods investigated so far, QVI with the fine quantization $N_b = 1000, N_u = 100$ manages to stabilize the plant most quickly, requiring only about ten samples. The controller with the coarse

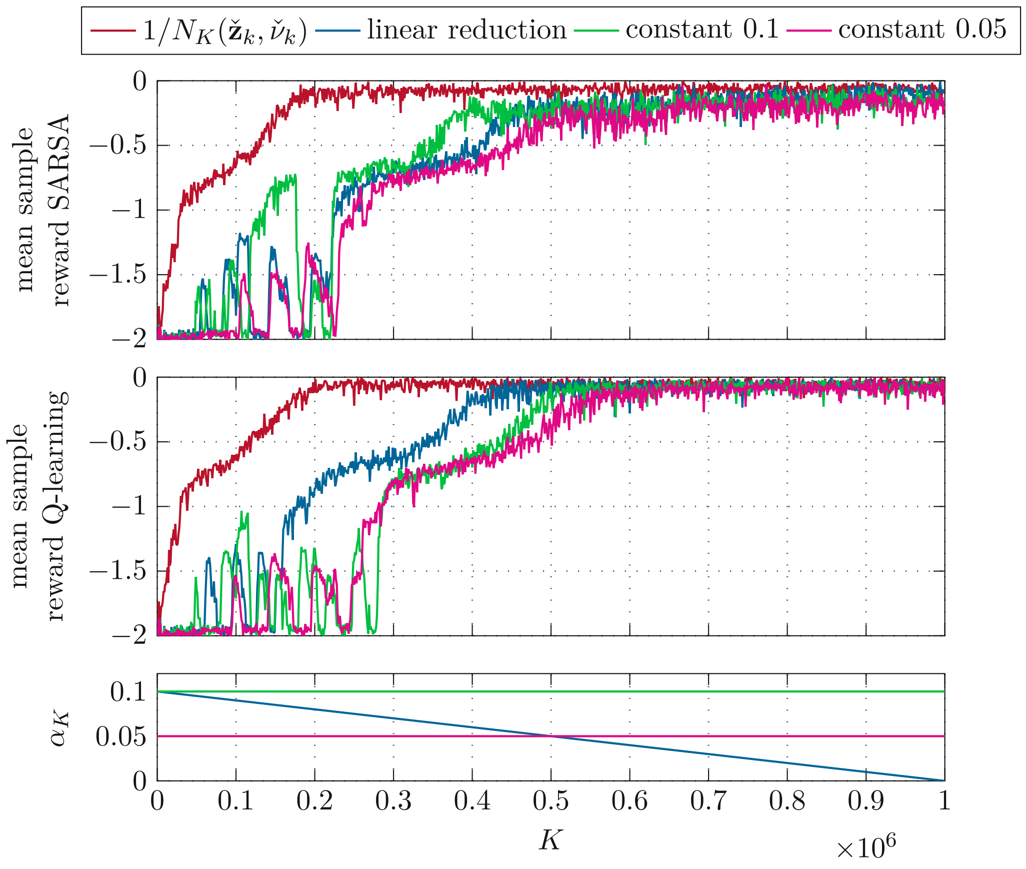

Figure 2.18: Influence of step size $\alpha_K$ on learning speed for $\varepsilon = 0.05$.

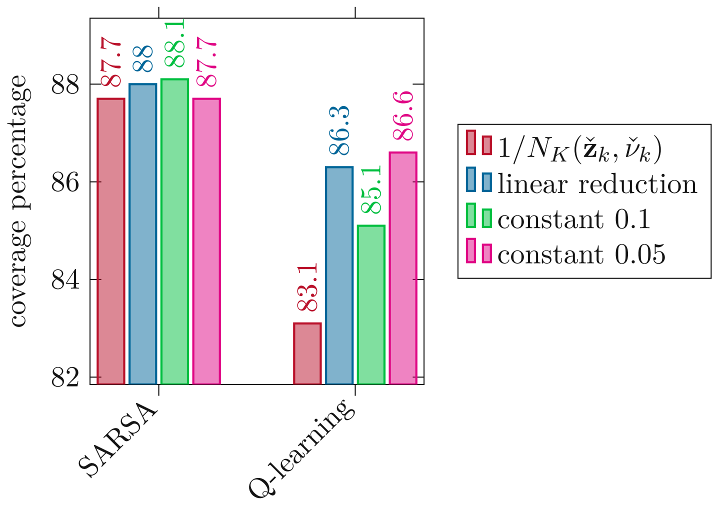

Figure 2.19: Influence of step size $\alpha_K$ on state-action space coverage.

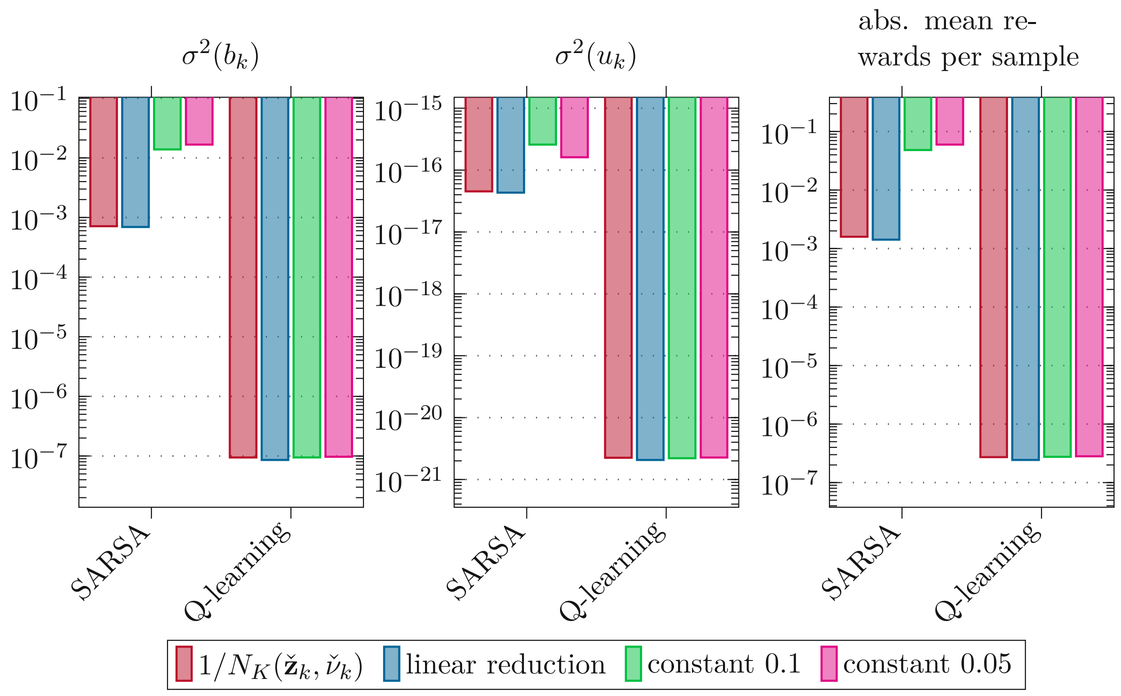

Figure 2.20: Influence of step size $\alpha_K$ on variance and rewards.

quantization $N_b = 50, N_u = 30$, on the other hand, produces a two-level limit cycle and does not stabilize the plant. In contrast, the model-free RL algorithms based on TD learning which all deal with the coarser quantization do not exhibit this behaviour. These methods all observe quantization-induced fluctuations between state connections and are able to incorporate this information into the control law. As comparable information is not available to QVI, the algorithm optimizes the control law, assuming that every transition starts exactly in the center of a tile, unknowingly enabling non-trivial limit cycles. However, if the initial state of the system is around the desired equilibrium point, QVI with the coarse quantization also manages to keep the plant in a small region around the steady state. This is due to the finer quantization in this area caused by the third order polynomial mapping.

Figure 2.22 shows the variances of build-up and control input as well as the absolute mean rewards per sample for the fine and coarse variant of QVI. In addition, the nominal Q-learning algorithm's performance is also shown again for comparison. A finer discretization clearly leads to better performance, lower variance and higher rewards. Thus, QVI with $N_b = 1000$ and $N_u = 100$ excels at all metrics. Surprisingly, Q-learning outperforms QVI with the same number of quantization levels even if the system always stays near the equilibrium point, indicating that information loss due to quantization is an issue even in the region of higher tile density.

The fact that QVI needs more discretization points to work sufficiently well is not as critical as for model-free algorithms. Calculations can be done beforehand, running

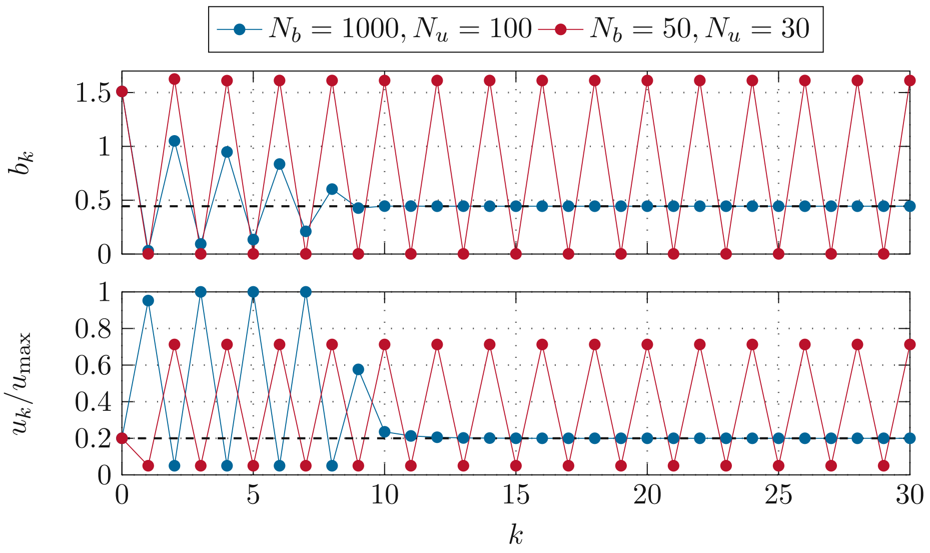

Figure 2.21: Transient closed-loop response with QVI controller, $x_0 = 1.1x_s, u_0 = u_s$.

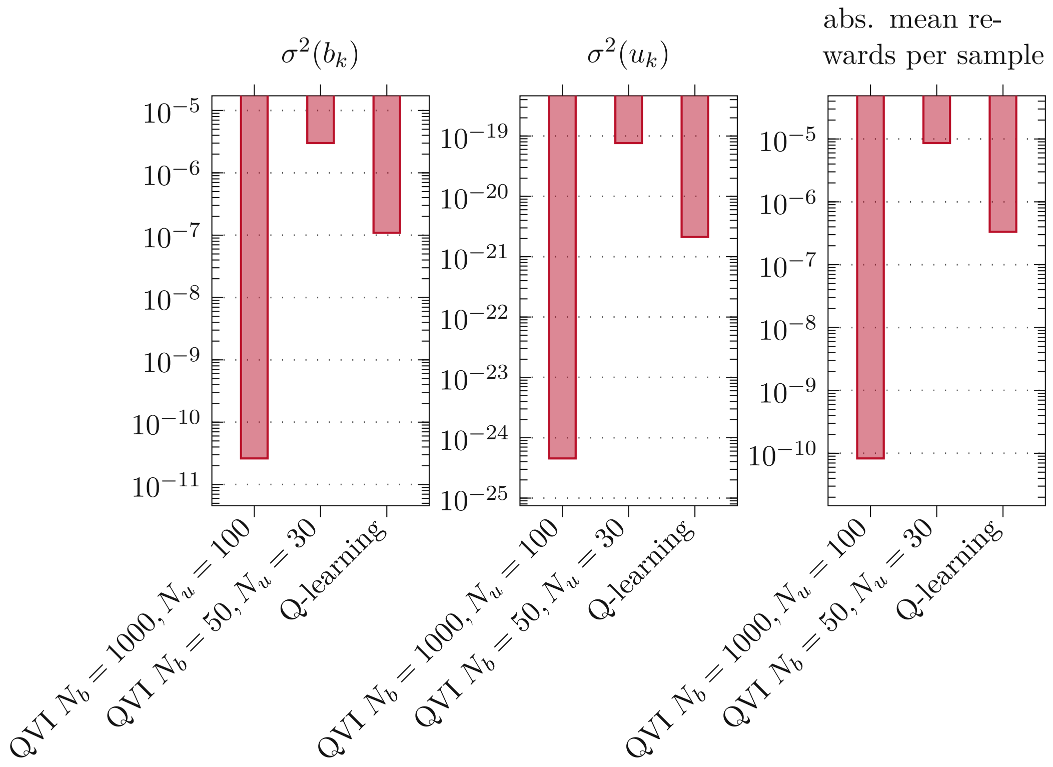

Figure 2.22: Variances and rewards for QVI.

iterations (2.20) over the state-action space. Model-free algorithms would take much longer to improve on their policy because the state-action space explodes in size, suffering from the curse of dimensionality [39, p. IX] even though the dimension for the state-action space investigated here is only three.

## Control law

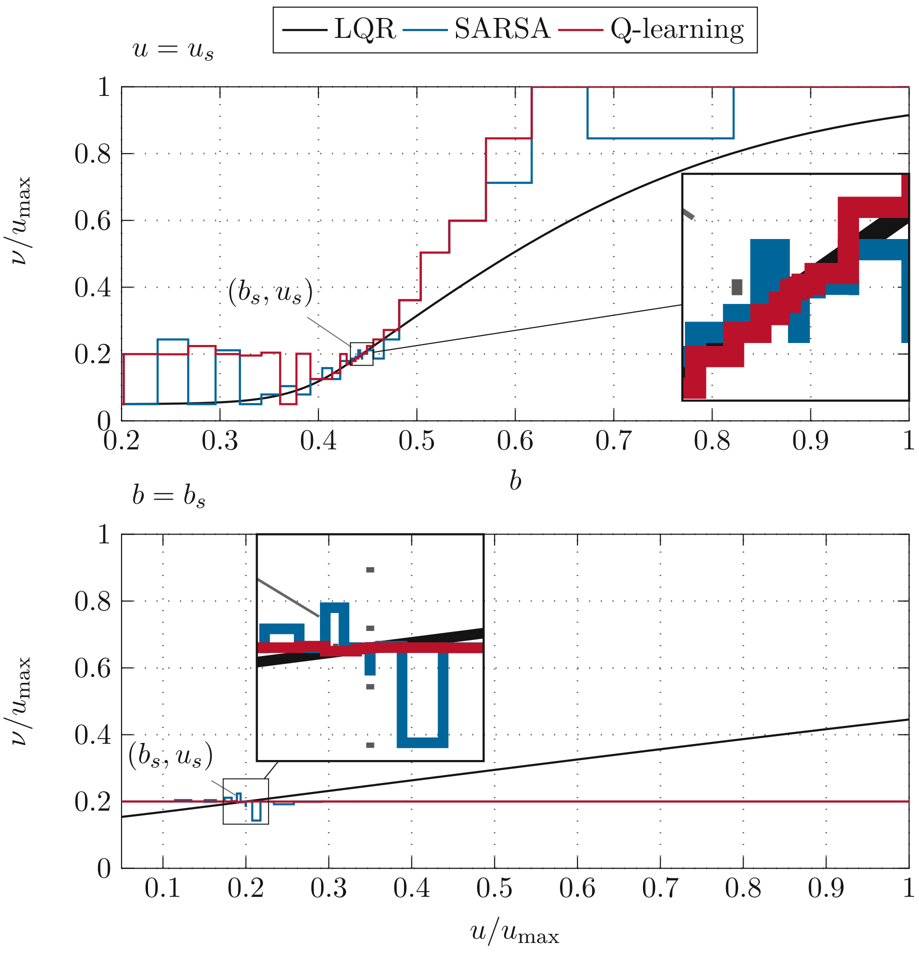

Figure 2.23: Control law for Q-learning, SARSA and linear controller.

Figure 2.23 shows the control law for the linear state feedback controller from Section 2.2 with a tanh-saturation function, as well as the control laws for nominal SARSA and Q-learning for two slices of the state-action space $\nu(b, u_s)$ and $\nu(b_s, u)$. When zooming in on the steady state for $\nu(b, u_s)$, the learned control laws exhibit a similar increase in $\nu$ over the local interval compared to the LQR. While the steps of Q-learning increase monotonically near the steady state, SARSA's steps include positive and negative steps which explains the noisy transient response in Figure 2.13. The other axis $\nu(b_s, u)$ seems to have been visited infrequently. An explanation for this might be that the steady state

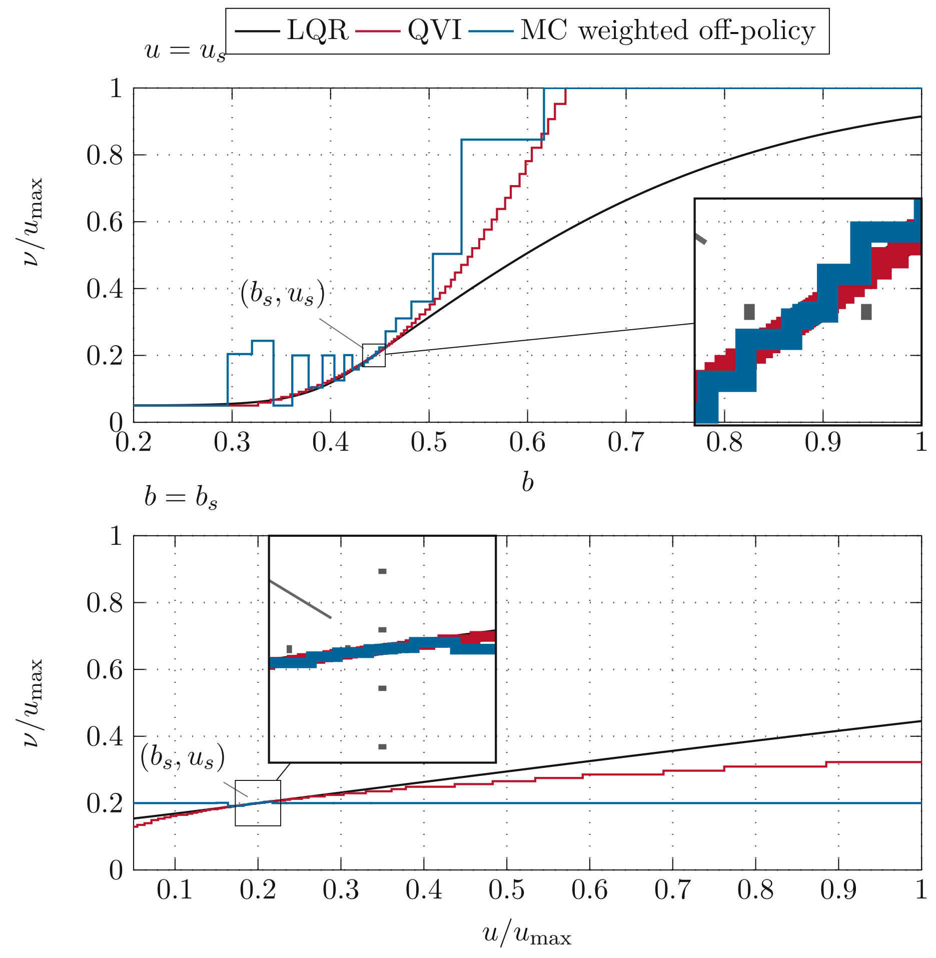

Figure 2.24: Control law for QVI, MC and linear controller.

build-up $b_s$ will most likely not occur if any other than the steady state control input has been applied to the plant due to the pulse-to-pulse dynamics which exhibits bifurcations, i. e. one build-up above steady state value will be followed by a build-up value below steady state.

Figure 2.24 shows the control law for QVI and the best performing MC variant, namely off-policy weighted importance sampling MC control. Near the steady state, the control law of QVI with $N_b = 1000$ and $N_u = 100$ overlaps almost perfectly with the control law from the LQR for $\nu(b, u_s)$. For larger values of the build-up, the control law of QVI increases more quickly than the one of the linear controller.

From the control law for QVI, an interesting observation can be made. Whenever the control law is not limited by the input constraints, i. e. for build-ups in the range [0.33, 0.65] and $u = u_s$, QVI designs the control law in a way that the (quantized) build-up is reduced to the steady state in about one step (near dead-beat behaviour). This is due to the fact that the reward for the build-up is weighted much more compared to the control input, i. e. $Q_{00}(b - b_s)^2 \gg R(\nu - u_s)^2$. An example is shown in Figure 2.25 where the maximum value for the build-up 0.65 is chosen such that control input constraints can just be met. Q-learning also manages to drive the system to the steady state in one step, while it takes a little bit longer for the linear controller (designed with the same weights for the cost function) to reach the steady state. Of course, if there is a mismatch between real values and rounded values for the build-up, the control laws obtained by the RL methods may take longer to stabilize the plant even if the build-up is within the range which would allow dead-beat behaviour.

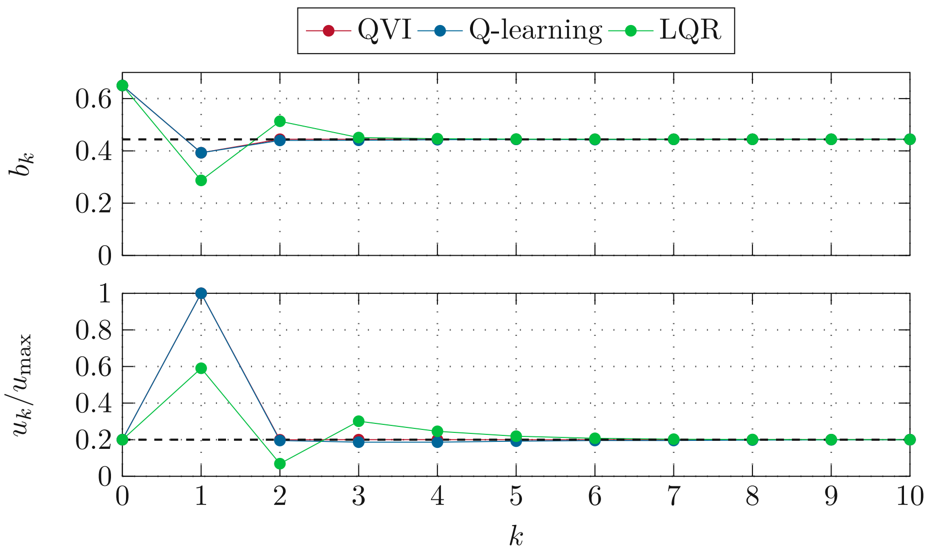

Figure 2.25: Transient closed-loop response for QVI, Q-learning and LQR controller, $b_0 = 0.65$, $u_0 = u_s$.

### 2.3.9 Summary

In this section, RL methods for stabilizing the pulse-to-pulse dynamics were introduced. Q-value iteration, the model-based algorithm, shows near dead-beat behaviour for fine quantization. SARSA and Q-learning, the model-free TD algorithms, both have similar learning times. SARSA, the on-policy algorithm, shows spikes in the closed loop occasionally. Q-learning's jumps in the control input are mainly caused by quantization and thus are very small around the set-point. Compared to the TD algorithms, Monte Carlo control methods are much slower to learn the policy because they do not bootstrap. Again, the off-policy methods clearly show better transient behaviour of the closed loop, indicating that off-policy methods might be better suited for stabilizing a set-point in general, as they are not confined to stochastic policies like on-policy methods.

## 2.4 Non-linear dead-beat controller

To benchmark the results of the reinforcement learning algorithms, this section shortly revisits another approach to controlling the plant based on the work of L. Tarra, yet unpublished. Afterwards, a neural network implementation of the control law is presented to stabilize the closed loop. This control strategy will be compared to the results of the reinforcement learning algorithms.

### 2.4.1 Control concept

The idea of this non-linear control law is to utilize stabilizing manifolds which exponentially drive the system into the steady state under certain conditions given by the Hadamard-Perron theorem. For the system, the simple structure of the linearized system dynamics around the steady state [18]

$$\Delta\mathbf{z}_{k+1} = \begin{bmatrix} \theta_1 & \theta_3 \\ 0 & 0 \end{bmatrix} \Delta\mathbf{z}_k + \begin{bmatrix} \theta_5 \\ 1 \end{bmatrix} \Delta\nu_k, \tag{2.39}$$

with poles at $\theta_1$ and 0, can be exploited with the linearization constant $\eta = \theta_3/\theta_1$ to define a subspace characterized by

$$\xi_k = \Delta b_k + \eta\Delta u_k = 0. \tag{2.40}$$

This corresponds to the linear space denoted by the eigenvector to the zero eigenvalue. This generalizes to the stable manifold around the steady state given by

$$\xi_k = \Psi(\Delta u_k) \tag{2.41}$$

in the non-linear case. The function $\Psi(\Delta u_k)$ is obtained by plugging in the desired behaviour, i. e. (2.41) being mapped onto itself under the system dynamics for $\Delta\nu_k = 0$, taking the absolute derivative, and solving the ordinary differential equation (ODE)

$$\frac{\mathrm{d}\Psi}{\mathrm{d}\Delta u_k} = \eta - \frac{\frac{\partial \tilde{f}}{\partial u}(b_s + \Psi(\Delta u_k) - \eta\Delta u_k, u_s + \Delta u_k, u_s)}{\frac{\partial \tilde{f}}{\partial b}(b_s + \Psi(\Delta u_k) - \eta\Delta u_k, u_s + \Delta u_k, u_s)}, \quad \Psi(0) = 0 \tag{2.42}$$

with $\tilde{f}(\cdot)$ from (2.8). The next control input $\nu_k$ is used to steer the system from an arbitrary point in state space onto the stable manifold around the steady state at the next time instant, making this controller a sort of non-linear dead-beat controller. Demanding this behaviour yields the implicit equation

$$\Psi(\nu_k - u_s) = \tilde{f}(b_k, u_k, \nu_k) - b_s + \eta(\nu_k - u_s) \tag{2.43}$$

which has to be solved numerically.

### 2.4.2 Nominal case

The control law for this non-linear controller is shown in Figure 2.26. In addition, the linear controller from Section 2.2 and the QVI results from Section 2.3.8 are also drawn. QVI and the dead-beat controller produce a very similar control law. For $u = u_s$, the control law of the dead-beat controller rises a little bit more slowly for increasing values of the build-up compared to QVI. For $b = b_s$, QVI and the non-linear controller align almost identically, only separated by a quantization step. Figure 2.27 shows the transient response of the closed loop with this dead-beat controller. This controller manages indeed to only use one control action $\nu_0 = u_1$ to reach the steady state, which is the result of the stable manifold locally being the eigenspace for an eigenvalue 0.

### 2.4.3 Adaptive LMS estimator

To increase robustness of the control law (2.43), two methods are pursued. The first one includes an adaptive control law for the estimation of the pump power (work of L. Tarra), the second the fitting of neural network parameters.

First, the adaptive law will be considered. As it was mentioned in Section 2.1, the function $\tilde{f}(\cdot)$ also depends on the pumping power, i. e. $\tilde{f}(b_k, u_k, \nu_k) = \tilde{f}(b_k, u_k, \nu_k, p_s)$ with the steady state pumping power $p_s$. To account for parameter drifts, the pumping power acts as a surrogate for all parameter uncertainties of the model and is estimated by a Least Mean Squares (LMS) update step

$$\Delta\hat{p}_{k+1} = \Delta\hat{p}_k + \mu_k \frac{\partial \tilde{f}}{\partial p}(b_k, u_k, \nu_k, p_s)\left( b_{k+1} - \tilde{f}(b_k, u_k, \nu_k, p_s) - \frac{\partial \tilde{f}}{\partial p}(b_k, u_k, \nu_k, p_s)\Delta\hat{p}_k \right) \tag{2.44a}$$

$$\mu_{k+1} = \mu_k \left( \overline{\mu} - \left( \frac{\partial \tilde{f}}{\partial p}(b_k, u_k, \nu_k, p_s) \right)^2 \mu_k \right) \tag{2.44b}$$

with some Taylor series approximations for both the pumping power estimate $\hat{p}_k = p_s + \Delta\hat{p}_k$ and the adaptive step size $\mu_k$ to simplify computation effort. $\overline{\mu} > 1$ determines the stationary step size and hence the sensitivity of the estimator.

### 2.4.4 Neural network implementation

The second approach is to have a neural network (NN) approximate the function $\tilde{f}(b_k, u_k, \nu_k)$. The strategy is as follows. First, in off-line mode, the parameters are

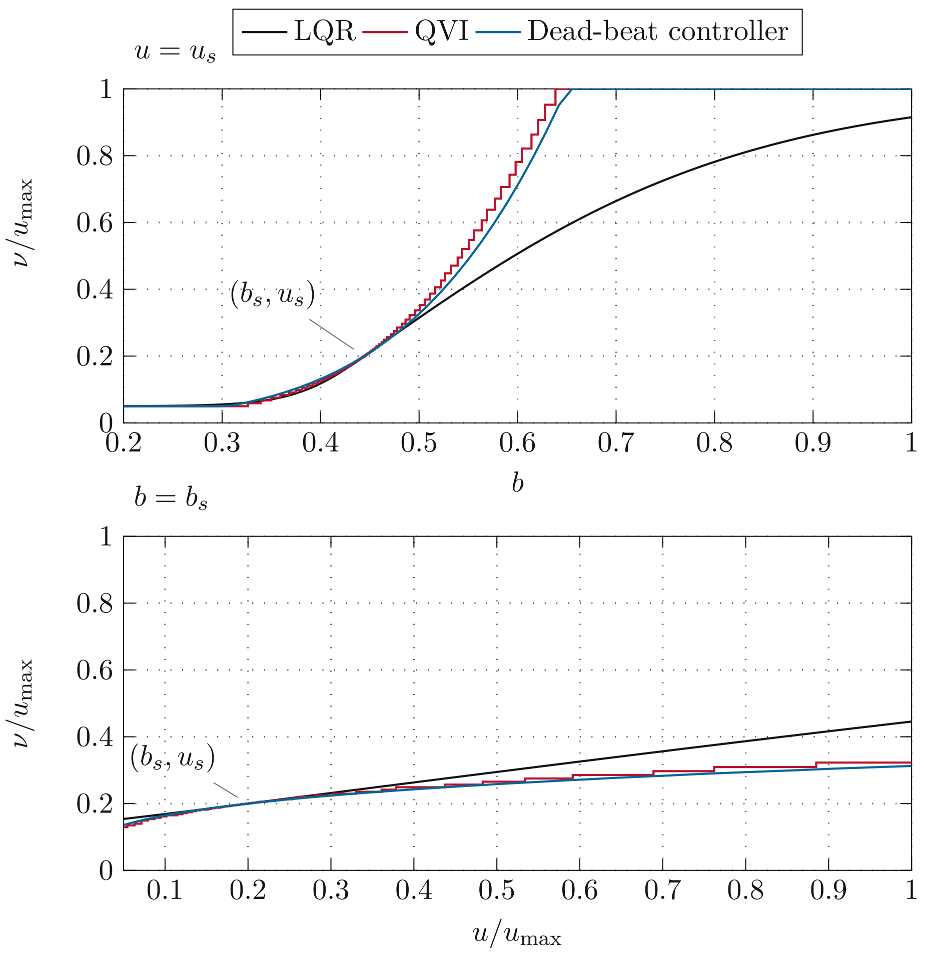

Figure 2.26: Control law linear controller, QVI and non-linear controller.

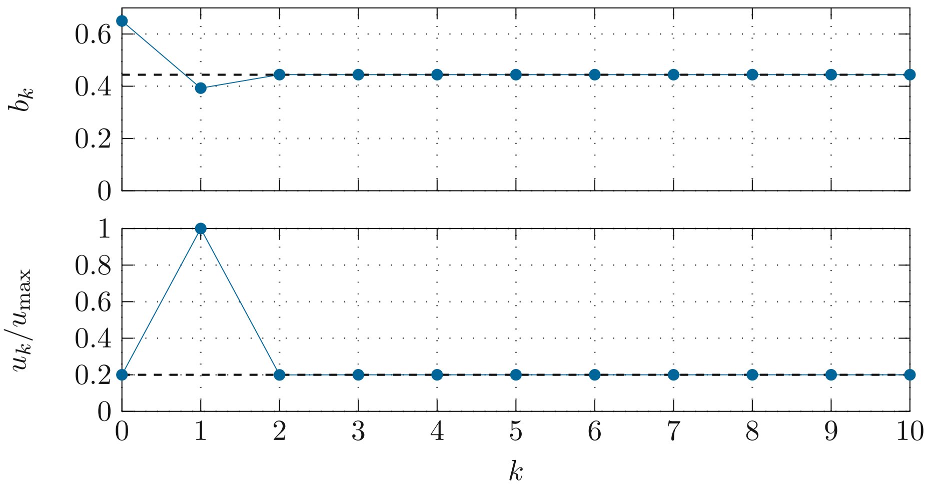

Figure 2.27: Transient closed-loop response for the non-linear controller on the nominal model, $b_0 = 0.65, u_0 = u_s$.

fitted to the nominal pulse-to-pulse dynamics. Then, these parameters are loaded onto the real-time controller, closing the loop with the control law (2.43), but replacing the nominal model $\tilde{f}(\cdot)$ with the NN approximation $\tilde{f}_{\mathrm{NN}}(\cdot)$. The controller records measurements for the build-up $b_{k+1}$ which will in general produce a mismatch between $b_{k+1}$ and the value $\tilde{f}_{\mathrm{NN}}(\cdot)$ predicted by the NN. As it is unlikely that the controller will have enough computation power to perform stochastic gradient descent [43, Chapter 8.3.1], i. e. to calculate new parameters after every measurement sample, (mini)batch updates [43, Chapter 8.1.3] are proposed to fine-tune the parameters. That means that the controller still evaluates the control law every step with the current parameters in the mean time. In addition, it calculates the gradient for some transitions and averages over the samples. If $\boldsymbol{\varphi}$ denotes the vector containing all parameters of the NN, the update is given as

$$L(\boldsymbol{\varphi}) = \sum_{l=k-N_f}^{k-N_f+N_{\mathrm{MB}}-1} \left(b_{l+1} - \tilde{f}_{\mathrm{NN}}(b_l, u_l, \nu_l, \boldsymbol{\varphi})\right)^2, \tag{2.45a}$$

$$\boldsymbol{\varphi} \leftarrow \boldsymbol{\varphi} - \alpha_{\mathrm{MB}} \nabla L(\boldsymbol{\varphi}) \tag{2.45b}$$

with the current time step $k$, minibatch size $N_{\mathrm{MB}}$, step size $\alpha_{\mathrm{MB}}$ and updating every $N_f$ steps.

The structure of the NN is illustrated in Figure 2.29. One interesting fact that helps constructing the NN is shown in Figure 2.28. The build-up $\tilde{h}(x, u_s)$ increases exponentially over a wide range of the gain $x$. Because $\tilde{f}(b_k, u_k, \nu_k)$ uses the inverse function $\tilde{h}^{-1}(x, u_s)$ to calculate the pulse-to-pulse dynamics, this motivates to include a logarithm in an additional input layer. The build-up is normalized to the steady state value $b_s$ and afterwards the natural logarithm is taken, cancelling the non-linearity of $\tilde{h}^{-1}(x, u_s)$. At the output, this normalization is undone by exponentiating the output of the last neuron.

The normalized input fluences $u_k$ and $\nu_k$ are very small in numbers and limited to $u_{\max} \ll b_s$, so the control inputs $u$ and $\nu$ are normalized to the maximum input fluence provided by the external pulse source. This overcomes the issue of ill-conditioned optimization problems [43, Chapter 8.2.1] which can inhibit effective parameter updates using stochastic gradient descents.

After the input normalization layer, two fully connected hidden layers with linear neurons and tanh activations functions follow, with 32 and 16 neurons, respectively. This yields a total of $3 \cdot 32 + 32 + 32 \cdot 16 + 16 + 16 \cdot 1 + 1 = 673$ parameters to optimize.

Figure 2.30 shows the typical update process of the NN in the closed loop. After stabilizing the plant to a constant build-up that is not equal to the nominal steady state build-up within the first update cycle $k < N_f = 50$, the parameters are updated every 50 steps, resulting in this step-like behaviour of the closed loop. The first update step corrects the parameters such that a steady state further away from the nominal equilibrium point is achieved. This is due to the fact that this update step contains only the first 20 measurements which are, in this case, mainly concerned with bringing the system to the steady state rather than keeping the system at the equilibrium point. Thus, the first update affects the approximation function of the NN in regions that are not so relevant for the actual control task. After the first update, however, all measurements which contribute to the parameter update step are near the steady state, leading to parameter

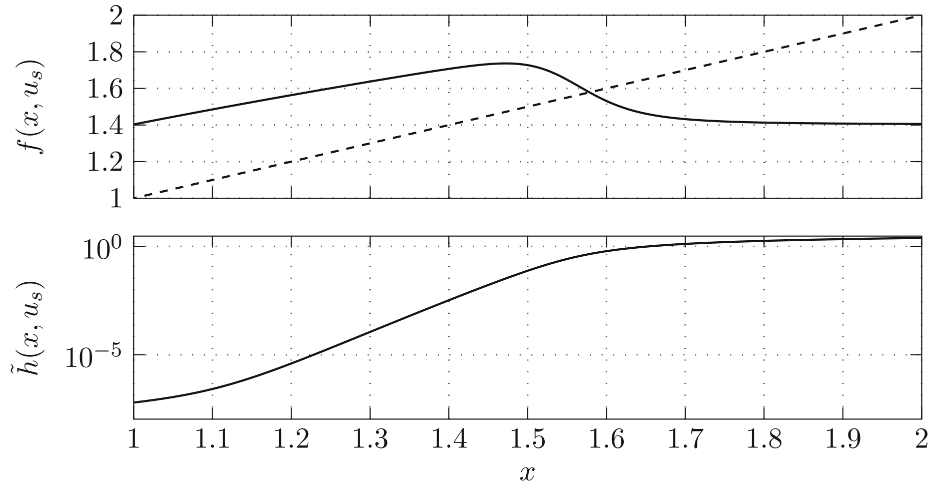

Figure 2.28: Pulse-to-pulse dynamics of gain (linear scale) and build-up (logarithmic scale).

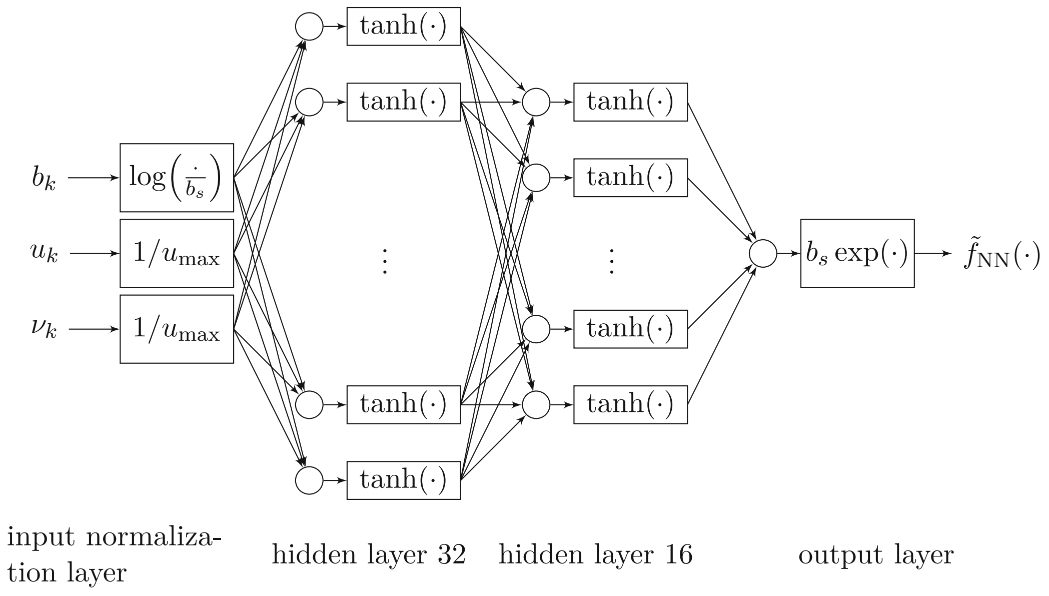

Figure 2.29: Structure of the neural network implementation.

updates which favor a good approximation of the pulse-to-pulse dynamics near the desired set-point. This reduces the approximation error of the NN, which results in small control errors $b_k - b_s$, eventually.

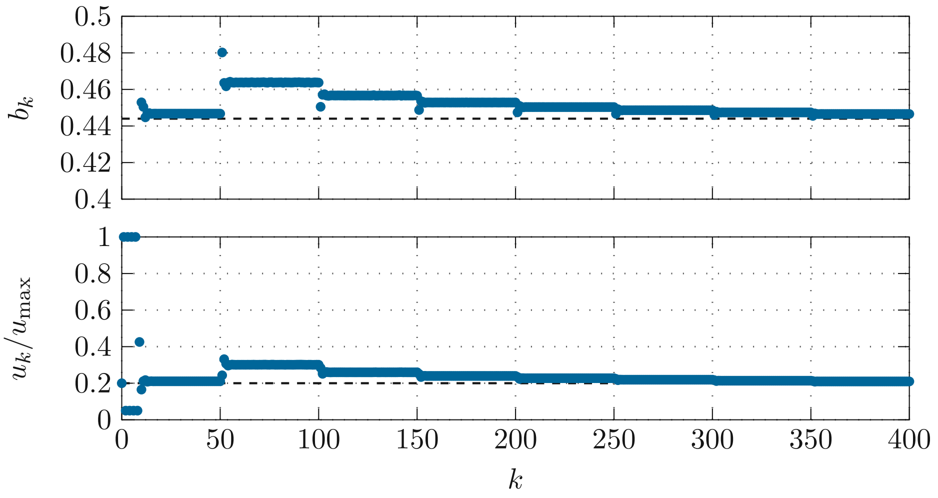

Figure 2.30: Transient closed-loop response for the non-linear controller with the NN approximation, $x_0 = 1.1x_s, u_0 = u_s, N_f = 50, N_{\mathrm{MB}} = 20, \alpha_{\mathrm{MB}} = 10^{-2}$.

## 2.5 Robustness analysis

Up until now, all methods have been evaluated for the nominal plant only. This section will now cover parameter uncertainties. In order to compare all methods systematically, only the transition cross-section $\sigma$, which occurs in the ODE for $f_{\mathrm{pump}}$ [18], will be varied. The corresponding value of the nominal plant is denoted as $\sigma_{\mathrm{nom}}$.

### 2.5.1 Reinforcement learning

As Q-learning shows the best results for the model-free RL algorithms, the effect of parameter uncertainties will be discussed for this model-free RL algorithm only.

Figure 2.31 shows the learning process for Q-learning for a constant model-plant mismatch $\sigma = 0.98\sigma_{\mathrm{nom}}$ if the initial state-action values match with the state-action values learned by the nominal plant. The learning process is performed with constant $\alpha_K = 0.1$ and constant exploration rate $\varepsilon = 0.05$ to allow the controller to constantly adapt to a potentially new environment with different or drifting parameters. Figure 2.33 shows the corresponding transient responses of the closed loop. The nominal controller with the control law fitted to the nominal plant manages at first to bring the system close to the steady state. However, it fails to hold the system there. On the other hand, the learned controller with adapted control law after the learning process fulfils this task well. Occasionally, some minor deviations from the steady state can be observed. These result from the coarser quantization. In Section 2.3.2, the quantization levels were designed to have a large density around the desired set-point, enabling fine distinctions between control actions around the set-point. If the actual set-point does not align with the nominal one, then the RL algorithm has to deal with coarser quantization, resulting in noisier control signals.

One option to mitigate this effect is to use interpolation between the control laws advised by neighbouring states after learning is completed and state action values are frozen. The round operator in (2.15) only rounds to the nearest tile center. For interpolation, the second closest tile centers in $b$ and $u$ direction are also utilized to refine the control law. First, the second best quantization levels

$$\check{\xi} = \mathrm{round}_2(\xi), \tag{2.46a}$$

$$\mathrm{round}(\xi) = \check{\xi}_i,\ i = \arg \min_{i=0,\dots,N_\xi-1} |\xi - \check{\xi}_i| \tag{2.46b}$$

$$\mathrm{round}_2(\xi) = \check{\xi}_j,\ j = \arg \min_{\substack{j=0,\dots,N_\xi-1 \\ j\neq i}} |\xi - \check{\xi}_j| \tag{2.46c}$$

are determined for both build-up $b$ and control input $u$. Then, the discrete control action for the best fitting tiles $\check{\nu}_{11} = \check{\nu}(\mathrm{round}(b), \mathrm{round}(u))$, $\check{\nu}_{21} = \check{\nu}(\mathrm{round}_2(b), \mathrm{round}(u))$ and $\check{\nu}_{12} = \check{\nu}(\mathrm{round}(b), \mathrm{round}_2(u))$ are utilized to compute an interpolated control input

$$\nu = \frac{1}{2}((1-\theta_u)\check{\nu}_{11} + \theta_u\check{\nu}_{12}) + \frac{1}{2}((1-\theta_b)\check{\nu}_{11} + \theta_b\check{\nu}_{21}) \tag{2.47}$$

with

$$\theta_b = \frac{b - \mathrm{round}(b)}{\mathrm{round}_2(b) - \mathrm{round}(b)} \tag{2.48a}$$

and

$$\theta_u = \frac{u - \mathrm{round}(u)}{\mathrm{round}_2(u) - \mathrm{round}(u)}. \tag{2.48b}$$

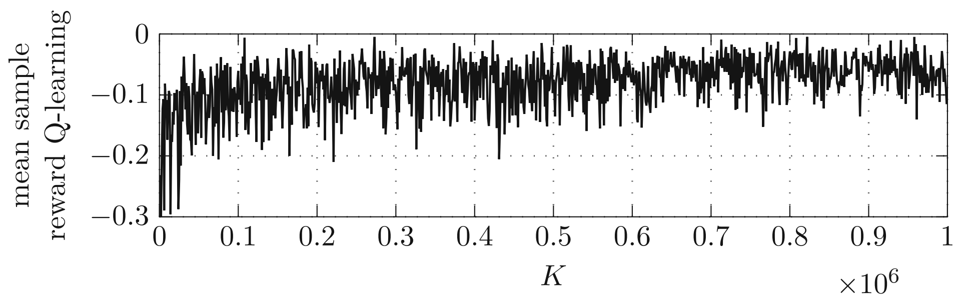

Figure 2.31: Mean rewards for Q-learning $\alpha_K = 0.1, \varepsilon = 0.05$ with model-plant mismatch $\sigma = 0.98\sigma_{\mathrm{nom}}$, starting with learned Q-values from the nominal plant.

The improvement for this modified control approach is significant. While the variances of build-up and control input for Q-learning with discrete control actions in Figure 2.32 are substantially higher compared to the nominal cases in Figures 2.17 and 2.20, the variances for Q-learning with linear interpolation reach the same levels as for the nominal plant.

While this seems promising at first, it has to be stressed at this point that while utilizing interpolation, no learning updates must be performed as an action cannot be associated with one tile only. Simulations show that if linear interpolation and RL updates are executed simultaneously, the closed loop shows inferior results compared to Q-learning with discrete control actions only. Hence, one feasible strategy might be to alternate between learning phases, where state-action values are updated to fit to slowly varying parameters and exploitation phases, where the optimal control action advised by the state-action values is refined with linear interpolation.

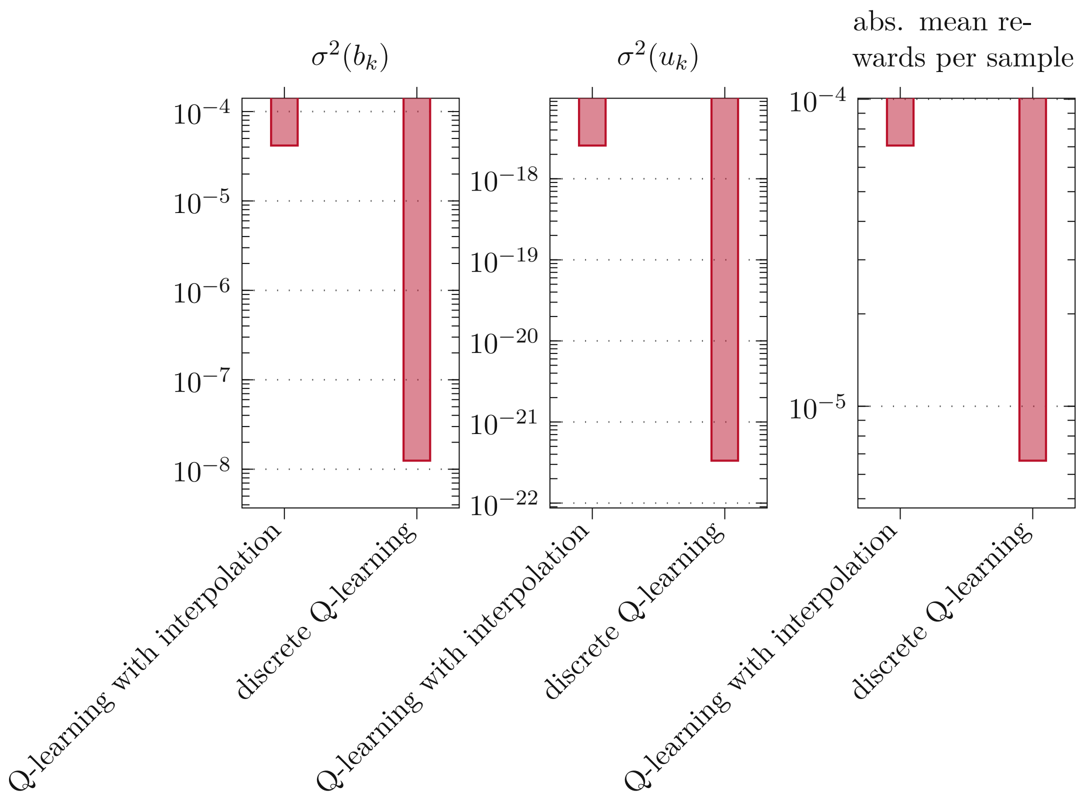

Figure 2.32: Performance for Q-learning with and without interpolation for model-plant mismatch $\sigma = 0.98\sigma_{\text{nom}}$.

## 2.5.2 Non-linear dead-beat controller

In Figure 2.34, the closed loop with the non-linear controllers is displayed for a given model-plant mismatch $\sigma = 0.95\sigma_{\text{nom}}$. While the NN controller stabilizes the plant after the first update at $k = 50$, the adaptive controller takes until $k = 280$ to suppress the bifurcations because it takes longer until the estimate for $\Delta\hat{p}_k$ changes due to a small initial value for $\mu_k$. Both variants do eventually reach a similar value for the build-up.

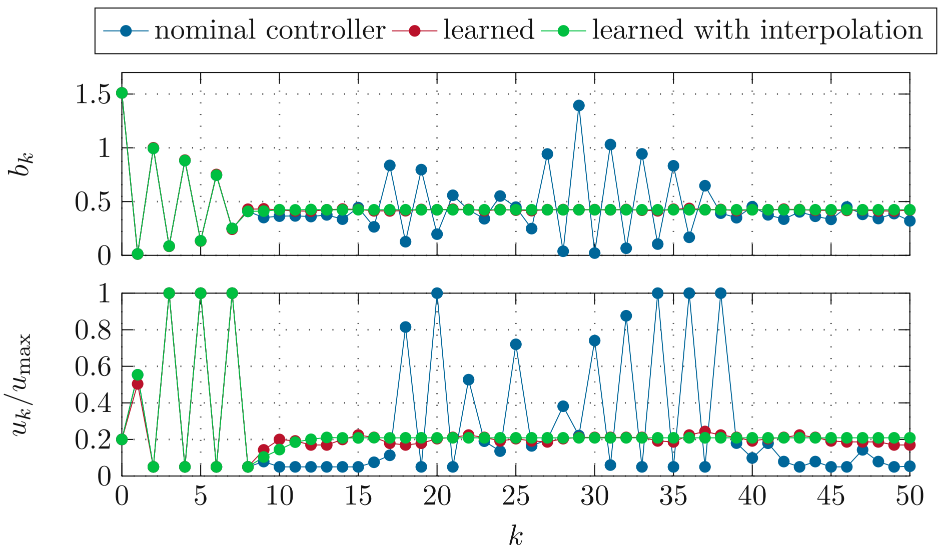

Figure 2.33: Transient closed-loop response for Q-learning before and after learning with model-plant mismatch $\sigma = 0.98\sigma_{\text{nom}}$, $x_0 = 1.1x_s$, $u_0 = u_s$.

### 2.5.3 Comparison between controllers

Figure 2.35 shows the region of the varying uncertain parameter $\sigma$ where each control approach is able to successfully stabilize the plant without undergoing limit cycle behaviour. LQR, QVI and nominal Q-learning are fixed control laws. QVI cannot be easily adapted on-line to learn new state-action values because $N_b$ and $N_u$ are too large to swiftly explore the entire state-action space. The other three methods are able to change their control law adaptively.

The two methods based on the non-linear dead-beat control approach manage parameter uncertainties well. The control law with the adaptive LMS update is practically only limited by the control input constraints, i. e. as long as the desired steady state build-up lies within the range which can be realized by the control input constraints, the closed loop remains stable. The NN is only limited by the lower control input bound which is reached for $\sigma = 1.05\sigma_{\text{nom}}$. When the steady state control input lies above $0.8u_{\max}$, the NN law shows bifurcations, limiting $\sigma$ to about $0.93\sigma_{\text{nom}}$.

The LQR and QVI exhibit similar performances when it comes to handling parameter uncertainties, but are not able to cover such a wide range of varying $\sigma$. Nominal Q-learning with the control law learned for the nominal plant lags behind in this regard, indicating that fine quantization crucially influences stability in the presence of parameter uncertainties.

Adaptive Q-learning, where the state-action value are continuously updated, is able to push the stabilizing regions for $\sigma$ further and exceeds both QVI and the LQR. Nevertheless, compared to the non-linear dead-beat control laws, Q-learning takes longer to update its control law. While in Figure 2.34, a steady state can be achieved after around 400 samples, it takes Q-learning a staggering 50,000 samples to significantly improve the policy.

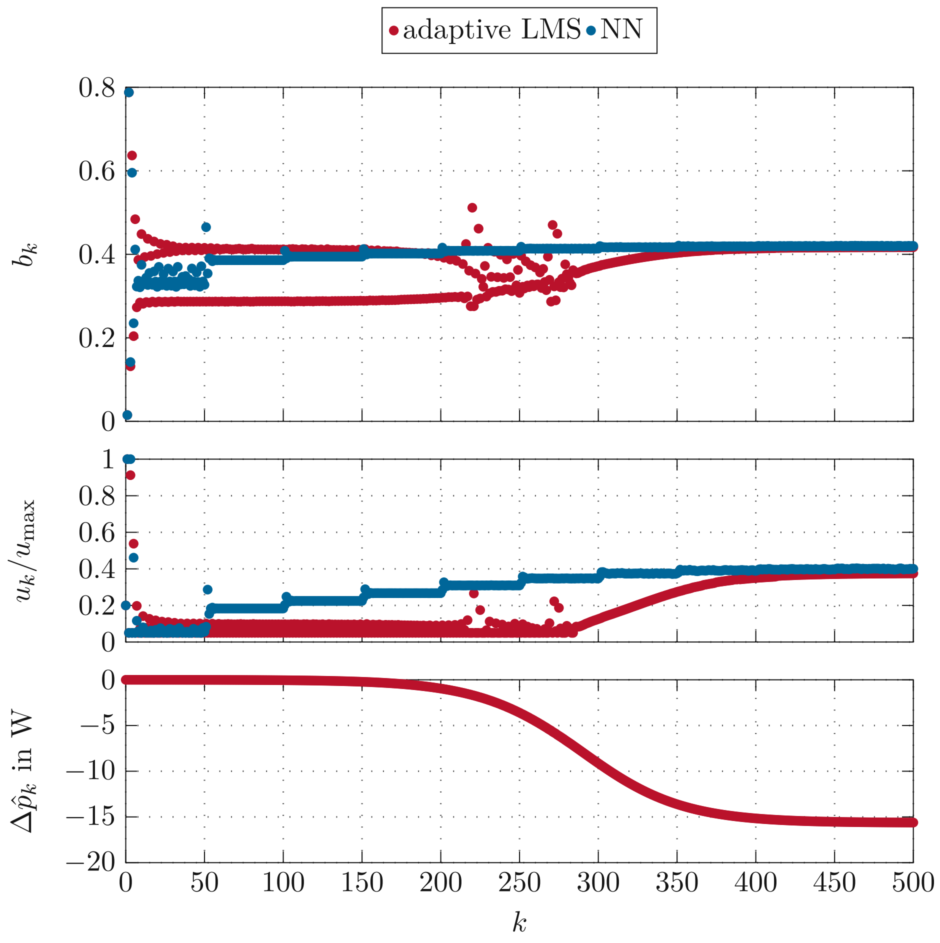

Figure 2.34: Transient closed-loop response controlled by non-linear controllers with adaptive pump power estimation and NN approximation for model-plant mismatch $\sigma = 0.95\sigma_{\text{nom}}$, $x_0 = 1.1x_s$, $u_0 = u_s$, $\overline{\mu} = 1.03$, $N_f = 50$, $N_{\text{MB}} = 20$, $\alpha_{\text{MB}} = 10^{-2}$.

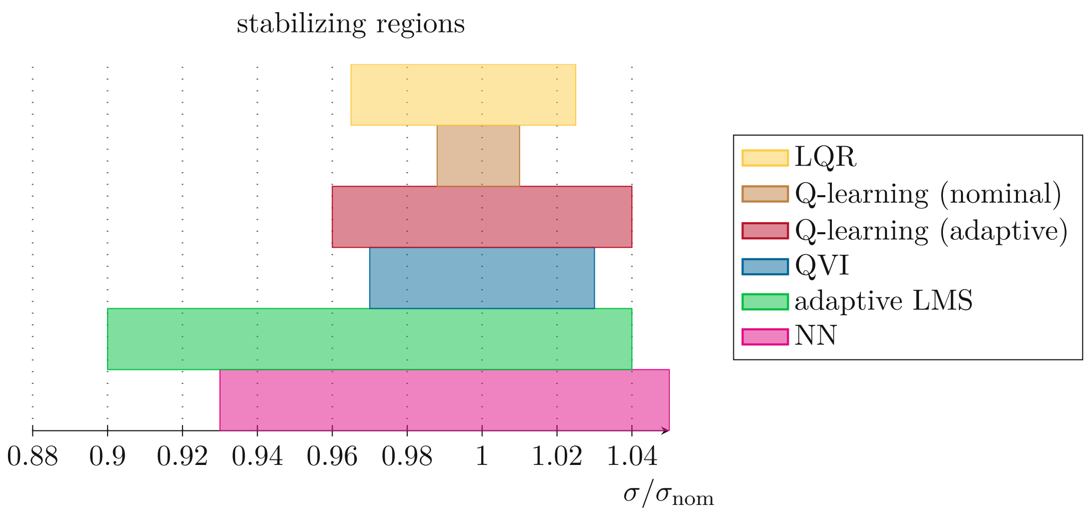

Figure 2.35: Stabilizing regions for the control algorithms for varying $\sigma$.

## 2.6 Summary

In this chapter, tabular-based RL methods were presented to stabilize the pulse-to-pulse dynamics of a regenerative amplifier. Q-value iteration is model-based and can produce dead-beat behaviour for the nominal plant, but requires a fine state-space discretization which makes it less applicable for on-line operation. For the model-free algorithms, off-policy methods outperform their on-policy counterparts consistently, making them a good choice for the given control task. Q-learning is the best-suited RL algorithm for on-line operation. Although Q-learning is able to adapt to changing system parameters, this takes usually very long compared to the non-linear controller based on the dead-beat approach and stable manifolds. While these adaptive controllers can handle parameter uncertainties swiftly and are able to stabilize a larger range of model-plant mismatches, they require lots of model information. Q-learning, on the other hand, is able to stabilize the plant without any prior model knowledge.

# 3 Cavity Dumping

This chapter covers the control of a pulsed laser system which utilizes cavity dumping for pulse generation. Section 3.1 presents the main models used in this work, based on [20, 21]. Section 3.2 investigates the same reinforcement learning as in Chapter 2, applied to this system. Section 3.3 investigates the non-linear controller of Section 2.4, while Section 3.4 deals with parameter uncertainties and examines the robustness of the RL controllers before the chapter is summarized in Section 3.5.

## 3.1 Model

The basis for the simulations conducted in Section 3.2 is the pulse-to-pulse dynamics model of a Nd:YAG laser system with a linear cavity which is published in [20]. The active medium is continuously pumped. Switching is performed with a Pockels cell where the reflection index $R(t)$ acts as a control input to generate pulses in the upper kilohertz regime. The model describes the evolution of the population inversion $N(t)$ and the intra-cavity power $P_j(t)$ of each longitudinal laser mode $j = 1, 2, ..., M$. In addition, the differential equation for $P_j(t)$ incorporates stochastic noise terms which can be approximated by a compound Poisson process, accounting for spontaneous emission.

To lessen the influences of these stochastic effects on the pulse-to-pulse dynamics, the authors suggest in [21] to use prelasing to stabilize the evolution of the intra-cavity power above noise level at the decision time. More specifically, the control input is not given as the reflection coefficient of the Pockels cell, but the time at which the reflection coefficient is changed from prelase level $R_{\text{pl}}$ to the maximum reflection $R_{\max}$. Due to a fixed switching frequency $f_{\text{sw}}$ and non-negative high-time of the reflection coefficient, the control input is limited by lower and upper bounds $u_{\min}$ and $u_{\max}$, respectively.

In the following, this work distinguishes between two different models:

- full stochastic model,
- average (deterministic) model.

The full stochastic model simulates a random compound Poisson process individually for every longitudinal mode $j$ every time it is executed. The average deterministic model replaces the stochastic noise term in the differential equation for $P_j(t)$ with its expected value and averages over modes to arrive at a single-mode approximation of the full stochastic model [21].

For the full stochastic model, the pulse-to-pulse dynamics

$$N_{k+1} = f_{\text{fs}}(N_k, u_k, P_k), \tag{3.49}$$

with the population inversion after the $k$-th pulse $N_k$, the sum over all modes of the intra-cavity power $P_k$ at the moment before switching to $R_{\max}$ (disturbance quantity) and the duration $u_k$ of the Pockels cell being in state with high reflection coefficient $R_{\max}$, can be obtained. Again, output energy $h(N_k, u_k, P_k)$ and build-up

$$b_k = \int_{k/f_{\mathrm{sw}}}^{(k+1)/f_{\mathrm{sw}}} \sum_{j=1}^{M} P_j(t)\mathrm{d}t = \tilde{h}_{\mathrm{fs}}(N_k, u_k, P_k) \tag{3.50}$$

can be defined to re-write the pulse-to-pulse dynamics

$$b_{k+1} = \tilde{h}_{\mathrm{fs}}(f_{\mathrm{fs}}(\tilde{h}_{\mathrm{fs}}^{-1}(b_k, u_k, P_k), u_k, P_k), u_{k+1}, P_{k+1}) = \tilde{f}_{\mathrm{fs}}(b_k, u_k, u_{k+1}, P_k, P_{k+1}) \tag{3.51}$$

using output measurements only.

Analogously, the average deterministic model is denoted as

$$N_{k+1} = f_{\mathrm{avg}}(N_k, u_k), \tag{3.52a}$$

$$b_{k+1} = \tilde{h}_{\mathrm{avg}}(f_{\mathrm{avg}}(\tilde{h}_{\mathrm{avg}}^{-1}(b_k, u_k), u_k), u_{k+1}) = \tilde{f}_{\mathrm{avg}}(b_k, u_k, u_{k+1}), \tag{3.52b}$$

omitting the disturbance quantities $P_k$.

Figure 3.1 shows the pulse-to-pulse dynamics of the average model for the given set-point and steady state control input where the fixed point $N_s = f_{\mathrm{avg}}(N_s, u_s)$ can be found by the intersection of the dashed curve $N$ with the continuous curve $f_{\mathrm{avg}}(N, u_s)$. For the region of interest around the steady state, the build-up function allows for a one-to-one mapping between $N$ and $b$ which is not possible for the energy output. Thus, the build-up is favoured once again over the output energy. This property of $\tilde{h}_{\mathrm{avg}}$ does not hold for arbitrarily large values of $N$, though. For large values, this curve decreases again due to premature pulse build-ups during the lossy prelasing phase. However, this can be neglected in practice as one is primarily interested in the region shown in Figure 3.1.

For control, the augmented state space model

$$\mathbf{z}_{k+1} = \mathbf{f}(\mathbf{z}_k, \nu_k) = \begin{bmatrix} \tilde{f}_{\mathrm{avg}}(b_k, u_k, \nu_k) \\ \nu_k \end{bmatrix} \tag{3.53}$$

is defined analogously to Chapter 2 with augmented state $\mathbf{z} = \begin{bmatrix} b & u \end{bmatrix}^{\mathrm{T}}$.

## 3.2 Reinforcement Learning

In this section, the performance of Q-learning and SARSA will be discussed. For computation reasons, these algorithms will interact with the average model because the order of the system is then reduced to 2 and does not have to simulate individual lasing modes.

Compared to the model of the regenerative amplifier in Chapter 2, the same approach for the quantization levels is used as described in Section 2.3.2. The reward function also remains the same as in Section 2.3.1. For details about the parameters, see Appendix A.2.

Another challenge of this plant compared to the regenerative amplifier is that it is easier to destabilize the plant, i. e. the plant is being pushed away from the steady state in fewer

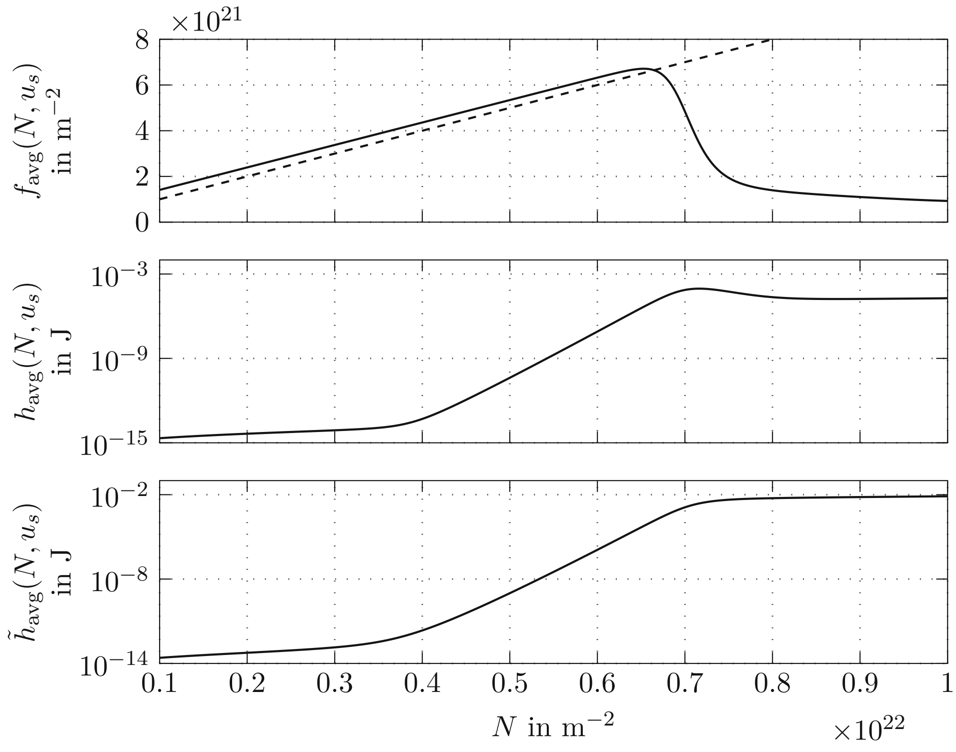

Figure 3.1: Pulse-to-pulse dynamics of the average model for the pulsed laser system with cavity dumping for steady state control input.

discrete steps/control actions compared to the RA. One explanation for this might be the larger spectral radius of the transition matrix of the dynamical system linearized around the steady state. In total, this makes learning difficult as one exploratory action might bring the system far away from the steady state, requiring a long time to re-stabilize the system around the steady state.

### 3.2.1 Rewards and control law

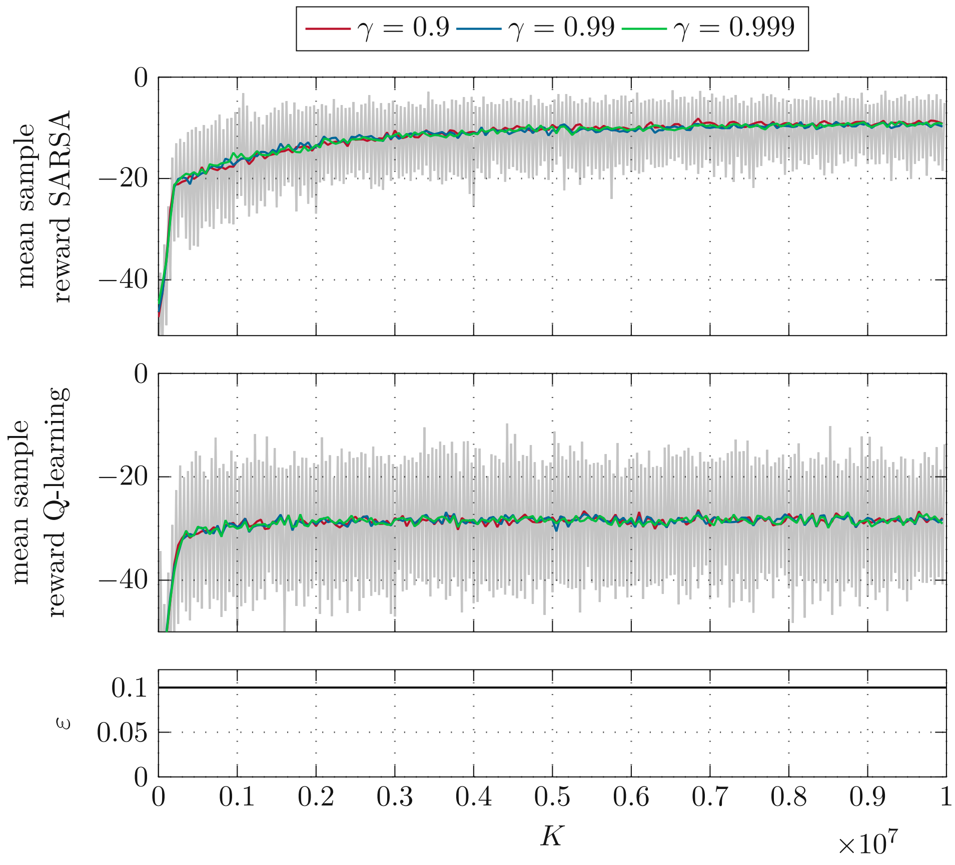

Figure 3.2: Episode rewards for SARSA and Q-learning simulation with constant exploration, comparing discount factors.

Figure 3.2 shows the learning process for SARSA and Q-learning for a constant exploration rate $\varepsilon = 0.1$, learning rate $\alpha_K = 1/N_K(\check{\mathbf{z}}_k, \check{\nu}_k)$ and varying discount factor $\gamma$. Learning takes much longer to reach satisfying results with $10^7$ samples compared to about $5 \cdot 10^5$ required for the RA. Again, the discount factor does not significantly influence learning speed or performance and is thus set to $\gamma = 0.99$ if not noted otherwise. Looking at the mean rewards only, SARSA performs better than Q-learning as the mean rewards converge to a value of $-9$ while Q-learning performs worse than that with a mean reward of about $-27$ after $10^7$ training samples. The variance of the reward gained during training is higher for Q-learning compared to SARSA. This agrees with the notion that off-policy methods typically have higher variance than their on-policy counterparts [22, p. 125].

However, when investigating the transient response of the closed loop in Figure 3.3, SARSA is clearly not able to properly stabilize the plant. Compared to the regenerative amplifier, SARSA does not occasionally produce spikes in the control input, but rather does not find any suitable policy at all. Q-learning and the LQR, on the other hand, stabilize the plant.

The problem with SARSA lies again in the conversion of the stochastic policy into a deterministic one. SARSA outperforms Q-learning in the deterministic environment, stochastic policy case over a large range of exploration rates $\varepsilon$, as shown in Figure 3.4. Only for really small exploration rates $\varepsilon < 0.01$, Q-learning is able to show its merits and dramatically reduces the variances of build-up, control input and rewards. This is due to the fact that Q-learning takes around 10 to 20 samples on average (cf. Figure 3.3) to stabilize the plant from a random initial state caused by e. g. one random action. Thus, control policies with high exploration rate do not allow Q-learning to restore the equilibrium point before being pushed away again by the next random action, leading to high costs and variances. SARSA takes future random actions into account and sacrifices smooth control laws for the ability to bring the system near the steady state quickly, without ever properly stabilizing the plant there, achieving an overall higher reward in the stochastic case. As shown in Figure 3.2, SARSA is the first control approach to let the build-up rise to values near the equilibrium point at $k = 8$. The other two methods lag behind in this regard, with build-up levels rising significantly at $k = 10$.

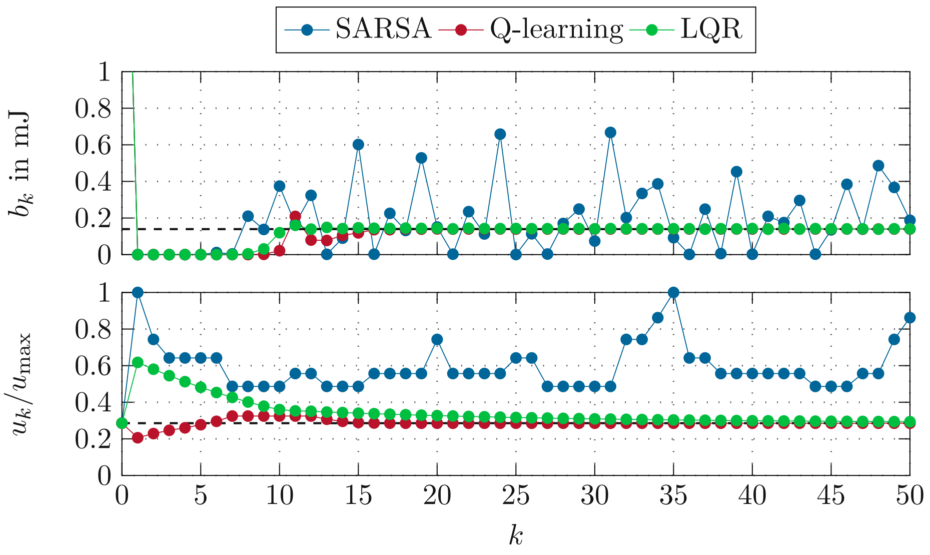

Figure 3.3: Transient closed-loop response of a cavity-dumped laser model with SARSA, Q-learning and LQR controller, $N_0 = 1.1N_s$, $u_0 = u_s$.

Figure 3.5 shows two slices of the control law for constant build-up $\nu(b_s, u)$ and past control input $\nu(b, u_s)$, respectively. While near the steady state, Q-learning shows a monotonic increase in the control input for both axes, aligning with the LQR control

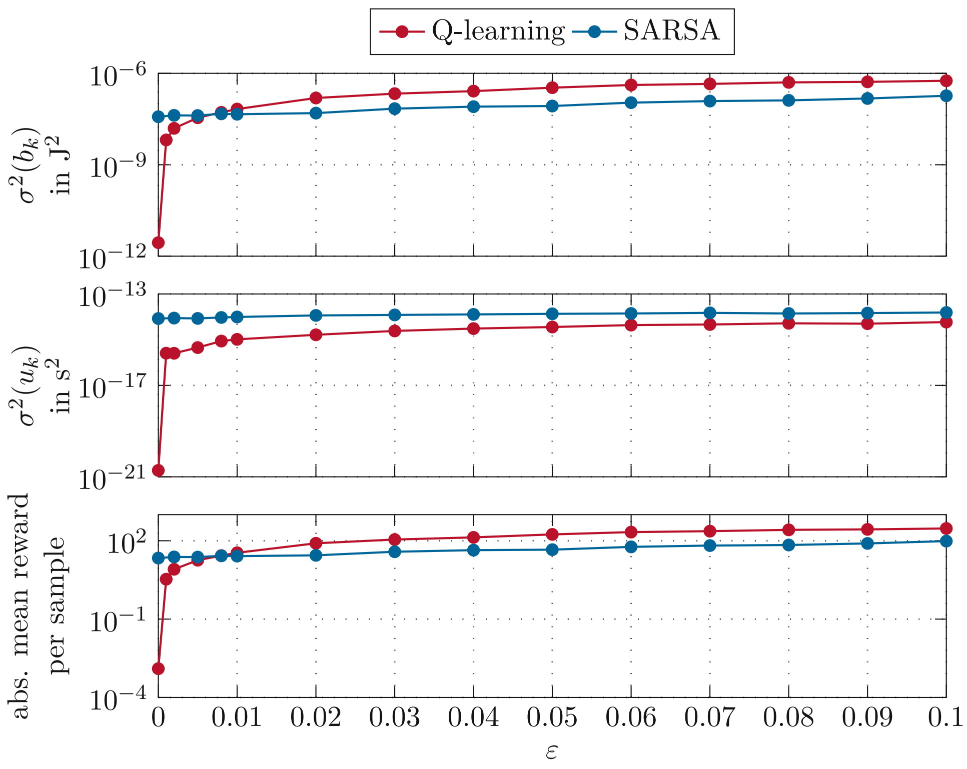

Figure 3.4: Variance of build-up, control input and absolute mean rewards over the course of 10,000 samples, starting in $(b_s,\, u_s)$.

law, SARSA's control law increases and decreases seemingly arbitrarily. This results in SARSA's inability to stabilize the plant.

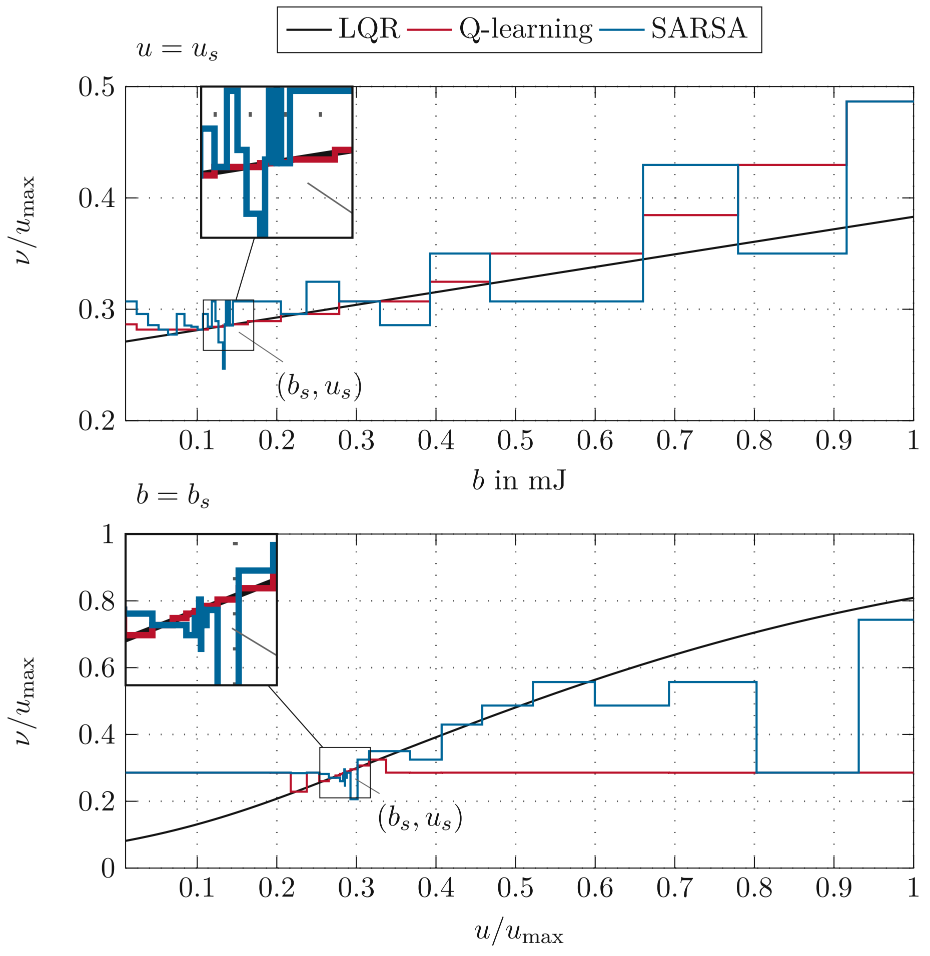

Figure 3.5: Control law for Q-learning, SARSA and LQR controller.

## 3.2.2 Hyperparameter analysis

Next, the influences of the exploration rate $\varepsilon$ and the step size $\alpha_K$ will be investigated. As SARSA does not stabilize the plant, only Q-learning will be considered.

In Figure 3.6, the mean rewards over the learning process, averaged over the local 500,000 samples, are drawn for different strategies to select $\varepsilon$. In contrast to the RA, where all algorithms reached approximately the same rewards, independent of $\varepsilon$, this is not the case for this laser system. After an initial improvement until $K \approx 3 \cdot 10^5$, the rewards follow the exploration pattern, i. e. a decrease in exploration leads to an increase in rewards. The reward curves for $\varepsilon = 0.05$ and the linear reduction intersect at $K = 5 \cdot 10^6$, the same value for $K$ as the exploration curves intersect. The same holds true for the intersection between $\varepsilon = 0.05$ and the exponential reduction as well the intersection between exponential and linear reduction, indicating that the exploration rate strongly determines the transient behaviour during learning, much more compared to the RA. Although not visible in the rewards, Q-learning's deterministic policy improves in terms of closed-loop stability as measured by variances. Figure 3.7 shows the variance for the build-up and control input as well as the mean rewards averaged over 10,000 samples for the deterministic control law of Q-learning obtained by $\nu(\mathbf{z}) = \arg\max_\nu Q(\mathbf{z}, \nu)$. Learning was done with constant $\varepsilon = 0.1$ and terminated after either $K = 10^6$ or $K = 10^7$ samples. The rewards in Figure 3.6 do not increase for the stochastic policy with constant exploration in the interval $t \in [10^6, 10^7]$ and yet the improvement of the closed loop for the deterministic policy is significant as measured by the variances in Figure 3.7.

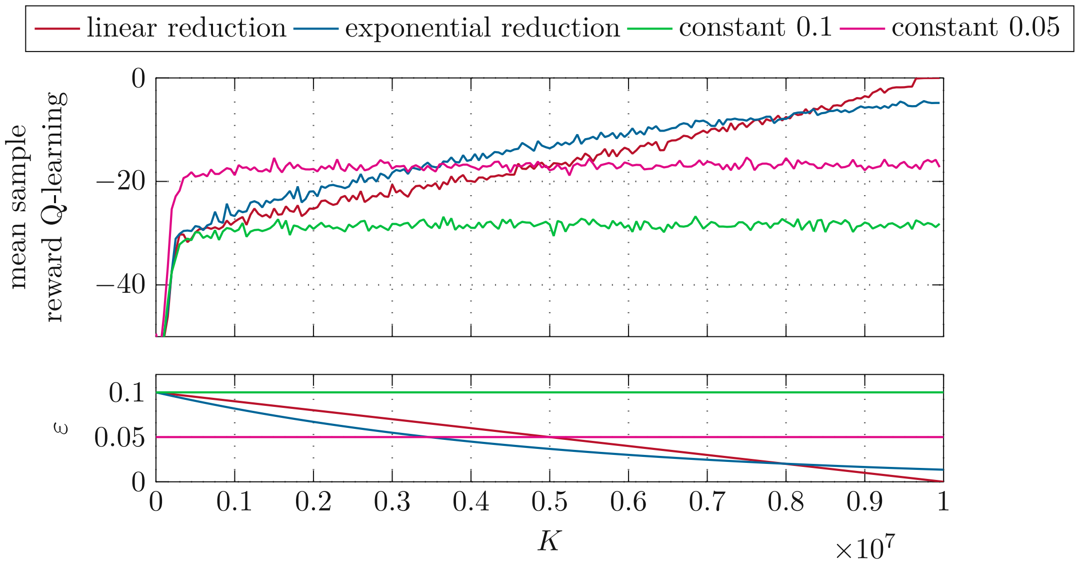

Figure 3.6: Mean episode rewards of Q-learning over the learning process for varying exploration rate $\varepsilon$.

Although changing the methods for selecting $\varepsilon$ influences the state-action space coverage as depicted in Figure 3.8, with constant $\varepsilon = 0.1$ yielding the largest state-action space coverage, the methods figure out identical control laws around the steady state. Hence, all methods achieve the same values for build-up and control input variances as well as rewards if the deterministic control law is applied to the plant over 10,000 samples after learning is completed. This is illustrated in Figure 3.9.

Thus, while the choice of $\varepsilon$ crucially affects the transient behaviour during learning and the state-action space coverage, it does not influence the final result of the deterministic policy. Nevertheless, if learning is necessary in the presence of parameter drifts, a non-zero exploration rate is mandatory to be able to adapt to the changing environment.

Next, the effects of the step size $\alpha_K$ will be discussed. Figure 3.11 shows the mean rewards over the learning process and Figure 3.11 depicts the state-action space coverage. Both quantities are largely independent of the choice of $\alpha_K$, regardless if the nominal case $1/N_K(\check{\mathbf{z}}_k, \check{\nu}_k)$, linear reduction or constant values for $\alpha_K$ are selected. When the controller is applied in the closed loop with a deterministic policy after learning, variances

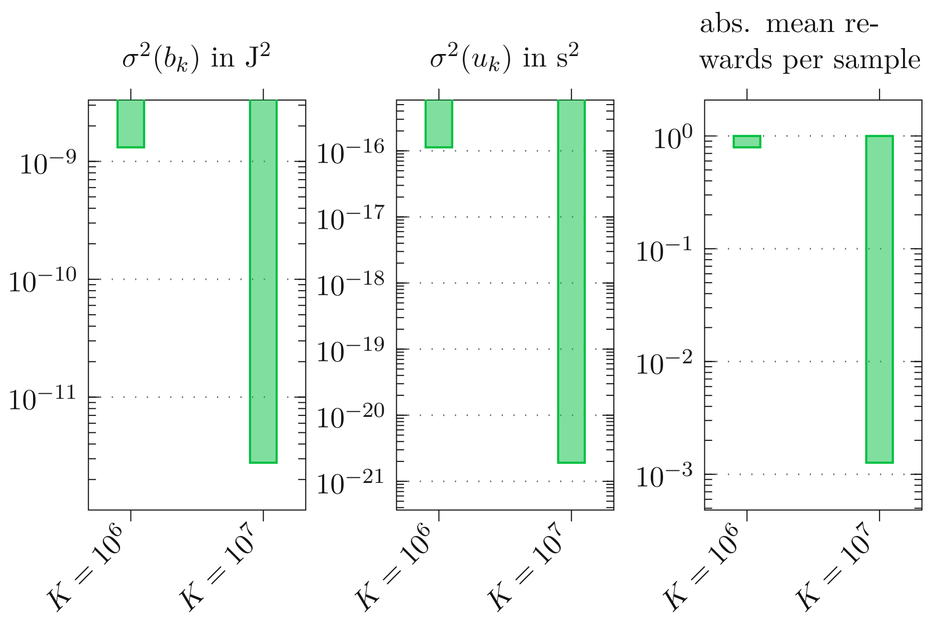

Figure 3.7: Variances of build-up, control input and absolute reward mean of Q-learning's deterministic policy, after learning with $\varepsilon = 0.1$ is terminated after either $K = 10^6$ or $K = 10^7$ samples.

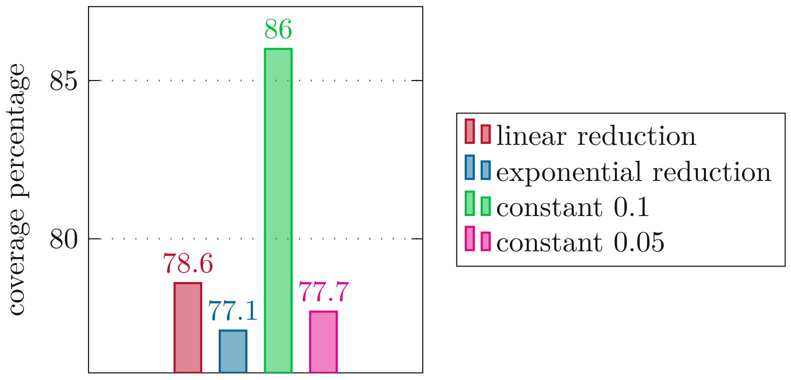

Figure 3.8: State-action space coverage of Q-learning for varying $\varepsilon$.

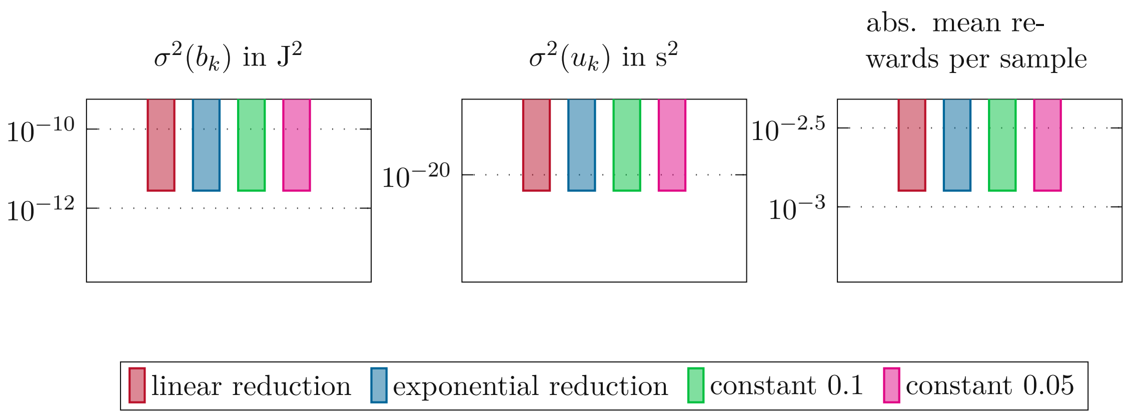

Figure 3.9: Variances of build-up and control input and mean absolute rewards for varying exploration rates $\varepsilon$ after learning.

of build-up and control input as well as rewards yield identical results to Figure 3.9, so they are not shown again.

With these results in mind, Q-learning with constant $\alpha_K = 0.05$ and $\varepsilon = 0.05$ is proposed to be able to continuously update the control law in the presence of parameter drifts.

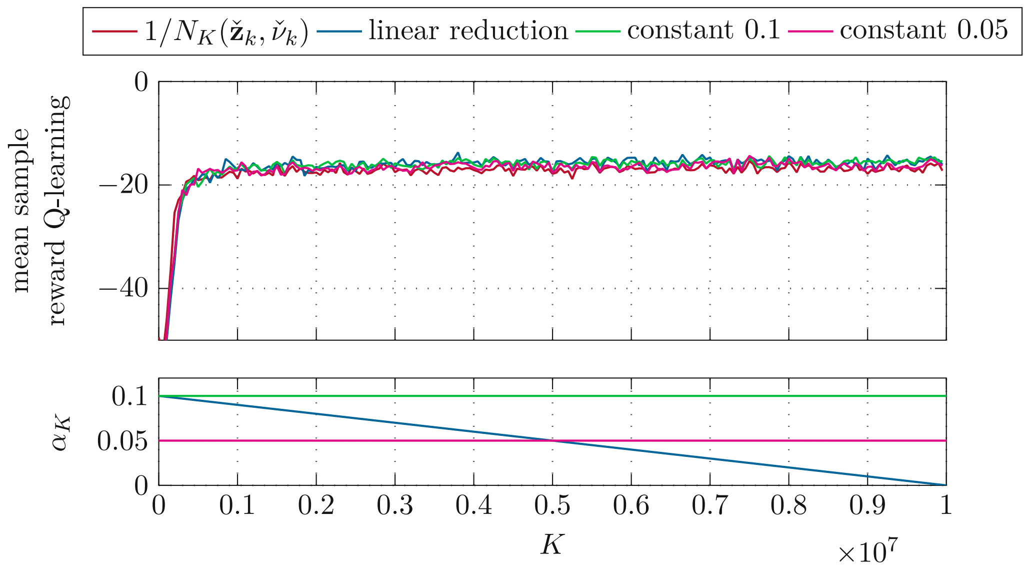

Figure 3.10: Mean rewards of Q-learning over the learning process for varying step size $\alpha_K$ and $\varepsilon = 0.05$.

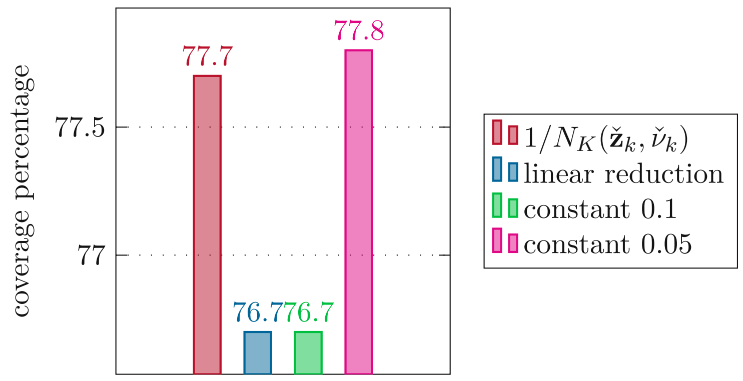

Figure 3.11: State-action space coverage of Q-learning for varying step size $\alpha_K$ after learning.

### 3.2.3 QVI

The model-based RL controller based on Q-value iteration is also capable of stabilizing the laser system. Figure 3.12 illustrates the control law for two slices of the whole state space, i. e. $\nu(b_s, u)$ and $\nu(b, u_s)$. For the axis $b = b_s$, the control law aligns perfectly with the LQR with tanh smoothing function. The other axis $u = u_s$ develops more interestingly by showing two spikes over $b$.

These two spikes stem from a different number of steps it takes to stabilize the plant. Figure 3.13 illustrates the transient response of the closed loop for two initial values for the build-up. The first $b_0 = 1.5$ mJ corresponds to the first peak in Figure 3.12, while the second $b_0 = 3$ mJ is associated with the second peak. For $b_0 = 1.5$ mJ, the control law drives the build-up close to the steady state within one step as desired by the reward function which favors a steady state value of the build-up over a control input steady state as selected by the chosen weights for $\mathbf{Q}$ and $R$. As control input levels are coarse away from the steady state, the build-up fluctuates during the first 25 samples before converging to the steady state. In the mean time, the controller gradually reduces the control input to steadily drive the population inversion $N_k$ (which is not directly visible to the controller) to the desired equilibrium point.

If $b_0$ is not contained in the first spike in Figure 3.12, then QVI cannot bring the build-up to the steady state within one step, i. e. no near dead-beat behaviour is possible. Thus, the control input zigzags once before the build-up is brought close to steady state. Again, the first samples dither a little due to quantized control levels.

### 3.2.4 Stochastic Plant

Until now, only the nominal average deterministic model has been considered. Q-learning and QVI are also able to stabilize the full stochastic model. This is illustrated in Figure 3.14. Q-learning restrains its actions to a limited range around the steady state, while QVI utilizes a larger range of control inputs. While this enables a fast increase of the build-up after $k = 9$ samples to values near the steady state, it takes QVI longer to arrive at the steady state with the control input $u_k$ compared to Q-learning. Occasionally, after

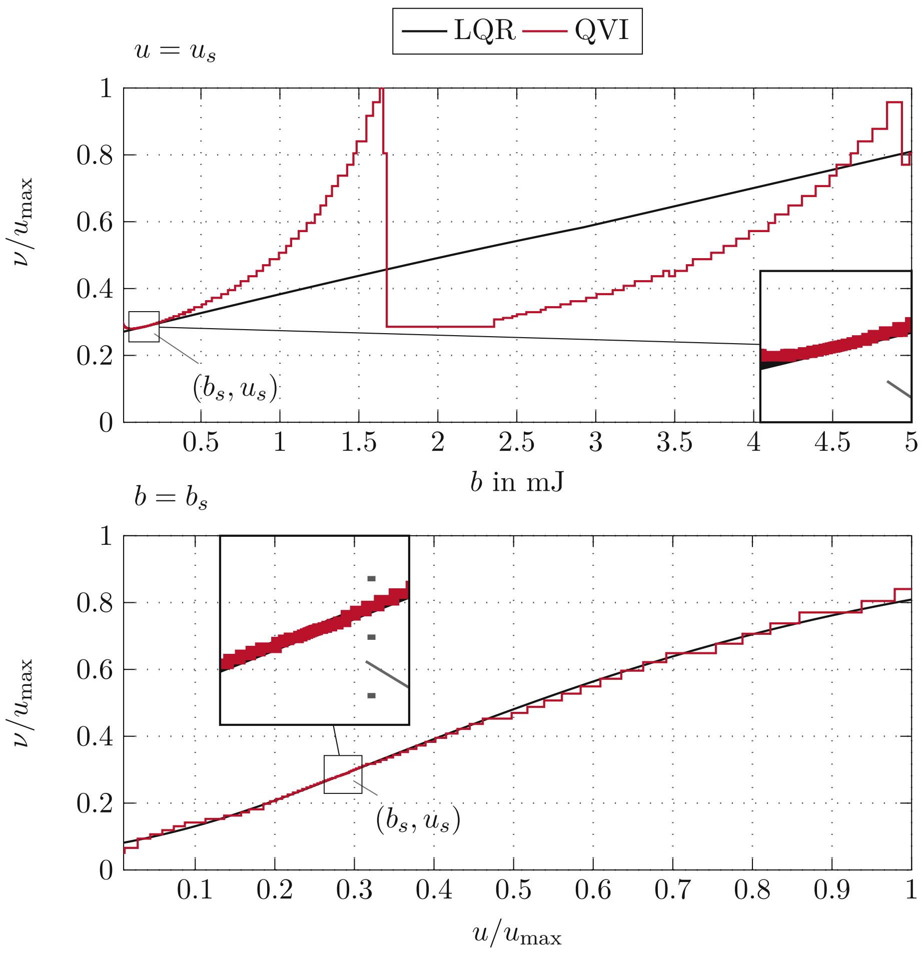

Figure 3.12: Control law of QVI for $b = b_s$ and $u = u_s$.

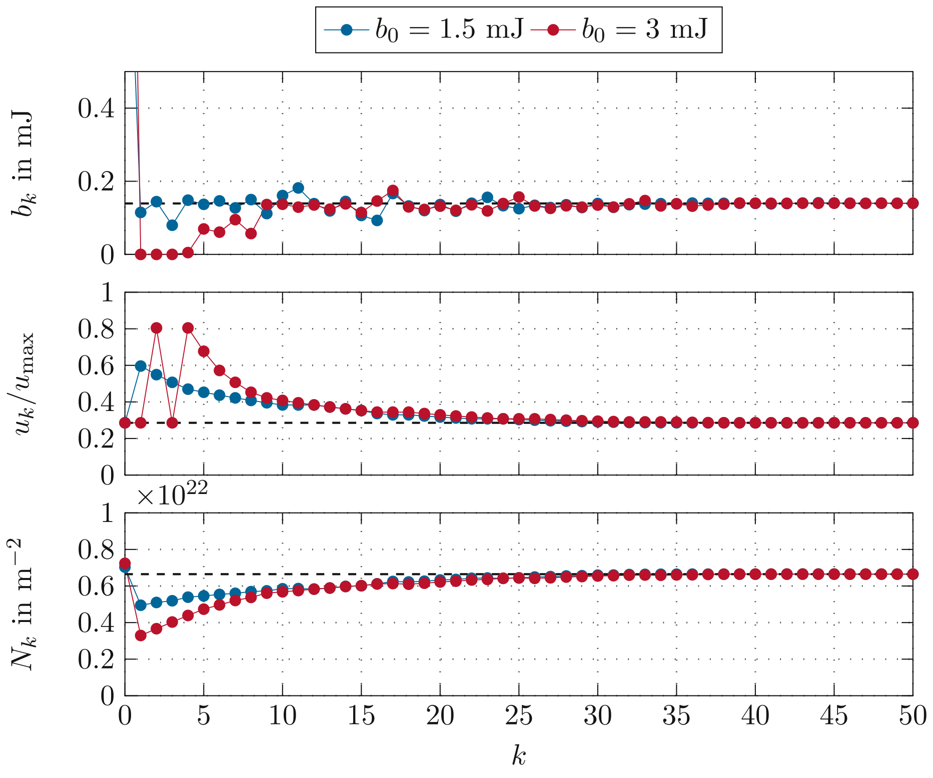

Figure 3.13: Transient closed-loop response with QVI controller with $b_0 = 3$ mJ or $b_0 = 1.5$ mJ and $u_0 = u_s$.

reaching the steady state, the values for the build-up fluctuate due to the noise terms caused by spontaneous emissions, but both controllers are able to stabilize the system. The open loop with $u_k = u_s$ is also shown to demonstrate the instability of the operating point.

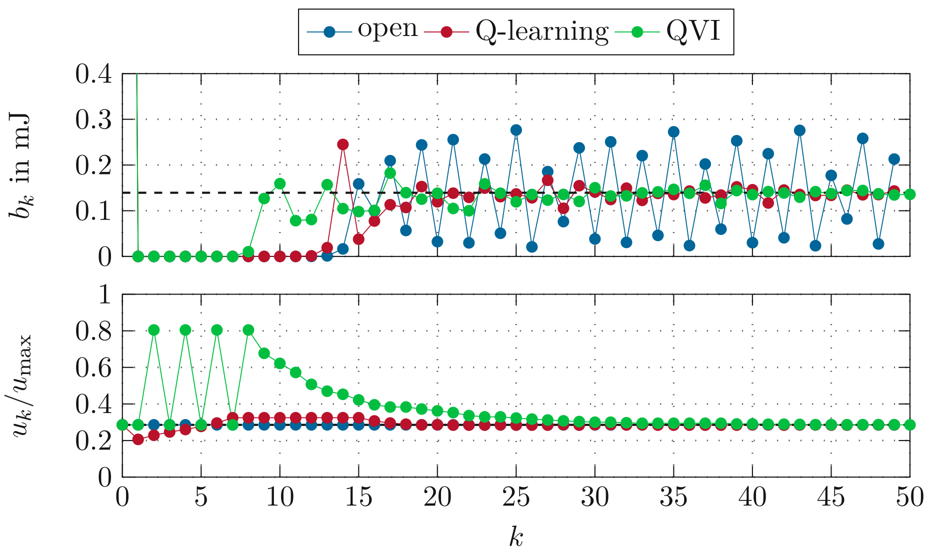

Figure 3.14: Transient response of full stochastic model: closed loop for Q-learning and QVI controller and open loop, $N_0 = 1.1N_s$, $u_0 = u_s$.

## 3.3 Non-linear dead-beat controller

This section investigates the control law based on stabilizing manifolds from Section 2.4, applied to the laser system with cavity dumping. These results will again be used to compare them to the performance of the RL algorithms.

### 3.3.1 Neural network implementation

As shown in Figure 3.1, the build-up spans multiple orders of magnitude. Thus, it is more challenging to find a neural network that is capable of approximating the dynamics function $\tilde{f}(\cdot)$ accurately. In comparison with the NN in Chapter 2, the size of the hidden layers of the neural network needs to be increased to 64 and 32 neurons, respectively. Hence, each hidden layer features double the amount of neurons. Nevertheless, the overall structure with the input normalization layer, tanh activation functions and output layer remains unchanged.

### 3.3.2 Control law

Figure 3.15 depicts the control law for the non-linear dead-beat controller. For comparison, the LQR and the results from QVI are also plotted. For $u = u_s$, both QVI and the dead-beat controller show a steeper increase of the control input compared to the linear controller. This enables them to stabilize the equilibrium point more quickly compared to the LQR which is illustrated in Figure 3.16. Although the LQR manages a quick increase in the build-up at the beginning, the non-linear controller reaches the steady state after $k = 6$ samples. The LQR, on the other hand, reduces the difference $u_k - u_s$ very slowly after the initial four transitions even if the build-up is already very close to the steady state.

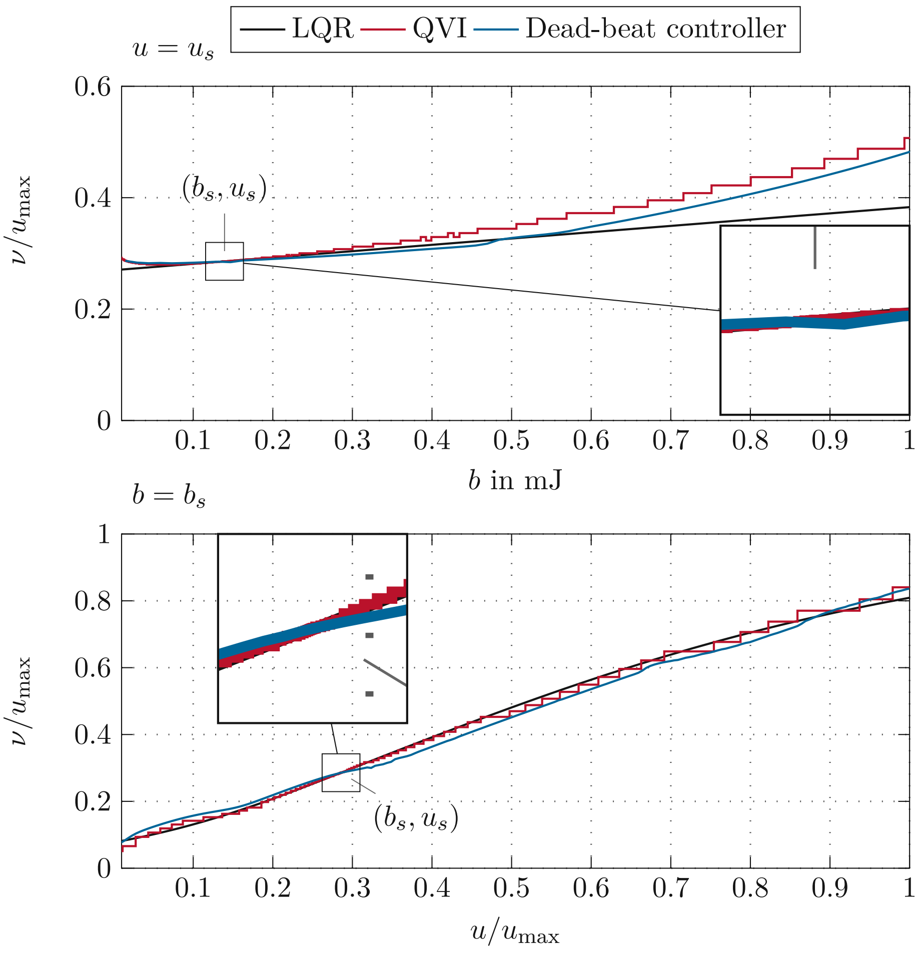

Figure 3.15: Control law for non-linear dead-beat controller, QVI and LQR.

## 3.4 Robustness analysis

Until now, only the nominal plant was considered. This section covers the performance of the control strategies in the presence of parameter uncertainties. For comparison reasons, only the absorption loss coefficient $\alpha_{\mathrm{P}}$ is varied where $\alpha_{\mathrm{P,nom}}$ denotes the absorption loss coefficient of the nominal model.

Figure 3.17 shows the transient response of the closed loop with the Q-learning controller

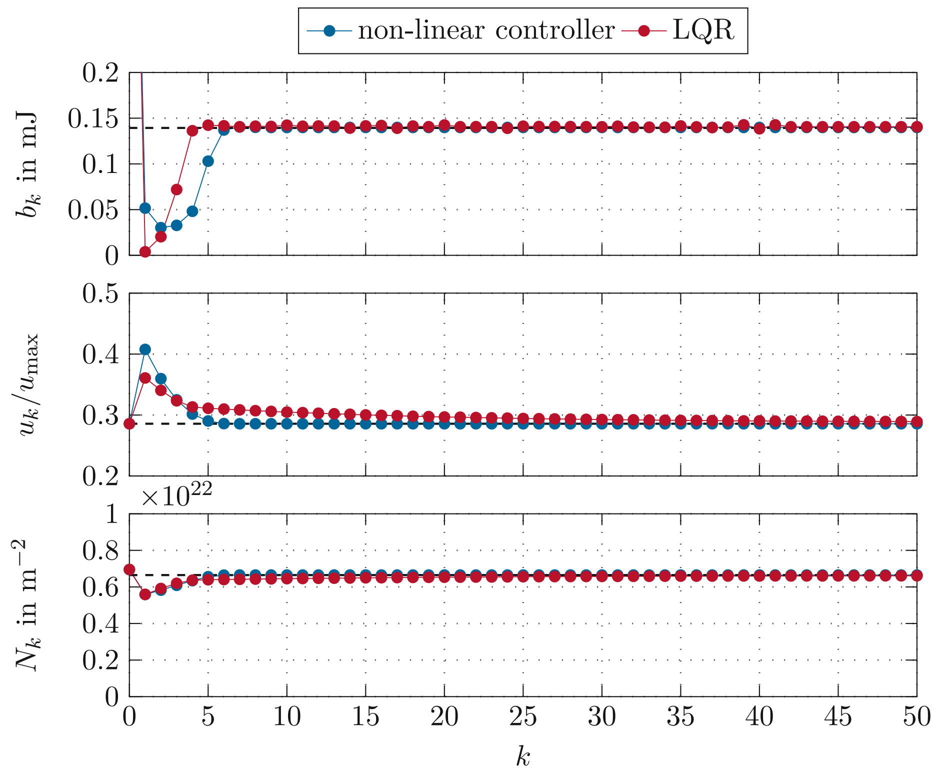

Figure 3.16: Transient closed-loop response for non-linear dead-beat and LQR controller, $b_0 = 0.8$ mJ, $u_0 = u_s$.

obtained by the nominal plant applied to the average model with model-plant mismatch $\alpha_\mathrm{P} = 1.1\alpha_{\mathrm{P,nom}}$. This controller yields a limit cycle with period 3. In contrast, a controller which has been updated over $2.5 \cdot 10^6$ samples with step size $\alpha_K = 0.05$ and exploration rate $\varepsilon = 0.05$ is able to avoid this undesirable phenomenon. This demonstrates that Q-learning again adapts to changing parameters, although the learning process is very slow.

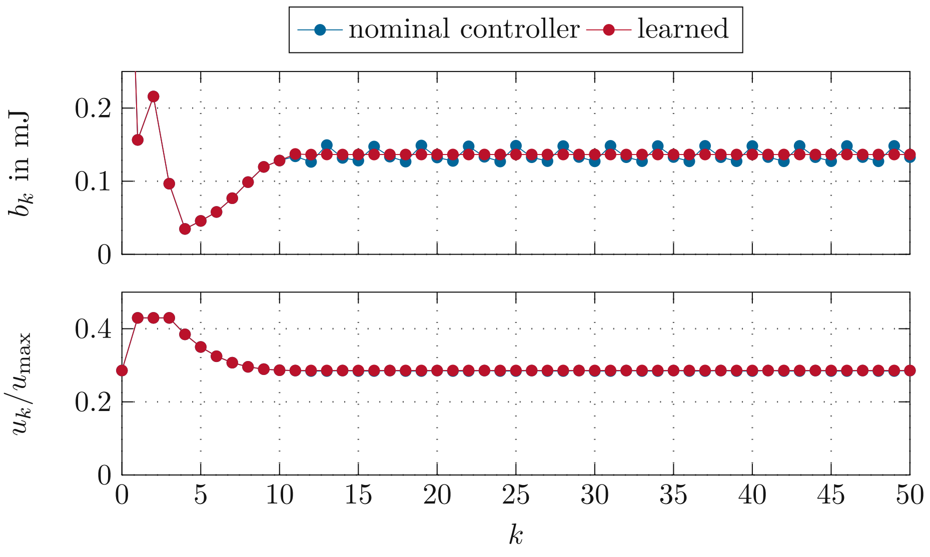

Figure 3.17: Transient closed-loop response for Q-learning before and after learning for model plant mismatch $\alpha_\mathrm{P} = 1.1\alpha_{\mathrm{P,nom}}$, $b_0 = 0.8$ mJ, $u_0 = u_s$, .

On the other hand, Figure 3.18 illustrates the closed-loop response for the non-linear dead-beat controller with the same model-plant mismatch as in Figure 3.17. While the adaptive controller based on the LMS estimate of the pumping power is immediately able to cope with this parameter uncertainty, the NN controller takes more samples to stabilize the equilibrium point. However, it also reaches the steady state before the first update occurs at $k = 50$. Thus, updates are not required for this mismatch and do not affect the closed loop in a noticeable way. Even for larger model-plant mismatches, no step-like behaviour as shown in Section 2.5 with the RA emerges.

Finally, Figure 3.19 shows the parameters $\alpha_\mathrm{P}$ for which the respective algorithms are able to stabilize the steady state. For $\alpha_\mathrm{P} < 0.5\alpha_{\mathrm{P,nom}}$ and $\alpha_\mathrm{P} > 1.5\alpha_{\mathrm{P,nom}}$, the set-point is stable. Both non-linear control variants based on the dead-beat law successfully manage to stabilize the whole range for $\alpha_\mathrm{P}$. QVI succeeds to stabilize the plant in the range $[0.75, 1.18]\alpha_{\mathrm{P,nom}}$. While the nominal Q-learning controller performs poorly due to coarse control input quantization, the adaptive Q-learning controller manages to improve its control law and stabilizes the plant for a similar range of $\alpha_\mathrm{P}$ compared to QVI, shifted to slightly larger values of $\alpha_\mathrm{P}$.

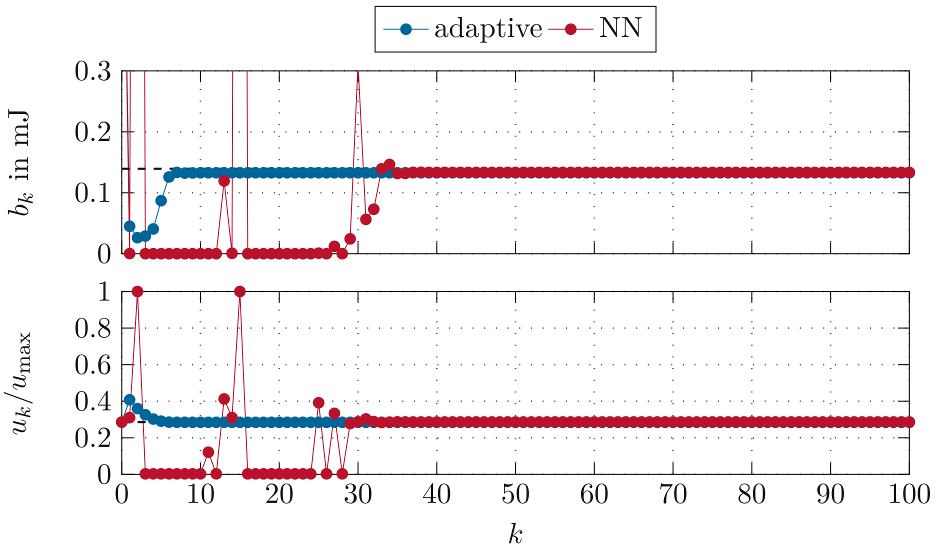

Figure 3.18: Transient closed-loop response for non-linear dead-beat adaptive and NN controller, $b_0 = 0.8$ mJ, $u_0 = u_s$, model plant mismatch $\alpha_P = 1.1\alpha_{P,\text{nom}}$ and parameters $\overline{\mu} = 1.03$, $N_f = 50$, $N_{MB} = 20$, $\alpha_{MB} = 10^{-2}$.

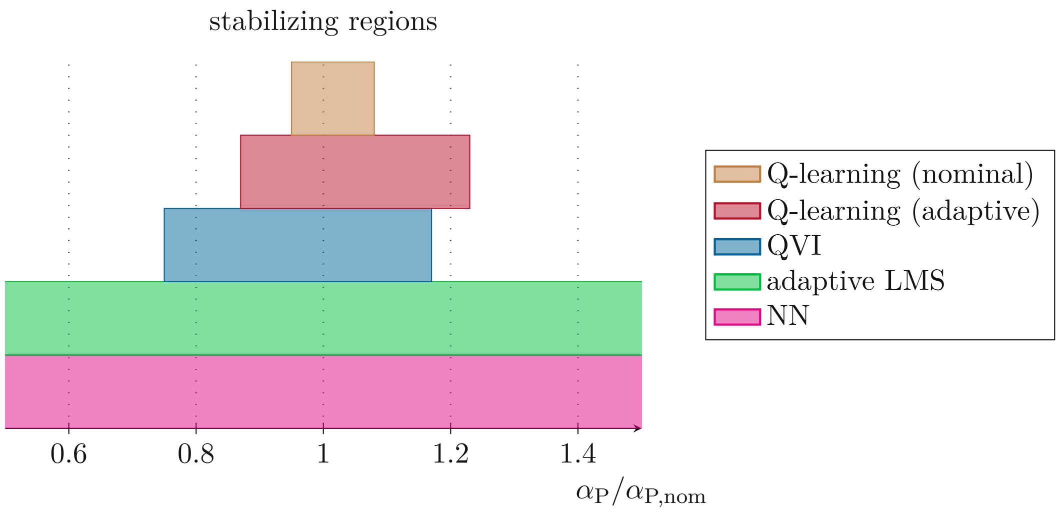

Figure 3.19: Stabilizing regions for the algorithms for varying $\alpha_P$.

### 3.4.1 Other operating points

In this section, only one operating point has been discussed with a switching frequency $f_{\mathrm{sw}} = 500$ kHz. While SARSA already struggles to properly stabilize this system, Q-learning and QVI still manage to perform well. This stands in contrast to the RA, where all three methods were able to fulfil the task, with SARSA only producing small spikes every now and then.

Simulations show that for lower switching frequencies $f_{\mathrm{sw}} < 300$ kHz, Q-learning and QVI are also no longer able to stabilize the system around the steady state due to a larger spectral radius of the transition matrix of the linearized system for these operating points. Thus, a small deflection from the equilibrium point, caused by e. g. quantized control actions, can quickly lead to larger deviations from the set-point, making tabular based RL algorithms unfit for this task. A solution to fix this problem might be to increase the state-action space size which would exacerbate the issue of long learning times. Thus, other control approaches are required for these operating points.

## 3.5 Summary

In this chapter, the control of a pulsed laser system based on cavity dumping was investigated using RL methods. SARSA does not stabilize the plant whereas Q-learning and Q-value iteration both are able to. For Q-learning, the choice of the exploration rate crucially influences the transient behaviour during learning, but is less relevant for the final result after converting Q-learning to a deterministic policy. QVI produces a very similar control law to the non-linear dead-beat controller and reduces the build-up very quickly to the steady state value. When it comes to parameter uncertainties, the adaptive non-linear controller based dead-beat control laws perform well and deal with a wide range of unknown parameters. The RL agents lack behind in this regard, with much more limited parameter ranges where they can stabilize the desired set-point. Moreover, refining the policy to these unknown model parameters takes very long compared to the adaptive controllers. When it comes to lower switching frequencies, tabular-based RL methods are no longer capable of achieving a stable closed loop, making the non-linear dead-beat controller a viable alternative there. However, regarding the high complexity of the system dynamics and the substantial degree of model knowledge leveraged by the non-linear dead-bead control concept, the RL-based controllers still exhibit convincing performance considering their applicability to a wide range of system classes and various types of parameter and model-plant mismatches.

# 4 Conclusion

In this thesis, reinforcement learning techniques which can be implemented in real-time were investigated to stabilize the unstable pulse-to-pulse dynamics of pulsed laser systems.

In Chapter 2, a reward function inspired by the linear-quadratic regulator problem is proposed. Slight modifications in the index shift of the build-up make it appropriate to be used in the RL setting. Tabular-based RL is then used to stabilize the pulse amplification of a regenerative amplifier. Thus, the state space is partitioned into tiles, with a higher density around the desired set-point. This enables Q-value iteration as a model-based RL framework to find an optimal control law using dynamical programming. The model-free RL algorithms investigated are comprised of Monte Carlo Control, SARSA and Q-learning. While Monte Carlo Control is very inefficient when it comes to learning speed due to its under-utilization of the inherent dynamical connections between the state-space tiles, SARSA and Q-learning need 20 times fewer samples to exceed the performance of Monte Carlo Control. However, when comparing Q-learning and SARSA, Q-learning shows superior performance in the closed loop after being converted to a deterministic policy. This is caused by the fact the SARSA as an on-policy algorithm is optimized for a stochastic policy whereas Q-learning as an off-policy algorithm is also able to fit to a deterministic policy. Simulations show that the discount factor $\gamma$ has a negligible effect on the result of the policy. While a high exploration rate $\varepsilon$ increases the learning speed, i. e. how fast the reward is improving during learning, it causes a more noisy closed loop as measured by the variances of the build-up and control input. The choice of the step size $\alpha_K$ influences the evolution of the mean reward over the learning process significantly, while it hardly affects the closed-loop performance when the policies are converted to deterministic ones after training. In general, off-policy methods outperform their on-policy counterparts in every case, indicating that these types of RL algorithms are more suited for the control task given.

Because of the choice of the reward weights, both QVI and Q-learning show near dead-beat behaviour if the limited control input permits. This is confirmed by comparing the control law of these algorithms with a non-linear dead-beat controller. In addition to a LMS estimator adaptive controller, a neural network implementation is proposed to cope with parameter uncertainties. These two non-linear controllers are able to adapt to unknown model parameters very well, while the RL agents are not capable of stabilizing the plant in such a wide range of varying parameters. However, while the non-linear controllers presented here exploit model information extensively, RL methods are able to stabilize the set-point without any prior system information.

Chapter 3 investigates a pulsed laser system based on cavity dumping. While SARSA achieves higher mean rewards compared to Q-learning, it completely fails to stabilize the system in a deterministic setting, whereas Q-learning successfully fulfils that task. This highlights again the fact that off-policy methods seem to be more suited than on-policy for stabilizing a set-point. For this system, the choice of $\varepsilon$ dominates the learning speed and the mean reward achieved during learning. Other parameters, such as the step size and the discount factor only have a minor influence. Nevertheless, the deterministic policy of Q-learning obtained after learning does not depend on $\varepsilon$. This is due to the fact that this system can be de-stabilized much more easily than the regenerative amplifier. Thus, Q-learning is punished for suboptimal actions much more compared to the regenerative amplifier, reducing the influence of the exploration rate on the final result. QVI demonstrates dead-beat behaviour already seen for the regenerative amplifier which is confirmed by comparing the control law with the non-linear dead-beat approach. For the set-point investigated, the non-linear controllers are able to stabilize the plant in the presence of large model parameter uncertainties. RL methods are not able to stabilize the set-point for these large model parameter uncertainties.

Further research must be conducted to find a better way to discretize the state-space. The challenge is to find a balance between point density which is required to accurately determine a suitable control input and the size of the state space which greatly influences learning speed. However, if the set-point is roughly known, RL methods manage to find policies which can stabilize the laser system within a few pulses, achieving this result with methods that can be implemented in real-time if computation time is tightly limited.

# A Model details

In this chapter, details about the model, including the ordinary differential equations, timing, and parameter values are given for the regenerative amplifier and the cavity dumping laser system.

## A.1 Regenerative Amplifier

This section restates the model of the regenerative amplifier presented in [18]. The single-pass function

$$\begin{bmatrix} G_{k,n+1} \\ H_{k,n+1} \end{bmatrix} = \mathbf{f}_{\mathrm{SP}}\left(\begin{bmatrix} G_{k,n} \\ H_{k,n} \end{bmatrix}\right) = \begin{bmatrix} f_{\mathrm{dep}}(G_{k,n}, H_{k,n}) \\ \eta_{\mathrm{RC}} f_{\mathrm{gain}}(G_{k,n}, H_{k,n}) \end{bmatrix} \tag{A.54}$$

includes the evolution of the gain $G$ and fluences $H$ over each pass $n$ of the pulse $k$ through the medium with the functions

$$f_{\mathrm{dep}}(G, H) = \frac{G}{G - (G-1)\exp(-H)} \tag{A.55a}$$

and

$$f_{\mathrm{gain}}(G, H) = \log(G\exp(H) - G + 1) \tag{A.55b}$$

where $\log(\cdot)$ denotes the natural logarithm. The function $G_1 = f_{\mathrm{pump}}(G_0, p)$ describes the restoration of the gain in between two consecutive pulses and is determined by the solution of the initial value problem

$$\frac{\mathrm{d}}{\mathrm{d}t}G(t) = -\gamma_{21} f_2[\sigma f_1(f_1 + f_2)N_{\mathrm{dop}}L + \log(G(t))]G(t) \tag{A.56}$$

$$+\sigma\frac{p}{A_B}\frac{(f_1+f_2)^2}{\hbar\omega_P}\left[1 - \exp(-\sigma_P\kappa_1 N_{\mathrm{dop}}L)G(t)^{\frac{\kappa_2\sigma_P}{(f_1+f_2)\sigma}}\right]G(t) \tag{A.57}$$

$$G(0) = G_0, \tag{A.58}$$

where $G_1 = G(1/f_{\mathrm{rep}})$, $\kappa_1 = \frac{f_0 f_2 - f_1 f_3}{f_1 + f_2}$ and $\kappa_2 = \frac{f_0 + f_3}{f_1 + f_2}$.

The parameters for the model and the weighting factors of the Reinforcment Learning algorithms are shown in Table A.1.

## A.2 Cavity Dumping

This section covers timing parameters as well as other parameters of the average model used for simulation.

| Symbol | Value | Unit |
|---|---|---|
| $\lambda$ | $1030 \cdot 10^{-9}$ | m |
| $\lambda_P$ | $979 \cdot 10^{-9}$ | m |
| $\gamma_{21}$ | 416.66 | $s^{-1}$ |
| $A_B$ | $2.0106 \cdot 10^{-6}$ | $m^2$ |
| $\sigma = \sigma_{\text{nom}}$ | $8 \cdot 10^{-25}$ | $m^2$ |
| $\sigma_P$ | $9 \cdot 10^{-25}$ | $m^2$ |
| $f_0$ | 0.9736 | 1 |
| $f_1$ | 0.0264 | 1 |
| $f_2$ | 0.9217 | 1 |
| $f_3$ | 0.0783 | 1 |
| $N_{\text{dop}}$ | $3.8 \cdot 10^{26}$ | $m^{-3}$ |
| $L$ | 0.006 | m |
| $\eta_{RC}$ | 0.8 | 1 |
| $\eta_{\text{out}}$ | 0.9 | 1 |
| $f_{\text{rep}}$ | 1 | kHz |
| $u_{\max}$ | $7.8241 \cdot 10^{-8}$ | 1 |
| $u_{\min}$ | $3.912 \cdot 10^{-9}$ | 1 |
| $p$ | 297 | W |
| $N_{\text{RT}}$ | 25 | 1 |
| $u_s$ | $1.56482 \cdot 10^{-8}$ | 1 |
| $b_s$ | 0.44435 | 1 |
| $y_s$ | 0.1648 | 1 |
| $x_s$ | 1.5784 | 1 |
| $Q_{00}$ | 5.0726 | 1 |
| $Q_{10} = Q_{01}$ | 0 | 1 |
| $Q_{11}$ | $1.6335 \cdot 10^{11}$ | 1 |
| $R$ | $1.6335 \cdot 10^{12}$ | 1 |
| $b_{\min}$ | $2.919 \cdot 10^{-7}$ | 1 |
| $b_{\max}$ | 2.959 | 1 |
| $N_b$ (model-free) | 50 | 1 |
| $N_u$ (model-free) | 30 | 1 |
| $N_b$ (QVI) | 1000 | 1 |
| $N_u$ (QVI) | 100 | 1 |

Table A.1: Parameters for the regenerative amplifier model.

The evolution of the population inversion and the intracavity power averaged over all modes are given as [21]

$$\frac{\mathrm{d}N(t)}{\mathrm{d}t} = -\, b_{\text{eff}}\gamma_{\text{eff}}N + \frac{b_{\text{eff}}\lambda_P}{hcA_p}(1 - \exp\{\sigma_P[(2B_{45}+1)N(t) - LN_{\text{dop}}] - \alpha_{\text{P}}L\} - \alpha_{\text{P}}L)p \tag{A.59a}$$

$$- \frac{b_{\text{eff}}P(t)}{hcA_s}\left(\sum_{k=0}^{\infty} N(t)^k q_k - \overline{\lambda}\right), \tag{A.59b}$$

$$\frac{\mathrm{d}P(t)}{\mathrm{d}t} = \frac{2q_1}{q_0 t_{\text{RT}}}N(t)P(t) - \frac{P(t)}{\tau} + \frac{\log(R(t))P(t)}{t_{\text{RT}}} + \frac{2}{t_{\text{RT}}}\frac{\Delta\Omega}{4\pi}\overline{\sigma}N(t). \tag{A.59c}$$

The fitting parameters $q_k$ are split into two phases. During the prelase phase $t_{\text{output}} \leq t \leq \frac{1}{f_{\text{sw}}} - t_{\text{sw}} - u_s$ a different set of values is used than for the remainder of the cycle. The evolution of the reflection coefficient $R(t)$ is illustrated in Figure A.1 for the steady state case, with the control input $u_s$ being the time duration at which $R(t) = R_{\max}$.

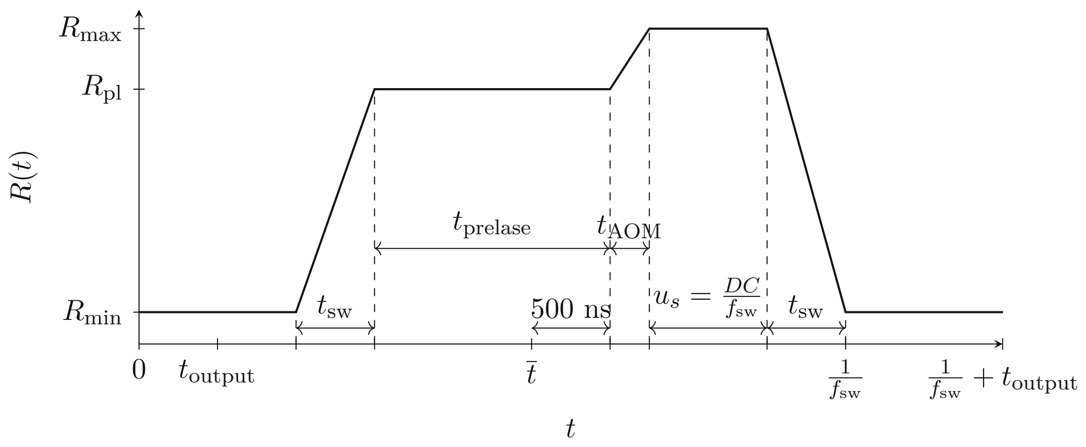

Figure A.1: Timing of reflection coefficient for one pulse cycle.

The associated parameters are shown in Tables A.2 and A.3.

| Symbol | Value | Unit |
|---|---|---|
| $\lambda_P$ | 806 | nm |
| $A_P$ | $3.5257 \cdot 10^{-7}$ | $\mathrm{m}^2$ |
| $A_s$ | $4.646 \cdot 10^{-7}$ | $\mathrm{m}^2$ |
| $b_{\mathrm{eff}}$ | 0.9837 | 1 |
| $\gamma_{\mathrm{eff}}$ | 8893 | $\mathrm{s}^{-1}$ |
| $\sigma_P$ | $7.7 \cdot 10^{-24}$ | $\mathrm{m}^2$ |
| $B_{45}$ | 0.01657 | 1 |
| $L$ | 0.012 | m |
| $\alpha_{\mathrm{P}} = \alpha_{\mathrm{P,nom}}$ | 12.5 | $\mathrm{m}^{-1}$ |
| $N_{\mathrm{dop}}$ | $6.9 \cdot 10^{25}$ | $\mathrm{m}^{-3}$ |
| $p$ | 22.5 | W |
| $\overline{\lambda}$ | $1.032 \cdot 10^{-6}$ | m |
| $t_{\mathrm{RT}}$ | 1.4357 | ns |
| $\tau$ | 7.2047 | ns |
| $\Delta\Omega$ | $2\pi$ | 1 |
| $\overline{\sigma}$ | $3.4226 \cdot 10^{-30}$ | $\mathrm{m}^2$ |
| prelase phase | | |
| $q_0$ | $1.0326 \cdot 10^{-6}$ | m |
| $q_1$ | $2.4516 \cdot 10^{-29}$ | $\mathrm{m}^3$ |
| $q_2$ | $3.0549 \cdot 10^{-52}$ | $\mathrm{m}^5$ |
| $q_3$ | $2.6099 \cdot 10^{-75}$ | $\mathrm{m}^7$ |
| $q_4$ | $1.7014 \cdot 10^{-98}$ | $\mathrm{m}^9$ |
| $q_5$ | $8.7949 \cdot 10^{-122}$ | $\mathrm{m}^{11}$ |
| $q_6$ | $3.9769 \cdot 10^{-145}$ | $\mathrm{m}^{13}$ |
| $q_7$ | $1.5195 \cdot 10^{-168}$ | $\mathrm{m}^{15}$ |
| $q_8$ | $5.1036 \cdot 10^{-192}$ | $\mathrm{m}^{17}$ |
| $q_9$ | $1.5293 \cdot 10^{-215}$ | $\mathrm{m}^{19}$ |
| $q_{10}$ | $4.1370 \cdot 10^{-239}$ | $\mathrm{m}^{21}$ |
| $q_{11}$ | $1.0200 \cdot 10^{-262}$ | $\mathrm{m}^{23}$ |
| $q_{12}$ | $2.3103 \cdot 10^{-286}$ | $\mathrm{m}^{25}$ |
| $q_{13}$ | $4.8392 \cdot 10^{-310}$ | $\mathrm{m}^{27}$ |

Table A.2: Parameters for the cavity dumping model.

| Symbol | Value | Unit |
|---|---|---|
| remaining phase | | |
| $q_0$ | $1.0326 \cdot 10^{-6}$ | m |
| $q_1$ | $2.8059 \cdot 10^{-29}$ | $m^3$ |
| $q_2$ | $3.8325 \cdot 10^{-52}$ | $m^5$ |
| $q_3$ | $3.5007 \cdot 10^{-75}$ | $m^7$ |
| $q_4$ | $2.4033 \cdot 10^{-98}$ | $m^9$ |
| $q_5$ | $1.3218 \cdot 10^{-121}$ | $m^{11}$ |
| $q_6$ | $6.0661 \cdot 10^{-145}$ | $m^{13}$ |
| $q_7$ | $2.3883 \cdot 10^{-168}$ | $m^{15}$ |
| $q_8$ | $8.2345 \cdot 10^{-192}$ | $m^{17}$ |
| $q_9$ | $2.5254 \cdot 10^{-215}$ | $m^{19}$ |
| $q_{10}$ | $6.9746 \cdot 10^{-239}$ | $m^{21}$ |
| $q_{11}$ | $1.7521 \cdot 10^{-262}$ | $m^{23}$ |
| $q_{12}$ | $4.0365 \cdot 10^{-286}$ | $m^{25}$ |
| $q_{13}$ | $8.5680 \cdot 10^{-310}$ | $m^{27}$ |
| $f_{\mathrm{sw}}$ | 500 | kHz |
| $t_{\mathrm{output}}$ | 19.8 | ns |
| $DC$ | 0.1 | 1 |
| $R_{\mathrm{min}}$ | 0.05 | 1 |
| $R_{\mathrm{pl}}$ | 0.8 | 1 |
| $R_{\mathrm{max}}$ | 0.995 | 1 |
| $t_{\mathrm{AOM}}$ | 0 | s |
| $t_{\mathrm{sw}}$ | 6.6 | ns |
| $t_{\mathrm{prelase}}$ | 1.6 | µs |
| $\bar{t}$ | 1.2934 | µs |
| $u_{\mathrm{max}}$ | 0.7 | µs |
| $u_{\mathrm{min}}$ | 0.02 | µs |
| $u_s$ | 0.2 | µs |
| $b_s$ | 0.1394 | mJ |
| $y_s$ | 11.134 | µJ |
| $N_s$ | $6.6476 \cdot 10^{21}$ | $m^{-2}$ |
| $b_{\mathrm{min}}$ | $9.103 \cdot 10^{-11}$ | J |
| $b_{\mathrm{max}}$ | $2.919 \cdot 10^{-3}$ | J |
| $N_b$ (model-free) | 50 | 1 |
| $N_u$ (model-free) | 30 | 1 |
| $N_b$ (QVI) | 800 | 1 |
| $N_u$ (QVI) | 100 | 1 |
| $Q_{00}$ | $5.1461 \cdot 10^{8}$ | $J^{-2}$ |
| $Q_{10} = Q_{01}$ | 0 | $J^{-1}s^{-1}$ |
| $Q_{11}$ | 0 | $s^{-2}$ |
| $R$ | $2.5 \cdot 10^{11}$ | $s^{-2}$ |

Table A.3: Parameters for the cavity dumping model (continued).

# Bibliography

[1] L. Li, "The challenges ahead for laser macro, micro and nano manufacturing," in *Advances in Laser Materials Processing*, 2nd ed., Woodhead Publishing, 2018, pp. 23–42.

[2] S. Pratheesh Kumar, S. Elangovan, R. Mohanraj, and V. Sathya Narayanan, "Significance of continuous wave and pulsed wave laser in direct metal deposition," in *Materials Today: Proceedings*, (Mar. 19–21, 2021), 3rd International Conference on Materials, Manufacturing and Modelling, vol. 46, Vellore, India, pp. 8086–8096.

[3] S. Royo and M. Ballesta-Garcia, "An overview of lidar imaging systems for autonomous vehicles," *Applied Sciences*, vol. 9, no. 19: 4093, 2019.

[4] M. Maiuri, M. Garavelli, and G. Cerullo, "Ultrafast spectroscopy: State of the art and open challenges," *Journal of the American Chemical Society*, vol. 142, no. 1, pp. 3–15, 2020.

[5] J. McLaughlin, L. K. Branski, W. B. Norbury, S. E. Bache, L. Chilton, N. El-Muttardi, and B. Philp, "Laser for burn scar treatment," in *Total Burn Care*, 5th ed., Elsevier, 2018, pp. 648–654.

[6] W. Koechner, *Solid-State Laser Engineering*, 6th ed. 2006.

[7] O. Svelto, *Principle of Lasers*, 5th ed. Springer, New York, 2010.

[8] W. Silfvast, *Laser Fundamentals*, 2nd ed. Cambridge University Press, 2004.

[9] A. E. Siegman, *Lasers*, 1st ed. University Science Books, 1986.

[10] J. Dörring, A. Killi, U. Morgner, A. Lang, M. Lederer, and D. Kopf, "Period doubling and deterministic chaos in continuously pumped regenerative amplifiers," *Optics Express*, vol. 12, no. 8, pp. 1759–1768, 2004.

[11] M. Grishin, V. Gulbinas, and A. Michailovas, "Dynamics of high repetition rate regenerative amplifiers," *Optics Express*, vol. 15, no. 15, pp. 9434–9443, 2007.

[12] M. Grishin, V. Gulbinas, and A. Michailovas, "Bifurcation suppression for stability improvement in nd:yvo4 regenerative amplifier," *Optics Express*, vol. 17, no. 18, pp. 15 700–15 708, 2009.

[13] P. Kroetz, A. Ruehl, G. Chatterjee, A.-L. Calendron, K. Murari, H. Cankaya, P. Li, F. X. Kärtner, I. Hartl, and R. J. D. Miller, "Overcoming bifurcation instability in high-repetition-rate ho:ylf regenerative amplifiers," *Optics Letters*, vol. 40, no. 23, pp. 5427–5430, 2015.

[14] Y. O. Barmenkov, A. V. Kiryanov, and M. V. Andres, "Experimental study of the nonlinear dynamics of an actively q-switched ytterbium-doped fiber laser," *IEEE Journal of Quantum Electronics*, vol. 48, no. 11, pp. 1484–1493, 2012.

[15] M. Kovalsky and A. Hnilo, “Chaos in the pulse spacing of passive q-switched all-solid-state lasers,” *Optics Letters*, vol. 35, no. 20, pp. 3498–3500, 2010.

[16] Y. Wang and C.-Q. Xu, “Switching-induced perturbation and influence on actively q-switched fiber lasers,” *IEEE Journal of Quantum Electronics*, vol. 40, no. 11, pp. 1583–1596, 2004.

[17] C. Fries, M. Weitz, C. Theobald, P. v. Löwis of Menar, J. Bartschke, and J. A. L’huillier, “Cavity-dumped yb:yag ceramic in the 20 w, 12 mj range at 6.7 ns operating from 20 hz to 5 khz with fluorescence feedback control,” *Applied Optics*, vol. 55, no. 24, pp. 6538–6546, 2016.

[18] A. Deutschmann-Olek, T. Flöry, K. Schrom, V. Stummer, A. Baltuška, and A. Kugi, “Bifurcation suppression in regenerative amplifiers by active feedback methods,” *Optics Express*, vol. 28, no. 2, pp. 1722–1737, 2020.

[19] A. Deutschmann-Olek, W. Kemmetmüller, and A. Kugi, “On the global feedback stabilization of regenerative optical amplifiers,” in *Proc. 21st IFAC World Congress 2020*, (Jul. 11–17, 2020), 21st IFAC World Congress 2020, pp. 5447–5452.

[20] L. Tarra, A. Deutschmann-Olek, V. Stummer, T. Flöry, A. Baltuska, and A. Kugi, “Stochastic nonlinear model of the dynamics of actively q-switched lasers,” *Optics Express*, vol. 30, no. 18, pp. 32 411–32 427, 2022.

[21] L. Tarra, A. Deutschmann-Olek, and A. Kugi, “Nonlinear feedback stabilisation and stochastic disturbance suppression of actively q-switched lasers,” *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 77–82, 2023, 22nd IFAC World Congress.

[22] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2nd ed. MIT press, 2018.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” 2013. arXiv: `1312.5602`.

[24] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[25] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” in *Proceedings of the 31st International Conference on International Conference on Machine Learning*, vol. 32, Beijing, China, 2014, pp. 387–395.

[26] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” 2016. arXiv: `1509.02971`.

[27] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” 2018. arXiv: `1801.01290`.

[28] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *Proceedings of the 32nd International Conference on Machine Learning*, (Jul. 7–9, 2015), vol. 37, Lille, France, 2015, pp. 1889–1897.

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. arXiv: `1707.06347`.

[30] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, eaau5872, 2019.

[31] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, (May 29–Jun. 3, 2017), Singapore, Singapore: IEEE Press, 2017, pp. 3389–3396.

[32] G. Masinelli, T. Le-Quang, S. Zanoli, K. Wasmer, and S. A. Shevchik, "Adaptive laser welding control: A reinforcement learning approach," *IEEE Access*, vol. 8, pp. 103 803–103 814, 2020.

[33] C. Sun, E. Kaiser, S. L. Brunton, and J. N. Kutz, "Deep reinforcement learning for optical systems: A case study of mode-locked lasers," *Machine Learning: Science and Technology*, vol. 1, no. 4, p. 045 013, 2020.

[34] Q. Yan, Q. Deng, J. Zhang, Y. Zhu, K. Yin, T. Li, D. Wu, and T. Jiang, "Low-latency deep-reinforcement learning algorithm for ultrafast fiber lasers," *Photonics Research*, vol. 9, no. 8, pp. 1493–1501, 2021.

[35] H. Zhao, Q. Lou, J. Zhou, F. Zhang, J. Dong, Y. Wei, G. Wu, Z. Yuan, Z. Fang, and Z. Wang, "High-repetition-rate mhz acoustooptic $q$-switched fiber laser," *IEEE Photonics Technology Letters*, vol. 20, no. 12, pp. 1009–1011, 2008.

[36] S. Raković and W. S. Levine, *Handbook of Model Predictive Control*, 1st ed. Birkhäuser, 2019.

[37] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st. USA: John Wiley & Sons, Inc., 1994.

[38] R. Postoyan, L. Buşoniu, D. Nešić, and J. Daafouz, "Stability analysis of discrete-time infinite-horizon optimal control with discounted cost," *IEEE Transactions on Automatic Control*, vol. 62, no. 6, pp. 2736–2749, 2017.

[39] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[40] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[41] S. Singh, R. Sutton, and P. Kaelbling, "Reinforcement learning with replacing eligibility traces," *Machine Learning*, vol. 22, pp. 123–158, 1995.

[42] C. Wang, S. Yuan, K. Shao, and K. Ross, "On the convergence of the monte carlo exploring starts algorithm for reinforcement learning," 2022. arXiv: `2002.03585`.

[43] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[44] D. Precup, R. S. Sutton, and S. P. Singh, "Eligibility traces for off-policy policy evaluation," in *Proceedings of the Seventeenth International Conference on Machine Learning*, San Francisco, CA, USA, 2000, pp. 759–766.

[45] G. Rummery and M. Niranjan, “On-line q-learning using connectionist systems,” *Technical Report CUED/F-INFENG/TR 166*, 1994.

[46] S. Singh, T. Jaakkola, M. Littman, and C. Szepesvári, “Convergence results for single-step on-policy reinforcement-learning algorithms,” *Machine Learning*, vol. 38, pp. 287–308, 2000.

[47] H. van Seijen, H. van Hasselt, S. Whiteson, and M. Wiering, “A theoretical and empirical analysis of expected sarsa,” in *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, (Mar. 30–Apr. 2, 2009), Nashville, TN, USA, pp. 177–184.

[48] T. W. Cusick and P. Stănică, *Cryptographic Boolean Functions and Applications*. Academic Press, 2009.

[49] T. Jaakkola, M. I. Jordan, and S. P. Singh, “On the convergence of stochastic iterative dynamic programming algorithms,” *Neural Computation*, vol. 6, no. 6, pp. 1185–1201, 1994.

[50] H. Hasselt, “Double q-learning,” in *Advances in Neural Information Processing Systems, 24th Annual Conference on Neural Information Processing Systems 2010*, (Dec. 6–9, 2010), vol. 23, Vancouver, British Columbia, Canada, pp. 2613–2621.

# Eidesstattliche Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß dem Code of Conduct – Regeln zur Sicherung guter wissenschaftlicher Praxis (in der aktuellen Fassung des jeweiligen Mitteilungsblattes der TU Wien), insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In– noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Wien, August 2024

Bernhard Schimkowitsch