Reinforcement Learning Framework for the self-learning Suppression of Clutch Judder in automotive Drive Trains

Reinforcement Learning Framework zur selbstlernenden Unterdrückung von Kupplungsrupfen in PKW-

Zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften
der Fakultät für Maschinenbau
Karlsruher Institut für Technologie (KIT)

genehmigte
Dissertation

von

Dipl.-Ing. Hermann Sommer Obando
Aus Mexiko Stadt / Mexiko

Tag der mündlichen Prüfung: 26.04.2016
Hauptreferent: Prof. Dr.-Ing. Dr. h.c. A. Albers
Korreferent: Prof. Peter Meckl

Kurzfassung

Mechanische Schwingungen in Antriebssträngen wirken sich negativ auf den Fahrkomfort aus und führen daher zu Akzeptanzproblemen beim Kunden. Ein Beispiel sind reibungsinduzierte Schwingungen während der Schlupfphase der Kupplung, die als „Rupfen“, „Shudder“ oder „Chatter“ bezeichnet werden. Es existieren zahlreiche Maßnahmen gegen Rupfen, denen konstruktive und tribologische Ansätze zugrunde liegen. Im Fall elektromechanisch aktuierter Kupplungen bietet sich darüber hinaus die Möglichkeit der aktiven Dämpfung von Reibschwingungen durch eine geeignete Regelung der Anpresskraft. Deren Umsetzung ermöglicht die Anwendung von Hochleistungswerkstoffen in der Reibpaarung, die als Folge eine kosten- und energieeffizientere Dimensionierung der Kupplung erlauben. Sie erfordert jedoch meistens den Einsatz komplexer Regler, für deren Design und Tuning möglichst genaue Modelle der Regelstrecke vorliegen müssen. Diese Regler sind oft auf die spezifische Regelstrecke angepasst und daher nicht ohne weitere Maßnahmen für den Einsatz in anderen Systemen oder bei Auftreten von beim Reglerdesign nicht berücksichtigter Änderungen geeignet. In dieser Arbeit wird eine mechatronische Maßnahme zur effektiven Unterdrückung von Rupfschwingungen vorgeschlagen, die sich modellunabhängig und selbständig auf die Regelstrecke und deren Änderungen anpasst. Sie besteht aus einem Reinforcement Learning Framework zur Regelung der Anpresskraft mit dem Ziel der aktiven Schwingungsdämpfung. In diesem Framework erlernt ein Algorithmus die optimale Kraftregelung indem er die Interaktion mit der Umgebung auswertet und sein Verhalten entsprechend anpasst. Die Entwicklung des Frameworks erfordert eine ganzheitliche Betrachtung der Systeme „Fahrer“, „Fahrzeug“ und „Umgebung“ mit allen Subsystemen und Komponenten sowie deren Wechselwirkungen untereinander. Aus diesem Grund wird in dieser Arbeit seine Entwicklung durch eine Kombination aus Simulation und Versuchen an physischen Prüfständen durch das IPEK „X-in-the-Loop“ Framework unterstützt. Zunächst wird ein Fahrmanöver als Testfall definiert und als Reinforcement Learning-Problem formuliert, bevor es auf einer abstrakten Ebene formuliert wird und alle wichtigen Parameter und Elemente identifiziert und definiert werden. Als nächstes wird das Framework in einer vereinfachten Simulations-umgebung implementiert, um mögliche Lösungskonzepte zu untersuchen. Anschließend wird das Framework mit den gewonnenen Erkenntnissen in einer simulierten Prüfstandsumgebung eingesetzt. Die viel-versprechendste Konfiguration wird anschließend auf einem physischen Prüfstand implementiert. Durch das erarbeitete Reinforcement Learning Framework konnten die in Simulation und Versuch reproduzierten Rupfschwingungen signifikant gedämpft werden. Die Aussichten auf eine erfolgreiche Implementierung an Prototypen oder Serienfahrzeugen sind daher äußerst vielversprechend.

Abstract

The presence of mechanical vibrations in automotive drive trains has a negative effect on the durability of its components and the perceived comfort of the vehicle by the driver. An example of these vibrations is clutch judder, also known as chatter or shudder, which refers to those mechanical vibrations that originate at the clutch during its slipping phase. Numerous countermeasures against clutch judder exist and can usually be categorized as either constructive or tribological. In electromechanically actuated clutches, the active damping of vibrations by means of control of the clamping force allow the use of high performance materials in the friction pairing, which makes a more energy and cost efficient design of the clutch. However, most solutions have a high complexity, since they involve detailed and accurate models of the drive train and its environment as well as relatively advanced and abstract mathematical operations for the design and tuning of the required controllers. This also leads to the solutions often being only suitable for the specific system for which they were developed. For this reason, a mechatronic countermeasure that relies as little as possible on the accuracy and detail of models and that is able to adapt to changes of these without compromising its effectiveness is proposed in this work. It consists of a reinforcement learning framework for the control of the clamping force for the active suppression of judder vibrations. In the proposed framework, an optimal control of the clamping force is learned by evaluating its interaction with its environment and adapting its behavior accordingly. For the further development of the solution a holistic consideration of the “vehicle”, “driver” and “environment” systems and all their interactions is required. For this reason, a combination of simulation and physical experiments in the context of the IPEK “X-in-the-Loop” framework is proposed in this work. First, the test maneuver is analyzed and the judder reduction task is formulated as a reinforcement learning control problem. After identifying the relevant parameters and elements, the framework is formulated on an abstract level and subsequently implemented on a simplified simulation model, where different solution concepts are analyzed. Afterwards, the results are used for the implementation of the framework on a simulated test bench environment. The most promising concept is later implemented on a physical test bench environment. The results, both on simulation and on the physical test bench, show a significant reduction of judder vibrations through the implementation of the proposed solution. Therefore, the reinforcement learning framework delivers very promising results in regards to an implementation on physical prototypes and series vehicles.

Acknowledgement

There have been more people who have in some way or another provided me with their help and support than I could possibly mention. However, I would like to thank those that I can remember and apologize for the omission of those whose name is not found in this acknowledgement.

First and foremost, I want to thank my doctoral advisor, Prof. Albert Albers, for his support and for the faith he has shown in me. Not only did he provide valuable input to me through his counseling, but the meetings with him were also ever motivating and encouraging.

I thank Prof. Peter Meckl of Purdue University for assuming the position as second examiner of my work and for his support and input throughout the last few years, despite the considerable geographical distance involved.

I would also like to thank Dr. Wolfgang Burger for his support and counseling during my time at the institute. His support throughout the, otherwise gruesome, bureaucratic procedures is something that I will fondly remember. Likewise, I want to thank Norbert Burkardt for his support during the bureaucratic procedures linked to the yearly extension of the DAAD-CONACYT scholarship.

Furthermore, I am thankful to all my colleagues at IPEK for the great and supportive atmosphere at the institute. I particularly thank my colleagues of the Condition Monitoring research group, Paul Martin and Tobias Pinner, for their support and the interest shown during our discussions.

A special thanks goes to Markus Frietsch, who greatly influence my decision to embark on this project.

The completion of this work could not have been possible without the help of the Mexican Council for Science and Technology (CONACYT), which carried most of the financial load throughout these years. Therefore, I am grateful to them and to their cooperation partner, the German Exchange Service (DAAD), for both their support and their belief in me.

I would like to thank all of my close friends for their support and understanding during often challenging times and for providing the necessary distraction and relaxation when I needed it most.

Finally, I want to express my profound and sincere gratitude to my family, particularly to my parents, whose unconditional love and support have been constant sources of reassurance, even at the most difficult of times and despite the great distance between us.

“Experience is the teacher of all things”

Gaius Iulius Caesar

Contents

Table of Symbols

Symbol	Unit	Description
$\hat{Q}_{ref}(s_i^*)$	-	Approximated Q-Value in RBF
$\dot{x_0}$	m/s	Linear oscillator output speed
$\dot{\varphi}$	rad/s	Rotational oscillator input speed
$\dot{\varphi_0}$	rad/s	Rotational oscillator output speed
$\dot{\varphi_A}$	rad/s	Rotational oscillator output disc speed
$\dot{\varphi_M}$	rad/s	Rotational oscillator input disc speed
Ω_{ge}	$1/min$	Gearbox entry angle speed value range
Ω_{go}	$1/min$	Gearbox output angle speed value range
F_{CA}	N	Rotational oscillator clamping force
F_{CB}	N	Linear oscillator clamping force
F_N	N	Normal force
F_R	N	Friction force
F_A	N	Clamping force
$F_{standard}$	N	Unmodulated progression of clamping force
J_A	kgm^2	Rotational oscillator output disc inertia
J_{CE}	kgm^2	Combustion engine inertia
J_{CG}	kgm^2	Added inertia of clutch and gearbox
J_M	kgm^2	Rotational oscillator input disc inertia
J_V	kgm^2	Vehicle inertia
M_C	Nm	Clutch torque
M_{CE}	Nm	Combustion engine torque
M_L	Nm	Load torque
P_a	-	Probability for softmax selection
Q^*	-	Action- value when following optimal policy π^*
$Q_{ref}(s_i^*)$	-	Reference Q-Value for nodes in RBF
Q^π	-	Action-value of policy π
R_M	m	Mean friction radius
R_t	-	Return at the time t
T_a	s	Sample rate (real time environment)
V^*	-	Value when following optimal policy π^*
V^π	-	Value of policy π
a_0	-	Action at initial state
a_T	-	Action at terminal state

a_t	-	Action at the time t
c_s	Nm/rad	Side shaft stiffness
d_{ce}	Nms/rad	Combustion engine damping
d_s	Nms/rad	Side shaft damping
$e_t(s)$	-	Value of eligibility trace for the state s at the time t
f_{eigen}	1/s	Eigenfrequency
m_b	kg	Linear oscillator mass
n_0	$1/min$	Idle speed (input)
n_{ce}	$1/min$	Combustion engine speed
n_{ge}	$1/min$	Gearbox entry speed
n_{go}	$1/min$	Gearbox output speed
r_t	-	Reward at the time t
s_0	-	Initial State
s_T	-	Terminal state
s_t	-	State at the time t
v_b	m/s	Linear relative speed
$\dot{x}$	m/s	Linear oscillator input speed
μ_{st}	-	Static friction coefficient (coefficient of adhesion)
π^*	-	Optimal policy
$\sigma_{n_{ge}}$	-	Standard deviation for gearbox entry speed
$\sigma_{n_{go}}$	-	Standard deviation for gearbox output speed
σ_a	-	Standard deviation for clamping force
σ_θ	-	Standard deviation for torsion angle
τ^*	-	Temperature for softmax selection
ω_{ce}	rad/s	Combustion engine angle speed
ω_{diff}	rad/s	Difference angular speed
ω_{eigen}	rad/s	Eigenfrequency (angular)
ω_{en}	rad/s	Engine angular speed
ω_{exc}	rad/s	Excitation angular speed
ω_{gb}	rad/s	Gearbox angular speed
ω_{ge}	rad/s	Gearbox entry angle speed
ω_{go}	rad/s	Gearbox output angle speed
ω_{st}	rad/s	Static angular speed
Δv	m/s	Relative speed
Δw	rad/s	Relative rotational speed

a	m/s^2	Acceleration
AI	-	Area index
D	Nms/rad	Simplified system damping
$h(\|\|x-c\|\|)$	-	Value of RBF in dependence of Euclidean distance from its center c
n	-	Amount of friction pads in clutch
S	-	Safety factor
VDV	-	Vibration Dose Value
ε	-	Exploration rate
Θ	rad	Torque angle value range
C	Nm/rad	Simplified system stiffness
J	kgm^2	Simplified system inertia
$M(\varphi)$	Nm	Clutch Torque
$N(s,a)$	1	Amount of visits to action pair *(s,a)*
a	-	Free parameter for RBF
$belief$	-	Belief factor in Gaussian update rule
g	m/s^2	Earth's gravity
k	-	Visits before a state is considered free of error
$n(t)_{lower}$	$1/min$	Lower envelope curve
$n(t)_{upper}$	$1/min$	Upper envelope curve
$need$	-	Need factor in Gaussian update rule
p	-	Exponent for dynamic learning rate
t	s	Time
α	-	Learning rate
γ	-	Discount rate
λ	-	Eligibility trace decay factor
μ'	s/m	Friction gradient
π	-	Policy
τ	s	Sampled time
φ	rad	Drive train torsion angle
ω	rad/s	Angular frequency
ϑ	-	Tolerance factor for disambiguation of clutch modes

Table of Abbreviations

ANN	Artificial Neural Networks
ASPM	Aviation System Performance Metrics
ATZ	Automobil Technische Zeitschrift
CAD	Computer-aided Design
CAN	Controller Area Network
CI	Computational Intelligence
CT	Computer Tomography
DC	Direct Current
DIN	Deutsches Institut für Normung
DLC	Distributed Learning and Control
DP	Dynamic Programming
EEG	Electroencephalography
FAA	Federal Aviation Administration
FSF	Full State Feedback
IPEK	Institute of Product Engineering Karlsruhe
iPeM	Integrated Product Engineering Model
JIT	Just-In-Time
MC	Monte Carlo
MDP	Markov Decision Process
Mini-HiL	Mini Hardware-in-the-Loop
MIS	Modular Interface System
NVH	Noise, Vibrations and Harshness
RBF	Radial Basis Function
RL	Reinforcement Learning
SARSA	State-Action-Reward-State-Action
SMDP	Semi Markov Decision Process
SOM	Self-organizing Maps
SuD	System under Development
TD	Temporal Difference
TD(0)	One-step Temporal Difference Learning Algorithm
TD(λ)	Multi-step Temporal Difference Learning Algorithm
TL	Transfer Learning
VDV	Vibration Dose Value
XiL	X-in-the-Loop

1 Introduction and Outline

1.1 Introduction

In the automotive development branch, the effect of noise, vibrations and harshness (NVH) play a decisive role. Also, the comfort perceived by the passenger of an automobile while driving is a determinant factor for his purchase decision.[1] One of the most prominent NVH phenomena is that of clutch judder or chatter and refers to vibrations that generate in the clutch during the synchronization of the primary and secondary sides of the powertrain.[2] Due to the ever increasing torque provided by modern combustion engines (c.f. Fig. 1.1) and an ever higher efficiency of modern drive trains, their susceptibility towards this kind of vibrations is heightened.[1]

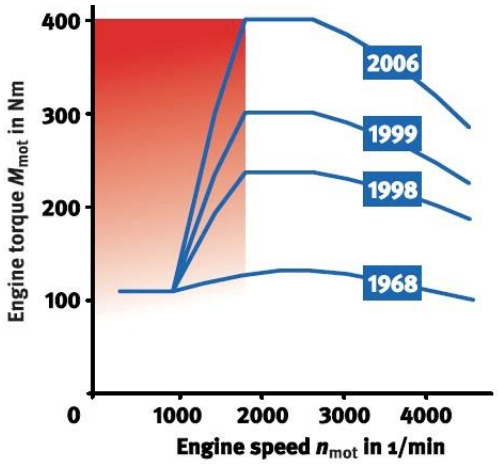

Figure 1.1: Development of the full load engine torque and speed of a 2.0 liter diesel engine[3]

For these reason, considerable effort has been invested in the prevention, or at least mitigation of clutch judder vibrations. The countermeasures can be categorized to belong into one of three types: constructive, tribological or mechatronic.[4] However, most constructive and tribological countermeasures are either costly or present practical limitations.[5] Furthermore, MOSBACH[6] points at the fact that friction vibrations

[1] Albers in the foreword of Krüger 2003
[2] Albers / Herbst 1998
[3] Zink et al. 2010
[4] Albers 2010 / Albers / Herbst 1998 and Krüger 2003
[5] c.f. 2.1.4
[6] Mosbach 2002

are generally avoided by means of case specific detail coordination and measures, which is only possible at the expense of high development and experimental effort.

The implementation of mechatronic countermeasures requires the availability of an automatic or a semi-automatic transmission. However, the availability of such transmissions is ever higher. In the United States and Japan, they have long displaced manual transmissions and make up to 80% of the market (c.f. Fig. 1.2). In Europe however, manual transmissions remain the most prevalent.[7]

Figure 1.2: Percentage of new car sales with manual transmission[8]

In the ever more important Chinese market, the demand for automatic transmissions has also risen in recent years. [9]

Most mechatronic countermeasures for clutch judder rely on active damping through a robust or adaptive control of the clutch actuator. However, in order to design these controllers, accurate models[10] are often necessary and the effectiveness of the solutions depends on their quality.[11]

Also, technical systems, and particularly automotive systems, are becoming ever more complex. This affects primarily their development, but also other aspects like their maintenance and servicing. Since vehicle components are usually conceived

[7] Albers / Matthiesen 1998
[8] Raia 2014
[9] www.researchinchina.com, 2012
[10] Such models include but are not limited to reference, adaption and simulation models.
[11] Åström / Wittenmark 2008

separately, their effect on the whole system is often unknown.[12] Therefore, the application of systematic and highly integrating processes and development environments is required for the development of whole automotive systems.[13] However, the complexity of the systems also imposes restrictions on the significance and conclusiveness of their models, particularly simulation models. Whereas the behavior of single components is relatively simple to model, their effect on the remaining system as the development process progresses becomes more difficult to describe and reproduce. Therefore, as the modelling/computing effort increases exponentially over the course of the process, the required discriminatory power and significance increase as well. This means that, in general, the effectiveness of simulation models decreases over the course of the development process.[14] (c.f. Fig. 1.3)

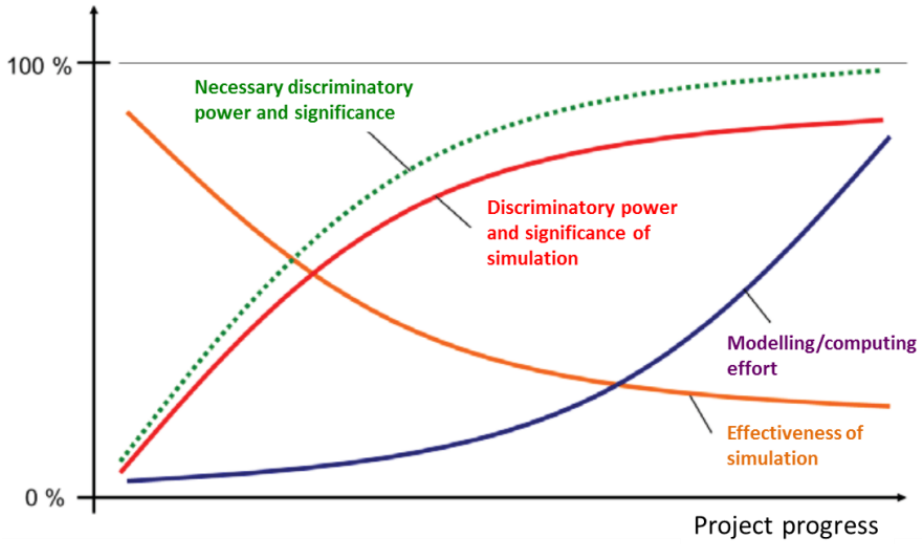

Figure 1.3: Simulation in the development process according to ALBERS AND SCHYR[14]

This rising complexity and increased information networking in technical systems, c.f. “System of Systems” and “Systems Engineering”[15], has given rise to the use of more and more methods and concepts from the computational intelligence (CI) and similar areas to handle and manage the huge amounts of information.[16]

Reinforcement Learning (RL) is one the three major subfields of machine learning. Its most distinctive feature is their ability to learn from their experienced direct interaction

[12] Drexl 1988

[13] Albers et al. 2008

[14] Albers / Schyr 2003

[15] Numerous definitions of “System of Systems” according to the field of application are available (e.g. for information-intensive orginizations by Carlock / Fenton 2001). An insight into the fundamentals and the application of “Systems Engineering” is provided by Haberfellner 2012. Also, a handbook for system engineering processes is provided by Haskins 2011

[16] Kroll 2013

with their environment. Although it originated in the field of robotics, the basic reinforcement learning framework is very flexible and thus applicable to a variety of tasks. In this work, reinforcement learning is proposed for the task of judder suppression as it is considered possible to develop a solution that does not rely on a specific model or individual system, e.g. test bench, prototype or series vehicle.

Despite reinforcement learning methods not needing models of their environment, in this work a combination of simulative and experimental approaches is preferred for the development of the judder suppressing controller. The synergetic effects of such a combination lead to an acceleration of the development process and a deeper understanding of the technical system and the task at hand.[17]

1.2 Outline

After the introduction provided in this chapter, the necessary fundamentals and an overview of the state of the art of clutch judder and reinforcement learning are presented. Furthermore, the IPEK X-in-the-Loop framework is introduced.

In the third chapter, the motivation of this work and the objectives it pursues are deduced from the state of the art.

The fourth chapter outlines the research design with which the objectives stated in the previous chapter are to be fulfilled.

The preliminary, theoretical stage of this work, in which an abstract definition of the reinforcement learning framework and its components is provided, is found in chapter five.

In the following chapter, the implementation of the reinforcement learning framework on an abstract simulation model of the drive train and the corresponding results are presented, evaluated and discussed.

In the seventh chapter the reinforcement learning framework is implemented on the IPEK Mini Hardware-in-the-Loop test bench both in simulation and on the physical test bench. The results of different RL-algorithms, including one developed in the previous stage, are again presented, evaluated and discussed.

Finally, a summary of the work and an outlook on future research topics is provided in the eighth and last chapter.

[17] Albers in the foreword of Krüger 2003

2 Fundamentals and State of the Art

This chapter contains the necessary fundamentals regarding clutch judder and reinforcement learning. Furthermore, it provides an insight into the validation process in product development, more specifically, into the X-in-the-Loop framework. Finally, an overview of the state of the art research in the area is presented.

First, the mechanical processes that underlie the generation of judder vibrations are presented. Subsequently, an overview of current methodology regarding the measurement and perception of judder vibration is provided. In the next part of the first subchapter an overview of countermeasures for clutch judder is provided, before the fundamentals of active damping of friction induced vibrations and a summary of state of the art solutions based on this principle is presented in the last part.

In the second subchapter the reinforcement learning problem and the fundamental elements of its agent-environment interaction framework are introduced. In addition, an overview of the elementary solutions to the reinforcement learning problem is provided. Finally, an overview of reinforcement learning applications and state of the art research is presented.

In the third and final subchapter, the engineering process is described. Particularly, the X-in-the-Loop framework for validation purposes is given special emphasis, since it is the foundation on which most models used for the validation of the simulation results of this thesis were designed.

2.1 Clutch Judder in automotive Drivetrains

Clutch judder is defined by WINKELMANN AND HARMUTH[18] as well as ALBERS AND HERBST[19] as a vibration in automotive drive trains during the slipping phase of the clutch. It occurs as a result of the periodically oscillating torque induced through the clutch to the remaining, dynamically separated drive train in the range of its eigenfrequency.[20]

The DIN 1311 PART 1[21] classifies vibrations according to their origination into autonomous and heteronomous vibrations. The frequency of autonomous vibrations is only dependent of the autonomous oscillating system itself, though there are

[18] Winkelmann / Harmuth 1985
[19] Albers / Herbst 1998
[20] Albers / Herbst 1998 and Albers / Herbst 2000
[21] DIN 1311 Part 1 2002

restrictions to this behavior in the case of non-linear systems[4]. In contrast, the frequency of heteronomous vibrations is determined by external influences on the oscillating system, e.g. the pulsating force excitation of a simple mass oscillator. Furthermore, heteronomous oscillating systems can underlie an additional autonomous excitation.[22]

Clutch judder vibrations can be classified analogously into either self-excited or externally excited judder.[23] The next two sections of this chapter offer an insight into the two types of judder. Afterwards, the perception by a passenger and quantification of judder are addressed. Finally, countermeasures against clutch judder are presented in the last two sections of this chapter. The last section emphasizes mechatronic, control based state of the art approaches.

2.1.1 Self-excited Judder

Self-excited oscillations are a common sight, since they appear in many technical systems where a friction contact is used to transmit a force or torque. Common examples are self-excited oscillations in brake systems[24], but there are numerous other examples[25]. Self-excited oscillations of the drive train occur as a result of the changes of the friction coefficient μ as a function of the sliding speed v_b between the clutch discs.[26] A simplified model of a self-excited oscillator according to MAUCHER[27] is depicted in Fig. 2.1.

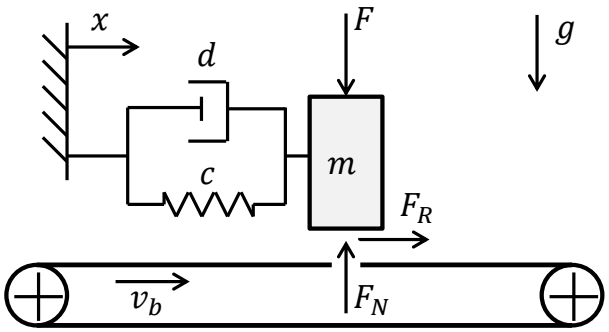

Figure 2.1: Simplified model of a simple mass self-excited oscillator according to MAUCHER[27]

[22] DIN 1311 Part 1 2002

[23] Newcombe / Spurr 1972, Winkelmann / Harmuth 1985, Albers / Herbst 1998 and Albers / Herbst 2000

[24] Spurr 1961

[25] Krüger 2003 mentions examples such as tool machines, drill standards and rail vehicles and provides the respective references

[26] Newcombe / Spurr 1972, Winkelmann / Harmuth 1985, Albers / Herbst 1998, Albers et al. 2005a and Albers; Meid 2010

[27] Maucher 1990

For this simplified system and after a linearization at the operating point, the differential equation of motion yields as follows:[28]

$$mx(\ddot{t}) + (D + F_N\mu')\dot{x}(t) + Cx(t) = 0 \qquad \textit{Eq. 2.1}$$

As evidenced by *Eq. 2.1*, μ' is of great importance for the behavior of the friction system, the friction gradient[29] can be formally defined as follows:

$$\mu' = \frac{\delta\mu}{\delta\Delta v} = \frac{\delta\mu}{\delta\Delta\omega} \qquad \textit{Eq. 2.2}$$

where μ is the friction coefficient and Δv and $\Delta\omega$ are the relative speed and rotational speed in the friction contact, respectively.

In reality, the friction coefficient, and thus its dependency on the sliding speed, is dependent on operating and environmental conditions[30], but it is possible to identify three general cases according to ALBERS AND HERBST.[31] This dependency is depicted in Fig. 2.2.

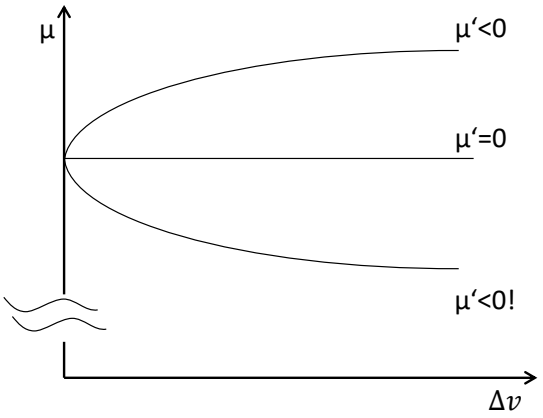

Figure 2.2: Dependency of the friction coefficient on the relative sliding speed

The upper curve in Fig. 2.2 corresponds to the case in which the friction coefficient falls with a decrease of the slipping speed. In this case the friction contact has a *damping* effect on the system.[32]

The horizontal line in the middle depicts the case in which the friction coefficient remains *unaffected* by a change in the sliding speed. Accordingly, the oscillating system is undisturbed by the friction contact.

[28] Heilig et al. 2002
[29] Maucher 1990
[30] Krüger 2003
[31] Albers / Herbst 1998
[32] Albers / Herbst 1998

Finally, the course of the lower curve exemplifies the case in which a decrease in the sliding speed entails a rise in the friction coefficient, thus a negative friction gradient.[33] In this case, the damping in the system is reduced and oscillations are *excited*.

The cycle by which self-excited oscillations in the drive train are generated in accordance with the previously presented mechanism is described by NEWCOMBE AND SPURR[34] as follows:

> *„Clutch judder is generally assumed to be caused by the coefficient of friction μ of the facing increasing as the sliding velocity decreases. The transmission line etc. deflects elastically under torque and the theory states that if for some reason the system is disturbed and the elastic deflection, for example, increased, the relative velocity between facing and opposing surface will be decreased momentarily and the μ increased. The increase in μ will result in a greater torque and a larger deflection so increasing μ and adding to the torque further still, and this will continue until the deflection reaches a maximum. The relative velocity between the surfaces now begins to increase reducing the μ and torque, and the stored elastic energy reduces the deflection until it becomes a minimum, when the whole cycle repeats itself and the system is set vibrating"*[34]

In the case of clutch judder vibrations, the rotatory oscillation system is dependent on further factors, such as the amount and mean friction radius of the friction pairings as well as the clamping force between clutch plates.[35] Furthermore, the description NEWCOMBE AND SPURR is limited to the case of self-excited judder and disregards external excitation sources.

The simulation models used in the context of this thesis are limited to the reproduction of self-excited judder.[36] The details will be presented in the corresponding chapters, however, they both rely heavily on the simplified dynamic model of the drive train according to MAUCHER[37] and depicted in Fig. 2.3.

---

[33] Winkelmann / Harmuth 1985; Newcombe / Spurr 1972, Maucher 1990, Albers / Herbst 1998 and Albers / Herbst 2000

[34] Newcombe / Spurr 1972

[35] Maucher 1990, Albers / Herbst 1998; Maucher 1990, Albers / Herbst 2000 and Albers et al. 2001

[36] The presence of externally-excited judder on the physical test bench is unavoidable as will be clarified later on in this work.

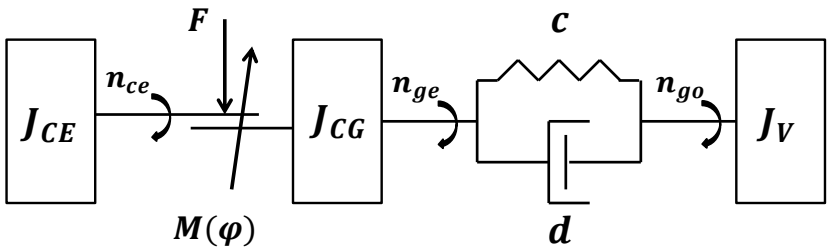

Figure 2.3: Simplified model of the power train according to MAUCHER[37]

In this simplified three-mass-oscillator model, J_{CE} is the inertia of the combustion engine, whereas J_{CG} is the aggregated inertia of the clutch discs and the gearbox and J_V is the inertia of the rest of the vehicle. Furthermore, $M(\varphi)$ is the clutch torque induced by the clamping force F and c and d are the stiffness and damping in the drive train respectively. Finally, n_{ce}, n_{ge} and n_{go} are the rotational speeds of the combustion engine, the gearbox input shaft and the gearbox output, respectively. The differential equation of motion for this system can be formulated considering the balance of torque. Due to the fact that the inertia of the vehicle is much greater than that of the clutch and gearbox it can be regarded to be infinitely high, therefore only the latter are considered to be determinant for the oscillation:

$$J_{CG} \ll J_V = \infty \rightarrow J_{CG} = J \qquad \textit{Eq. 2.3}$$

After this simplification, the equation of motion yields:[38]

$$J\ddot{\varphi} + d\dot{\varphi} + c\varphi = M(\varphi, \dot{\varphi}) \qquad \textit{Eq. 2.4}$$

Furthermore, the clutch torque is defined as follows:

$$M(\varphi, \dot{\varphi}) = F(\varphi) 2r\mu(\varphi) + A_C\dot{\varphi} \qquad \textit{Eq. 2.5}$$

In *Eq. 2.5* the periodical clamping force is $F(\varphi)$, whereas $\mu(\varphi)$ is the friction coefficient. Its torsion angle dependency is mainly due to geometrical deviations. Finally, A_C is a factor that describes the excitation due to the friction pairing in the clutch. Considering *Eq. 2.3 – Eq. 2.5* the equation of motion can be reformulated and yields:

$$J\ddot{\varphi} + (D - A_c)\dot{\varphi} + C\varphi = F(\varphi) 2r\mu(\varphi) \qquad \textit{Eq. 2.6}$$

The eigenfrequency of the simplified model is given by:

[37] Maucher 1990
[38] Albers / Herbst 1998

$$\omega = \sqrt{\frac{C}{J}} \qquad \textit{Eq. 2.7}$$

In modern drive trains, the eigenfrequency lies in the range of approx. $8Hz - 12Hz$.

KRAUSE[39] performed some of the first experiments on judder vibrations in dry and wet clutches. He observed the vibrations on friction pairings where the friction coefficient rises with a falling slip in the clutch. Furthermore, he exposes discrepancies between static friction coefficients and dynamic friction coefficients during judder vibrations.

On the basis of this experimental analysis, GÖRLICH[40] shows the effects of a variation of the stiffness, damping, the friction gradient and the inertia of a simulated two-mass rotational oscillator. Furthermore, he shows that the amplitude of the judder vibration decreases with an increasing stiffness and/or inertia in the drivetrain. This was also later observed by MAUCHER[41]. Also, PFEIFFER[42] offers an insight into friction vibrations and offers clutch judder among other cases as an example.

In the context of this work only dry clutch systems are considered. For a summary of references regarding self-excited judder in wet clutch systems please consult KRÜGER[43].

2.1.2 Externally excited Judder

Judder vibrations in service are often observed, even when friction facings known to have a nearly constant friction gradient are in use. This is mainly due to geometrical deviations in the system and / or external periodical excitation sources.[44]

A simplified model can be obtained analogously to the self-excited one-mass oscillator depicted in Fig. 2.1 by substituting the constant force F by a periodically oscillating external exciting force $F(t)$. The simplified model is shown in Fig. 2.4.

[39] Krause 1965
[40] Görlich 1968
[41] Maucher 1990
[42] Pfeiffer 1992
[43] Krüger 2003
[44] Newcombe / Spurr 1972

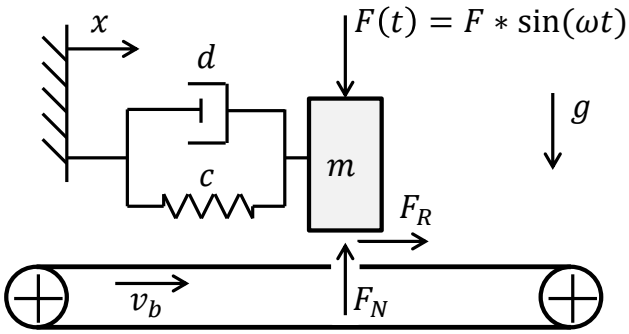

Fig. 2.4: Simplified model of a simple mass externally excited oscillator according to MAUCHER[45]

According to ALBERS AND STIER[46], there are two basic mechanisms of forced excitation: forced movements of the entire clutch system and a superposition of geometrical deviations. Forced movements of the clutch can occur as a result of the higher combustion pressures in modern combustion engines. The high pressures lead to an axial oscillation of the crankshaft and the clutch system, consequently causing an oscillating actuation of the clutch.

WINKELMANN AND HARMUTH[47] and later ALBERS AND HERBST[48] analyzed the dependency of misalignments and judder vibrations. According to WINKELMANN AND HARMUTH[47] at least two deviations from different types need to occur in order to cause clutch judder vibrations, whose excitation frequency is dependent of the rotational speeds of the underlying deviations.

The Geometrical deviations that can lead to clutch judder are classified according to ALBERS AND HERBST[48] and ALBERS AND KRÜGER[49] in three categories: deviations rotating with engine speed, deviations rotating with gearbox speed and static deviations or misalignments. An overview of these deviations and a list of examples are contained in Table 2.1.

[45] Maucher 1990
[46] Albers / Stier 2010
[47] Winkelmann / Harmuth 1985
[48] Albers / Herbst 1998
[49] Albers / Krüger 2003b

Category 1 **Deviations rotating with engine speed** $\omega = \omega_{en}$	Category 2 **Deviations rotating with gearbox speed** $\omega = \omega_{gb}$	Category 3 **Static deviations / misalignments** $\omega = \omega_{st} = 0$
Crankshaft axial and radial run-out Periodic run-out of the diaphragm spring fingers Skewed lift-off of the clutch pressure plate Run-out or parallelism deviations of the clutch pressure plates	Thickness variation of the clutch disc Irregularities of the clutch facing spring Radial run-out of the gearbox input shaft	Radial and angular misalignment of gearbox and the engine Skewed clutch engagement

Table 2.1: Classification of deviations and misalignments according to ALBERS AND STIER[50]

Depending on the combination of deviations present in the drive train, three characteristic types of externally excited judder are observable:[51]

- *Engine speed dependent judder*

Result of the superposition of deviations from the first and third category. The excitation frequency of the oscillation results from the difference of the rotational speeds of the deviations involved:

$$\omega_{exc} = \omega_{en} - \omega_{st} = \omega_{en} \qquad \textit{Eq. 2.8}$$

- *Gearbox speed dependent judder*

Result of the superposition of the second and third category of deviations. Analogously, the excitation frequency of the oscillation yields:

$$\omega_{exc} = \omega_{gb} - \omega_{st} = \omega_{gb} \qquad \textit{Eq. 2.9}$$

- *Differential speed dependent judder*

Result of the superposition of deviations of the first and second category. The excitation frequency of the oscillation is given by:

$$\omega_{exc} = \omega_{en} - \omega_{gb} = \omega_{diff} \qquad \textit{Eq. 2.10}$$

[50] Albers / Stier 2010

[51] Albers / Herbst 1998, Krüger 2003 and Albers / Stier 2010

It is worth mentioning, that all three types of externally excited judder are present to a certain extent in real drive trains.[52] This is due to the fact that all deviations fluctuate within a (sometimes extremely narrow) set of tolerances, which makes externally excited judder a statistical problem.[53]

ALBERS AND HERBST[53] and KRÜGER[54] offer a more detailed description of the three types of externally excited judder, whereas ALBERS AND STIER[52] perform an experimental investigation of the effect of variation of geometrical deviations on judder vibrations. Furthermore, they present an approach for the modelling of judder excitation mechanisms.

The focus of this work lies on systems with dry clutches, however, BAUER[55] lists additional deviations that can lead to externally excited judder vibrations in systems with wet clutches.

2.1.3 Perception and Quantification of Judder

As has been mentioned above, the main cause for judder is an oscillating torque induced through the clutch to the remaining, dynamically separated drive train. However, the passengers perceive it in the form of an oscillation of the longitudinal acceleration of the vehicle.[53] These oscillations are perceived more or less strongly by the passengers in dependence of their amplitude and frequency[56]

The consideration of customer comfort and the perception of it during the use phase of a product is a main requirement for a successful product development, especially so in the automotive industry.[57] Depending on the task at hand, it makes sense to differentiate the measurement of physical parameter values from their perception by a human being.

A common evaluation or assessment method in drive train development is the so called ATZ-scale, in which a grade of 1 to 10 is given by an evaluator according to his perception of noise and vibrations in a vehicle. Whereas the scale offers a basis for the comparison and evaluation of NVH phenomena, it is highly subjective, since different evaluators tend to have a different perception of comparable situations, depending on their experience and training.

---

[52] Albers / Stier 2010
[53] Albers / Herbst 1998
[54] Krüger 2003
[55] Bauer 1993
[56] Verein Deutscher Ingenieure 2009
[57] Albers / Albrecht 2002, Albers / Albrecht 2004 and Albrecht 2005

For this reason, considerable effort has been invested in the objectivization of the perception of comfort and the evaluation of NVH phenomena such as judder. It is often difficult to formulate a statement about customer comfort perception in early stages of development, since some effects on it might only become apparent late in the product development process. For example, DREXL[58] points at the fact, that it is hard to make predictions regarding the vibration of a fully assembled drivetrain from the vibration of its components. For this reason, ALBERS AND ALBRECHT[59] analyzed possibilities to predict the perceived comfort of passengers on a real life vehicle during start-up maneuvers. The work includes the standard regression modelling of passenger comfort as well as a method employing an artificial neural network (ANN). The basis for the modelling are tests with laymen evaluators in a test vehicle with start-up properties that could be varied in a customer-relevant range. The perception of comfort is measured with the help of an evaluation scheme. Afterwards, the approximation and prediction qualities of both methods, the standard regression and the ANN-method, are compared and the ANN-based method considered superior upon comparison. The result is consistent throughout real and simulated start-up procedures, allowing an estimation of customer comfort at early stages of the development process. Later, LERSPALUNGSANTI[60] developed a user friendly platform for the use of the ANN-based method so that it allows the developer to vary experimental parameters such as the clutch actuation or the damping in the system in order to determine their effect on customer comfort. Furthermore, he extends the applicability of the method to the NVH-phenomenon of gear rattling.

Often, however, it is necessary to provide a measure of clutch judder in the form of a measured numeric value, free from human interpretation. Based on the work of MÜLLER[61], KRÜGER[62] and KARRAR[63] present a method to evaluate judder vibrations with the help of a numeric value. Instead of taking the oscillation of the clutch torque into consideration, the value of the area of the envelope curves around the gearbox input speed is used to evaluate the judder vibrations. The value is called the “area index”. The upper and lower envelope curves of the gearbox speed during a start-up and synchronization maneuver are depicted in Fig. 2.5.

[58] Drexl 1988
[59] Albers / Albrecht 2002, Albers / Albrecht 2004 and Albrecht 2005
[60] Lerspalungsanti 2010
[61] Müller 2002
[62] Krüger 2003
[63] Karrar 2009

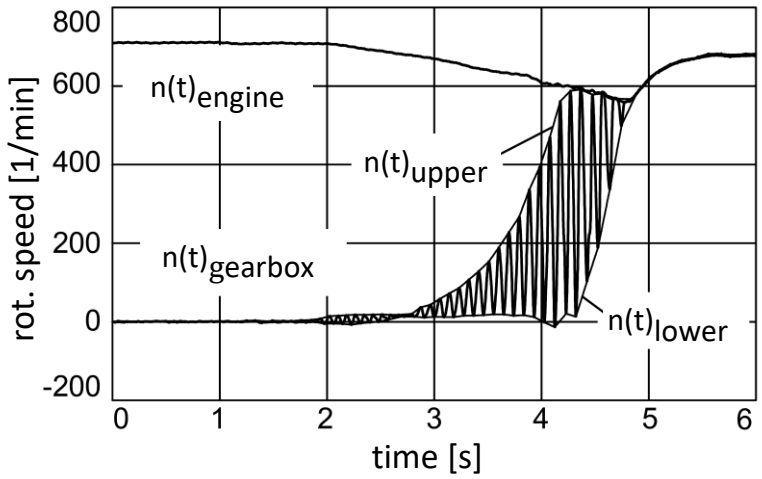

Figure: 2.5: Envelope curves of the gearbox speed after KRÜGER[62]

The “area index” is computed as follows:

$$\mathrm{AI} = \int (n(t)_{upper} - n(t)_{lower}) d\tau \qquad \textit{Eq. 2.11}$$

However, KRÜGER[62] points at three disadvantages this method of evaluation bears:

- The sensors in conventional drive trains do not allow recognition of the direction of rotation and are considerably inaccurate when the rotational speed is close to zero. Thus, leading to a faulty calculation of the area index.
- The engine-speed decrease during a clutch synchronization maneuver has a major effect on the area index. In some cases, strong judder can be present despite a small area index.
- The area index does not provide information about the frequency range of the drive train vibrations.

Furthermore, KRÜGER also describes the possibility of describing the judder vibration with the transfer function of the drive train. The transfer function is determined using a multi-mass model of the drive train and allows to determine the frequency and amplitude of the judder vibrations.

GRIFFIN[64] presents a detailed description of the effects of vibrations on human beings. Later, he focused on the effect of vibrations in vehicles on the passengers and introduces the application of the “Vibration Dose Value (VDV)”[65] to measure vibrations. The VDV is a value meant to evaluate the effect of mechanical vibrations on human beings. A technique called root-mean-quad is used to ensure its sensitivity towards

[64] Griffin 1990
[65] Griffin 2007

peaks in the acceleration levels. Thus, the VDV for a vibration with the duration T is given by:

$$\mathrm{VDV} = \left(\int_0^T a^4(t)d(t)\right)^{1/4} \qquad \textit{Eq. 2.12}$$

The VDV would be an appropriate measure to evaluate judder vibrations and provide a numeric value to them. However, in this work a comparison of different methods to evaluate judder vibrations, including the VDV, is performed in 5.4 and the most suitable is used for the definition of the reward-signal.

2.1.4 Countermeasures for Clutch Judder

There are different approaches and methods to reduce judder vibrations in drive trains and they can be classified in three groups: constructive, tribological and control based (mechatronic) approaches. In this subchapter an overview of the first two groups is presented, before the mechatronic approaches, referred to as active damping approaches, are emphasized in the next.

2.1.4.1 Constructive Countermeasures for Clutch Judder

The effect of judder vibrations can be reduced in different ways according to their underlying causes.

The occurrence of externally excited judder vibrations due to geometrical deviations and assembly errors can be reduced by narrowing the production and assembly tolerances, however, this necessarily translates to higher costs in both fields.[66] Keeping in mind that externally excited judder requires deviations of two different types to occur, it might prove sensible to tackle only the one that is easier / cheaper to handle.[67] A further possibility is the development of robust clutch systems as proposed by ZINK ET AL.[68]. In their approach, the authors show a reduction of axial oscillations of the crank shaft by an optimized bearing of the clutch force actuator.

Constructive solutions also include the deployment of additional components meant to ameliorate the occurrence of vibrations in the drive train, despite any deviations and errors in it. For example, JÖRG[69] proposes a rotational vibration damping unit in the form of an additional component built in behind the gearbox. It can be described as an additional inertia, thus a flywheel, mounted on a damping ring of polymer material. However, the use of these components is only possible at the expense of higher costs

[66] Albers / Herbst 1998 and Krüger 2003
[67] Krüger 2003
[68] Zink et al. 2002
[69] Jörg 1988

due to their production and assembly cost. Furthermore, their inclusion in the drive train necessarily imposes restrictions on space and weight.

Finally, it is possible to reduce the amplitude of vibrations in the drive train by increasing its damping. This can be achieved either by increasing the damping of its components individually and/or that of the assembled drive train. However, an increase in damping implicates a lower degree of energy efficiency due to dissipation effects.

A very interesting and promising concept is presented by KOOY[70] where the concept of a centrifugal pendulum is applied to the clutch, thus realizing a long-sought-for compromise between a “merely” dampened clutch disc and dual mass flywheel. However, in high torque systems, such as heavy load vehicles, its implementation often leads to load spikes leading to so called “impact situations”. These may lead to a stalling of the engine or similar, undesired scenarios. This issue could be addressed through an accordingly customized engine control and / or the implementation of high-capacity-springs in the clutch.

2.1.4.2 Tribological Countermeasures for Clutch Judder

As mentioned in chapter 2.1.1 self-excited judder occurs when the friction coefficient in the contact decreases with the slip, but disappears if it either remains constant or if it rises with a decrease in slip.

ALBERS ET AL.[71] point at the fact that the use of friction pairings showing a positive friction coefficient gradient would avoid self-excited judder entirely and would additionally have a dampening effect on externally excited vibrations. However, they also point at the fact that no friction pairings known to date present such behavior throughout the whole temperature scale in dry running systems and would therefore require an improved heat removal at friction contact.

MAUCHER[72] also points at the dependence on the temperature of the friction coefficient gradient of pairings that showed a constant or slightly positive gradient during experiments. High thermal stress can lead to the gradient becoming negative, thus the effect is known as “fading”.

JÜRGENS AND FISCHER[73], however, show that a positive friction coefficient gradient can be obtained in wet running systems with organic friction pairings in combination with a certain type of oil and additives.

---

[70] Kooy 2014
[71] Albers / Herbst 1998
[72] Maucher 1990
[73] Jürgens / Fischer 1988

Furthermore, some tribological systems are known to show a negative, thus self-exciting, friction coefficient gradient only in the run-in period. This effect is known as “green shudder”.[74]

2.1.5 Mechatronic Countermeasures for Clutch Judder

This subchapter is dedicated to offer an overview of different mechatronic approaches and implementations of judder suppression through active damping of vibrations.

RABEIH AND CROLLA[75] formulate the mathematical description of judder vibrations and determine its dynamics to be entirely predictable as long as all required input variables regarding the state of the drive train are known, thus they conclude it should be possible to reduce the torsional vibration using an intelligent controller in the form of a robust or an adaptive controller. However, they do not present nor specify a concrete solution, which hints at the questionable practicality of the approach.

ALBERS AND KRÜGER[76] demonstrate the possibility of implementing such controllers on a drive train simulation and a corresponding test bench. Later, they introduced the simplified model for the control of friction induced vibrations, depicted in Fig 2.6.

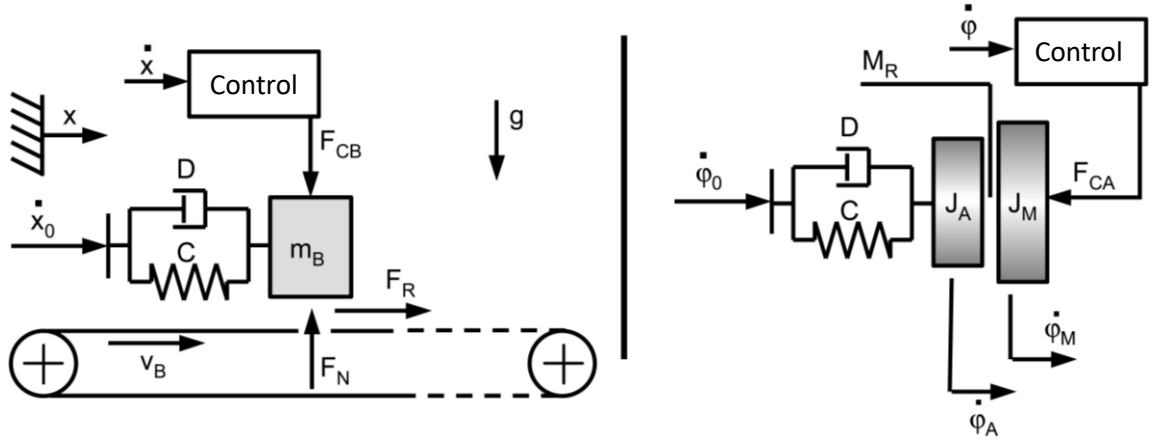

Figure 2.6: Simplified model for the control of friction induced vibrations according to ALBERS AND KRÜGER[77]; left: translational friction vibration system; right: rotational friction vibration system

On the basis of this simplified model, KRÜGER[78] presents an active clamping force control using the root-locus method and a full state feedback (FSF) method both in simulation and on a reduced drive train test bench. The results show a significant improvement compared to open loop clutch engagement. Also, high requirements on

[74] Mosbach 2002
[75] Rabeih / Crolla 1996
[76] Albers / Krüger 2002a
[77] Albers / Krüger 2003b
[78] Krüger 2003

actuator dynamics and sensor signal quality are found to be of great importance. However, neither the effect of wear in the system in the long term nor the influence of different degrees of externally excited judder on the controller performance is considered.

A synchronization assistance solution, as well as an optimal standing start controller for automated manual transmissions is presented in DOLCINI ET AL.[79]. Furthermore, a "friction-coefficient observer" is introduced. The observer includes a least square algorithm in order to react to slow changes in the relationship between the clamping force and the hydraulic piston position due to wear and squashing of the flat spring. The results are validated on a real life vehicle.

ALBERS ET AL.[80] introduce both a PI and a Fuzzy-Logic controller for an electromechanical clutch actuator on a reduced drive train test bench. The PI controller was implemented using operating maps calculated in real time. The fuzzy logic controller was implemented following the Mamdani approach. The simulation and test bench results are promising, but a need for adaption of the controllers over the lifetime of the clutch is deemed necessary.

A robust controller to suppress clutch judder is presented by NAUS ET AL.[81]. The authors apply the H∞-controller synthesis for the robust controller design. Furthermore, additional control of the rotational speed of the motor is necessary for this solution. The validation of the concept takes place in drive train simulations. The simulation model includes the actuation and the dynamics of measurement devices.

Finally, PINTE ET AL.[82] and later DEPRAETERE ET AL.[83] use a policy search reinforcement learning algorithm for the hydraulic open-loop engagement of a wet clutch. The authors employ a policy gradient method with parameter exploration unlike the value based method employed in this work. In their approach, no feedback of the system state is considered, although it is mentioned that information about possible state variables such as oil pressure and temperature could be included in future research. In their work, the authors already hint at the fact that reinforcement learning algorithms do not require a model of the environment, in this case the test bench, to compute a working operation strategy. However, the authors note that even an inaccurate model can be used to compute preliminary solutions in simulations. The lack of feedback about the state of the system leads to a process more comparable to a Monte Carlo method in which an evaluation is only possible after a whole synchronization maneuver is

[79] Dolcini et al. 2010
[80] Albers et al. 2010b
[81] Naus et al. 2008 and Naus et al. 2010
[82] Pinte et al. 2010
[83] Depraetere et al. 2011

completed. In this thesis, better results are expected from the implementation of an incremental value based method with system state feedback.

2.2 Reinforcement Learning

Machine learning is a subfield of computer sciences and artificial intelligence that focuses on systems capable of learning data instead of merely following programmed routines.[84] A more specific definition is provided by MITCHELL[85], who defines artificial intelligence as follows: “A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E*”.

The learning entity in a machine learning algorithm or system is commonly denominated as agent. However, providing a definition of an agent that is widely accepted is rather difficult, as WOOLDRIDGE AND JENNINGS[86] point out. In the context of reinforcement learning, SUTTON AND BARTO[87] describe the agent simply as the learner and decision maker but do not provide any particular characteristics to it.

In machine learning, there are three main categories of methods: supervised learning, unsupervised learning and reinforcement learning. They differ from each other in the mechanism by which the learning entity, the agent, acquires and evaluates new information regarding a learning task.

According to MOHRI ET AL., *supervised learning* is the machine learning task of inferring a function from labeled training data. This implies that the learning entity is given feedback about the desired outcome of the learning process by a perfect teacher, thus supervised. Training data is required to learn an inferred function, which can later be used for mapping new examples. Some common approaches and algorithms that belong to this category are most artificial neural networks[88], decision trees[89] and case-based reasoning technology[90] among many others.

In *unsupervised learning* the learner attempt to find a pattern in a set of data without any information regarding an error or a reward during the recognition. The examples the learner works with are unlabeled and its purpose is generally to find a hidden structure in the data, thus of data mining. Some of the most prominent approaches in

[84] Mohri et al. 2012
[85] Mitchell 1997
[86] Wooldridge / Jennings 1995
[87] Sutton / Barto 1998
[88] Bishop 1996
[89] Rokach / Maimon 2008
[90] Lenz et al. 1998

unsupervised learning include different clustering approaches like hierarchical clustering[91], a type of artificial neural networks called self-organizing maps (SOM) or Kohonen network[92] and association rule learning algorithms like the Apriori algorithm[93].

Finally, in *reinforcement learning (RL)* algorithms, the learning takes place in the form of an interaction between the learner, or agent, and its environment. They differ from supervised learning methods in that they lack the knowledge provided by an external teacher. The acquisition of information results from a trial-and-error mechanism, throughout which the agent strives to learn by itself through interaction. This is of particular advantage when knowledge about the right behavior at given situations is not available. However, the inherent problem to RL is the so called *"curse of dimensionality"*[94] that describes the exponential growth of the state space as a response of a linear growth in state variables and the explosion in the computing effort RL applications might incur as a consequence.

In the following subchapters an overview of the reinforcement learning problem, the basic elements of its framework and the elementary solution methods are introduced in accordance to SUTTON AND BARTO[95], whose work is widely regarded as the most influential in the subject.[96] Afterwards, an overview of the state of the art of RL-methods and their application in different domains is presented.

2.2.1 The Reinforcement Learning Problem and its Elements

In this subchapter an overview of the elements of reinforcement learning is provided. Subsequently, the Markov property and Markov decision processes (MDP) are introduced.

In simple words, a reinforcement learning algorithm can basically be described as an algorithm which learns what to do in a given situation in order to maximize a numerical reward-signal. Thus, its result is a mapping of the actions the agent can perform and the different states of the environment.

As mentioned previously, RL is different from supervised learning since it lacks the external input provided in the form of examples by an external teacher. Whereas this feature may seem disadvantageous, it is a key element of RL since the learning agent

---

[91] Hastie et al. 2009
[92] Kohonen 1982
[93] Agrawal / Srikant 1994
[94] Bellman 1957
[95] Sutton / Barto 1998
[96] Röttger 2009 and Szepesvári 2010

should be able to learn based on its own experience when confronted with uncharted territory. This feature is also the reason for one of the characteristic challenges in reinforcement learning: the trade-off between *exploration* and *exploitation*. On the one hand, the agent has to take the actions it has learnt to be more effective in a given situation, but on the other hand, it needs to discover what these actions are first.

The final distinction made by SUTTON AND BARTO[97] is that RL considers the whole problem of the goal-oriented agent-environment interaction, where other approaches generally are restricted to subproblems. In RL, other machine learning methods (e.g. supervised learning methods) are used for specific reasons, in order to achieve a goal. This characteristic has led to a greater contact between artificial intelligence and other engineering disciplines.

Finally, a reinforcement learning problem or task can be *episodic* or *continuous*. Episodic tasks have at least one clearly defined *terminal state*. Such is the case of a game of chess, where a game ends with a checkmate or a draw. There is a vast number of different states that can be regarded as a terminal state, since they all represent the end of the game. A *continuous* task, on the other hand, has no specified terminal state, although very extensive episodic tasks are sometimes regarded as continuous. An example would be the temperature controller of a fluid container in any chemical plant. At some point in the future the plant will no longer stand, but since the precise end is not known, this very lengthy episodic task can be regarded as continuous.

Two elements of reinforcement learning have been identified so far, the agent and the environment. Beyond these two, there are four important subelements of a RL system: a reward function, a value function, a policy and a model of the environment, whereas the latter is optional.[98]

2.2.1.1 Reward function and returns

A reward function is used to define the goal in a RL problem. Its purpose is to associate a perceived state, or the combination of taking an action at a certain state (state-action pair), with a numerical signal that indicates the desirability of that state or state-action pair. The sole purpose of a RL agent is to maximize the reward-signal in the long run. However, a reward-signal defines what immediate good or bad events for the agent are, regardless of what can happen later on. Reward-signals cannot be altered by the agent and can, in general, be of stochastic nature.

It is essential that the design of the reward actually describes what the goal of the agent is. While it might make sense to reward subgoals achieved by the agent, it might also

[97] Sutton / Barto 1998
[98] Sutton / Barto 1998

lead to an undesired behavior or simply learning the wrong task. An excellent example is yet again provided by the game of chess. In order to win a match, it might make sense to gain control over the middle of the board or take the opponents pieces. However, the game is only won by achieving a checkmate, regardless of how many pieces each of the players have remaining. It is therefore sensible to reward the agent for winning a match only.

Thus far, the concept of a long term reward has been used without providing a proper formal definition of it. In the simplest of cases, the long term reward yields the sum of rewards accumulated after each decision. However, knowledge about future rewards may not necessarily be without error, thus RL seeks to maximize the long term *expected return*, where the return R_t is a specific function of the reward sequence.[98] In the simplest case, the return R_t at the time t for an episodic task yields: the sum of the rewards:

$$\mathrm{R}_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots + r_T \qquad \textit{Eq. 2.13}$$

where T is the final step.

However, this approach is problematic since it weighs all rewards equally and it requires the existence of a terminal state at the time T. For continuous tasks, the latter is a problem since it can easily lead to the return being infinite since $T = \infty$. Maximization is not possible if the return for all policies is infinite. For episodic tasks, even if $T < \infty$, weighing all rewards equally only makes sense if the transition into future states is deterministic. This is often not the case due to the explorative nature of action selection methods and/or a stochastic environment, where only the probability of future rewards can be determined.

Thus, the concept of *discounting* is introduced in order to determine how short- or farsighted an agent should be when assessing future rewards. The return R_t is now computed considering a *discount rate* γ and yields:

$$\mathrm{R}_t = r_{t+1} + \gamma * r_{t+2} + \gamma^2 * r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k * r_{t+k+1} \qquad \textit{Eq. 2.14}$$

The discount rate has to satisfy the rule $0 \leq \gamma \leq 1$. For $\gamma = 0$ the agent takes only immediate rewards into consideration when selecting an action. Future rewards are taken into account more strongly as γ approaches 1.

2.2.1.2 Value function

The value function is used to determine what behavior is good in the long run. The value of a specific state is the total amount of reward an agent can expect to accumulate over the future, starting from this state. In contrast to reward functions, the value functions take into account the long term desirability of a state (or state-action

pair), taking into consideration the states that are likely to follow and their desirability. A very simple example of the difference between reward and value can be provided by looking at the simple binary tree depicted in Fig. 2.7.

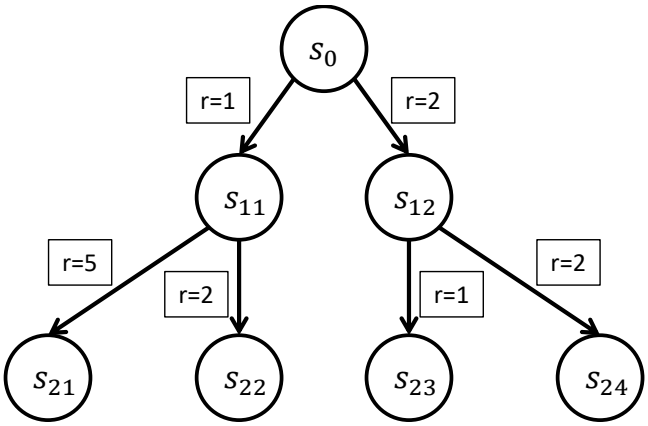

Figure 2.7: Exemplary binary tree: The transition from the state s_0 to the state s_{11} bears the reward $r = 1$. Rewarding the transition into other states is analogous. For simplicity purposes the rewards are not labeled.

It is evident that if an agent transitions down the tree solely pursuing the greatest reward at any given state it will achieve a maximal reward of $r = 4$ when it ends up in s_{24}. The greatest possible reward, however, is achieved if the agent ends up in s_{21} where he would have acquired $r = 6$. In this simple case, the value of s_{11} is higher than the value of s_{12}, even though the reward perceived by the agent for transitioning into s_{12} is initially higher.

SUTTON AND BARTO[99] offer an analogy with human beings by considering the reward to be like pleasure (if high) and pain (if low), whereas the values correspond to our more refined and farsighted judgment. Sometimes human beings will accept pain or forgo pleasure in the short term to achieve goals that are important to them.

Even though one could argue that without rewards there can be no value and that the purpose of considering value is only to maximize rewards, it is values with which it is more worthwhile working. Thus, estimating values is a central activity in most reinforcement learning methods.

SUTTON AND BARTO[99] state that methods that do not rely on value function estimation have been used to solve RL problems. Most of these methods search directly for a solution (policy) in the solution space. The authors denote these methods as evolutionary due to the fact that the solutions (individuals) produced with these methods are incapable of learning during their individual life time. They deem these methods appropriate for tasks in which the solution space is sufficiently small or the environment feedback to the agent is problematic.

[99] Sutton / Barto 1998

The definition of the value function at a given state s can be defined as the expected return R_t described previously, when following a given policy π starting from said state:

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k * r_{t+k+1} | s_t = s\} \qquad \textit{Eq. 2.15}$$

Thus far the value of the states has been considered regardless of the actions taken, whereas in RL it makes more sense to take the interaction with the environment, thus the action available for selection, into account as well. Therefore, action-value functions are introduced in order to evaluate the value of state-action pairs, i.e. to evaluate the value of performing an action a at a given state s:

$$Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k * r_{t+k+1} | s_t = s, a_t = a\} \qquad \textit{Eq. 2.16}$$

Analogously, $Q^{\pi}(s,a)$ describes the expected return when starting from the state s and selecting the action a while following the policy π.

One fundamental property of value and action-value functions in the context of RL (and dynamic programming) is that they satisfy a set of particular recursive relationships. For every policy π and every state s the following can be formulated about the value of s and the value of the probable successor states:

$$\begin{aligned} V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} &= E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\} \\ &= E_{\pi}\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s\} \\ &= \sum_{a} \pi(s,a) \sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_{t+1} = s'\} \\ &= \sum_{a} \pi(s,a) \sum_{s'} P^a_{ss'} [R^a_{ss'} + \gamma V^{\pi}(s')] \end{aligned} \qquad \textit{Eq. 2.17}$$

Analogously, this recursive relationship can be formulated for the action-value function as follows:

$$\begin{aligned} Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\} &= E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\} \\ &= E_{\pi}\{r_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_t = s, a_t = a\} \\ &= \sum_{s'} \mathcal{P}^a_{ss'} \Bigg[\mathcal{R}^a_{ss'} \\ &\quad + \gamma \sum_{a} \pi(s',a') E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+2} | s_{t+1} = s', a_{t+1} = a'\} \Bigg] \\ &= \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma \sum_{a} \pi(s',a') Q^{\pi}(s',a') \right] \end{aligned} \qquad \textit{Eq. 2.18}$$

Each of these equations is a so called *Bellman equation*[100]. It expresses a relationship between the value of a state-action pair and the values of its successor pairs. Analogously, for state-value functions, it expresses the value of a state and its successors.

The concept of optimal value functions and optimal action-value functions will be briefly introduced after the characteristics of Markov decision processes are defined later in this subchapter.

2.2.1.3 Policy

A policy defines the behavior of the learning agent at any given time. It can be described as a mapping of perceived states of the environment to actions to be taken when in those states. Roughly speaking, the numeric values in a policy can be considered a mapping from states to probabilities of selecting each possible action. Policies may have the form of simple lookup tables or functions, in some cases however, they may require extensive computation. A policy corresponds to the set of stimulus-response associations in psychology.[101]

In order to find a good, or even an optimal policy, the agent needs to address the previously mentioned trade-off between *exploration* and *exploitation*[102]. Therefore, an appropriate action selection method is almost always part of the policy. On the basis of the values of the value function at a given state, the action selection method determines which action is to be taken next. The simplest selection method involves always selecting the action that yields the highest value, thus its name: a *greedy selection* method. This method consists of *exploitation* of knowledge only. However, pure exploitation is likely to lead to a non-optimal policy.

One of the simplest, yet effective, action selection methods is the *ε-greedy* method. In this policy a percentage ε of the actions taken is selected at random. This explorative behavior of the agent may lead him to find better policies instead of converging towards

---

[100] Bellman 1957

[101] The Russian Nobel Prize holder Ivan Petrovich Pavlov is widely regarded as the pioneer in the study of stimulus-response associations and his work on classical conditioning is considered the basis of behaviorism. His most famous experimental contribution involves the conditioning of a dog using an acoustic signal and food. Over an extended period of time, Pavlov rang a bell before feeding a dog. He then realized that afterwards, he could induce the dog to salivate after it heard the bell ring even in the absence of food after the sound. The dog had associated the acoustic signal with the imminent feeding. For an insight into behaviorism theory refer to Baum 1994.

[102] The exploration vs. exploitation trade-off is known in control engineering as the conflict between identification (or estimation) and control, for example c.f. Witten 1976. In the context of genetic algorithms, Holland 1992 refers to it as the conflict between the need to exploit and the need for new information.

suboptimal ones. SUTTON AND BARTO[103] demonstrate the effectiveness of the ε-greedy action selection in an example regarding an n-armed bandit problem[104].

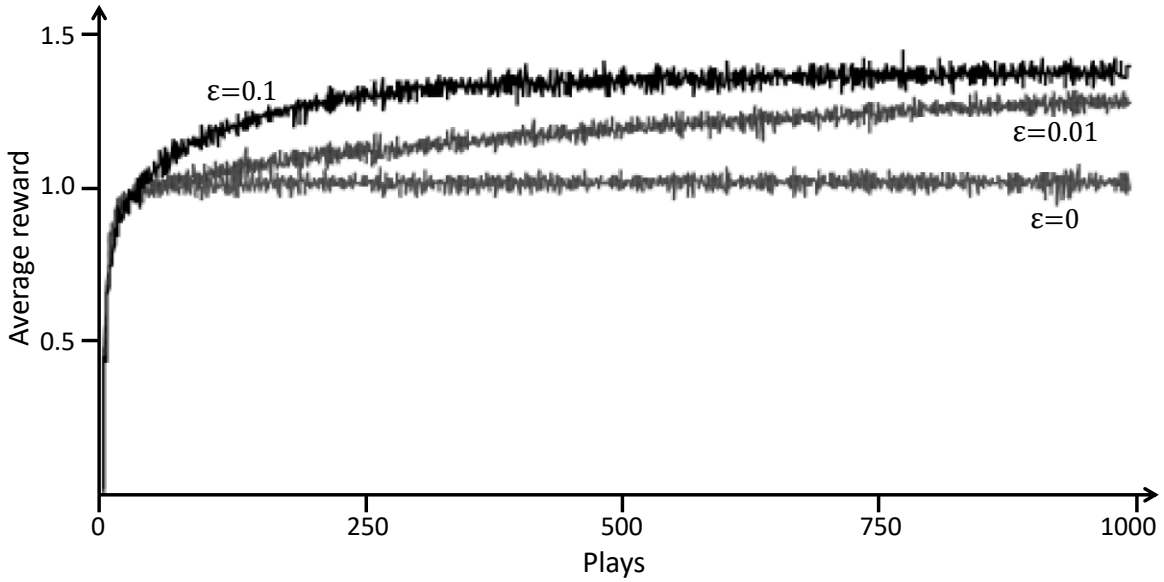

Figure 2.8: Average performance of ε-greedy methods on the 10-armed bandit problem over 2000 tasks. One task consists of 1000 plays.[103]

Initially, the greedy selection method outperforms the explorative methods over the course of a very small number of plays. Shortly afterwards, the explorative ε-greedy methods discover better policies and are able to achieve higher rewards. However, higher exploration rates hinder the convergence behavior of the policy, so an optimum rate needs to be computed, usually empirically. Also, the value of the exploration rate, ε, is usually reduced over time in order to explore more at the beginning of a task and exploit later on, when more information has been gathered. In the context of this work, ε was defined to obey the following rule:

$$\varepsilon_{k+1} = \varepsilon_k * \mu, \text{ where } 0 < \mu < 1 \qquad \textit{Eq. 2.19}$$

The resulting dynamic progression of ε over the episodes of a RL task can be taken from Fig. 2.9.

[103] Sutton / Barto 1998

[104] An n-armed bandit game can be seen as a slot-machine with n levers that can be pulled by the player. At every turn the player can choose to pull one lever in order to cash the jackpot. The goal is to maximize the cumulative reward through jackpots over a certain number of turns.

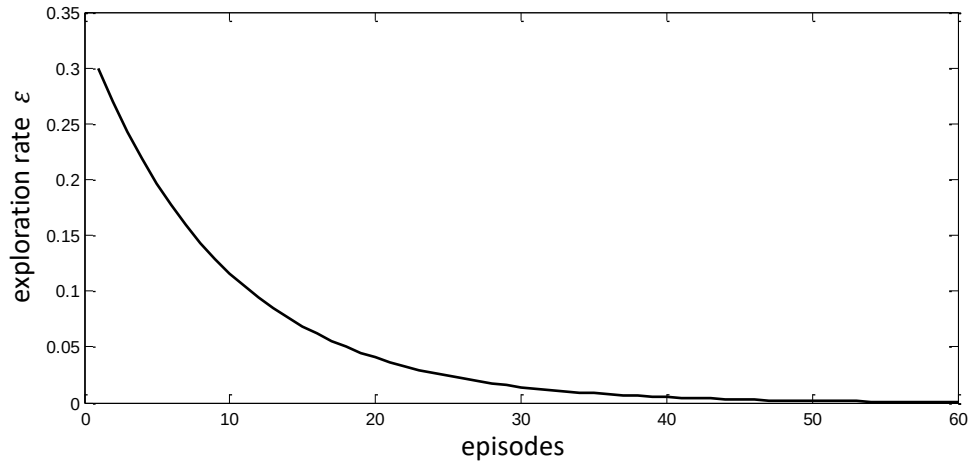

Figure 2.9: Progression of the exploration rate ε for $\varepsilon_1 = 0.3$ and $\mu = 0.9$.

The ε-greedy action selection method is an effective and popular approach to balance exploration and exploitation. However, it has one main drawback: it is as likely to choose the worst appearing action as it is to choose the best appearing one during exploration. Whereas it might seem unimportant at first glance, it can be substantially detrimental in tasks where the worst actions lead to very bad results. An approach to counter this disadvantage is to provide a distribution of selection probability according to the value of the actions. The greedy selection would still be assigned the highest selection probability, whereas the selection probability of the remaining actions is weighed according to the actions value.

These action selection methods are called *softmax* methods.[105] Commonly, softmax methods use a Gibbs or Boltzmann distribution and the probability of choosing an action a at t-th play yields:

$$P_a = \frac{e^{Q_t(a)/\tau^*}}{\sum_{b=1}^{n} e^{Q_t(b)/\tau^*}} \qquad \textit{Eq. 2.20}$$

where τ^* is a positive parameter called temperature. High temperatures cause the actions to be nearly equally probable, whereas low temperatures cause a greater difference in selection probability for actions that differ in their value estimates. For $\tau \to 0$ the softmax selection becomes the same as greedy selection.

It is unclear whether ε-greedy or softmax selection is better and no careful comparative studies had been found by SUTTON AND BARTO[106]. The two methods have only one parameter that has to be set, however, setting ε is more intuitive, whereas setting τ requires knowledge of the likely action values. Due to its simplicity and its easier

[105] Though the method was apparently first introduced by Luce 1959, the term softmax is attributed to Bridle 1990.

[106] Sutton / Barto 1998

implementation, all approaches in the context of this work use an ε-greedy action selection method.

2.2.1.4 Model of the environment

The final element of a reinforcement learning system is a model of the environment. This element is meant to reproduce the behavior of the environment. SUTTON AND BARTO[106] consider the use of a model of the environment only as a planning tool, which they use to predict the resultant next state and reward in order to decide on a course of action considering future situations. They consider the fact that the agent could simultaneously learn by trial-and-error, learn a model of the environment, and use the model for future decision making.

Nevertheless, especially in the case of technical tasks, a model of the environment can be used to make predictions about how the physical plant might react to the RL agent. In particular, a policy can be computed in simulation and then applied to the physical task in order to reduce the risk of harming hardware as a result of especially harmful explorative actions. Furthermore, a simulation model of the environment can accelerate development of RL solutions if the computation time is actually shorter than the time step itself.

However, a model of the environment is not necessary and its use is often not even desired, since producing an accurate model might require a considerable amount of effort.

2.2.2 Agent-Environment Interaction Model

Thus far, all elements of the RL framework have been introduced. However, the framework becomes much easier to understand after regarding the actual agent-environment interaction procedure.

As was mentioned earlier, *the agent* is the entity considered the learner and decision maker. It interacts with *the environment*, which comprises everything outside the agent. The interaction takes place continually; the agent taking actions and the environment transitioning into new states and giving rise to rewards in accordance to these actions. The sole purpose of the agent is to maximize these rewards in the long run. A *task* is defined by a complete description of the environment.

Formally, at each time step t, the agent interprets a representation of the environments *state,* $s_t \in S$, where S is the set of possible states, and selects an *action,* $a \in A(s_t)$, where $A(s_t)$ is the set of available actions in s_t. A time step later the agent receives a numeric *reward,* $r_{t+1} \in R$, as a consequence of the selected

action and finds itself in a new *state* s_{t+1}. The agent-environment interaction model in accordance to SUTTON AND BARTO[107] is depicted in Fig. 2.10.

Whereas this framework is conceived to be very specific about the interaction of an agent with its environment, it is remarkably flexible and abstract at the same time. Virtually any task could be conceived to conform with the framework if it is represented properly. Time steps have not necessarily have to be actual time intervals, they could easily be assumed to be any other process describing element such as a play during a game or a turn in a match. Consider a robot designed to play tennis; actions can be as low level as the current applied to the joints of the robot in order to move, as they can be complex and/or abstract such as choosing a forehand over a backhand shot. The same goes for states, they can describe the physical parameters of said robotic joint as well as they could describe a game situation such as a service.

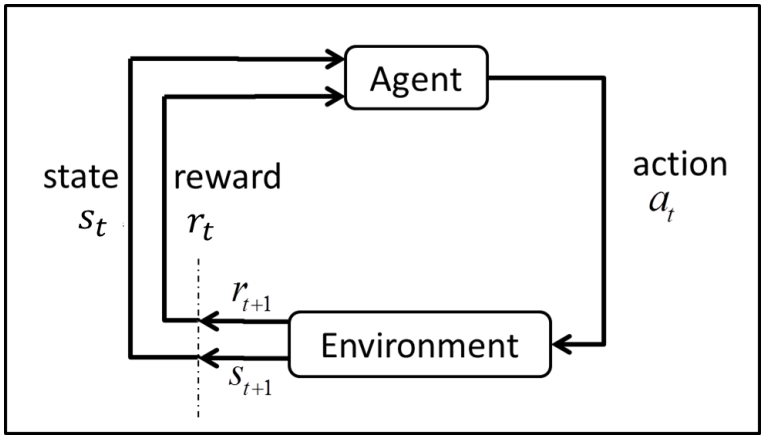

Figure 2.10: The agent-environment interaction framework

Furthermore, the boundaries between an agent and its environment are worth discussing. Generally, the boundaries are generally different from the physical boundaries in a RL task. One might intuitively consider the robot in the tennis playing task the agent, however, its joints and links are actually considered part of the environment, since their behavior does not entirely underlie the agent's control. Therefore, a general rule to define the boundary between an agent and its environment is to consider everything that cannot be changed arbitrarily by the agent a part of its environment.

2.2.3 The Markov Property and Markov Decision Processes (MDP)

The only information available to the agent before making a decision is contained in description of the environments state. It is therefore of great importance that this description accurately describes all the elements of the environment that the agent requires in order to make appropriate decisions.

[107] Sutton / Barto 1998

One important requirement is that the information about any given state includes all the information about its history. Ideally, a state-signal should be able to retain all relevant information successfully. A signal that satisfies this requirement is said to be *Markov*, or to have the *Markov property*.[108,109]

In general, the response of the environment at the time $t+1$ to an action taken at the time t can be formulated in the form of the probability P_r for the expected state and the expected return:

$$Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, a_{t-1}, \dots, r_1, s_0, a_0\} \qquad \textit{Eq. 2.21}$$

for all s', r as well as for all states, actions and returns in the past $s_t, a_t, r_t, s_{t-1}, a_{t-1}, \dots, r_1, s_0, a_0$. However, if the environment or the state-signal that describes it has the Markov property, *Eq. 2.18* can be simplified as follows:

$$Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\} \qquad \textit{Eq. 2.22}$$

Thus, an environment that has the Markov property allows the prediction of any future states and rewards given the current state and action. Since decisions and values are assumed to be a function of the current state only, i.e. of its one step dynamics, the Markov property is of great importance in RL.

An example of a Markov environment, or the Markov description of an environment, is a snapshot of all the figures and the board at any given turn during a game of chess. As long as the position of all the figures is included in the description of the state, all possible outcomes of the game can be predicted regardless of how the game situation came to be, i.e. regardless of the game's history.

SUTTON AND BARTO[110] consider it still appropriate to think of the state in RL as an approximation of a Markov state, even if the signal is in fact non-Markov. However, they insist that RL systems will perform better as the state-signal approaches the ability of Markov states.

A RL task that satisfies the Markov property is called a *Markov decision process* or *MDP*. If the state and action spaces are finite, then it is called a finite MDP. Even though most technical problems are of continuous nature, thus involving theoretically an infinite amount of states and actions, they are usually modelled as finite MDPs, e.g. by discretizing the state and action spaces into a finite number.

---

[108] Markov / Nagorny 1988

[109] The groundwork of MDPs goes as back as Bellman 1957, but they are named after the Russian mathematician Andrey Markov. The theory of MDPs is treated by, e.g., Bertsekas 1995, Ross 1983 and White 1969.

[110] Sutton / Barto 1998

In addition to its state and action sets, a finite MDP is characterized by the one step dynamics of the environment. The transition probabilities for a transition from s to s' after performing the action a yields:

$$P_{ss'}^{a}\{s_{t+1} = s' | s_t = s, a_t = a\} \qquad \textit{Eq. 2.23}$$

Analogously, the expected value of the next reward can be formulated as follows:

$$R_{ss'}^{a}\{r_{t+1} | s_t = s, a_t = a, s_{t+1}\} \qquad \textit{Eq. 2.24}$$

The elementary solutions presented later in this chapter assume implicitly that the environment is a finite MDP.

It is possible to define optimal policies and optimal value functions for finite MDPs.[110] A policy π is defined to be better than or equal to a policy π' if its expected return is greater than or equal to that of π' for all states (or state-action pairs). There can be more than one optimal policy, but they are all denoted as π^* and they all share the same optimal state-value function V^*:

$$V^*(s) = \max_{\pi} V^{\pi}(s) \qquad \textit{Eq. 2.25}$$

They also share the same optimal action-value function Q^*:

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a) \qquad \textit{Eq. 2.26}$$

The relationship between V^* and Q^* can be expressed as follows:

$$Q^*(s, a) = \{r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a\} \qquad \textit{Eq. 2.27}$$

For optimal value and action-value functions, the consistency condition expressed by the Bellman equation can be written in a special form without referencing any specific policy called the *Bellman optimality function*.[111] For optimal value functions it yields:

$$V^*(s) = \max_{a \in A(s)} \sum_{s'} P_{ss'}^{a} [R_{ss'}^{a} + \gamma V^*(s')] \qquad \textit{Eq. 2.28}$$

Analogously, for optimal action-value functions it yields:

$$Q^*(s, a) = \sum_{s'} P_{ss'}^{a} [R_{ss'}^{a} + \gamma \max_{a'} Q^*(s', a')] \qquad \textit{Eq. 2.29}$$

[111] For a detailed derivation of the Bellman optimality equation please refer to either Bellman 1957 or Sutton / Barto 1998

Having $V^*(s)$ or $Q^*(s,a)$ allows it to turn the optimal expected long term reward into a quantity available locally and immediately for each state so that a one-step-ahead search yields the optimal results.

2.2.4 Elementary Solution Methods

SUTTON AND BARTO[112] define reinforcement learning (RL) not by the characterization of the learning methods, but by characterizing the learning problem and considering any method suited to solve such problems a reinforcement learning method. However, they recognize three fundamental methods: *dynamic programming* (DP), *Monte Carlo methods* (MC), and *temporal difference learning* (TD). All of them have particular strengths and weaknesses that often determine their suitability for a certain task. DP methods are well developed mathematically, but a complete and accurate model of the environment is necessary for their implementation. MC methods, on the other hand, require no model of the environment, but are not suitable for step-by-step incremental computation. TD methods combine the advantages of the former methods, i.e. they require no model of the environment and are fully incremental, but their analysis is more complex. At last, it is possible to combine these methods in order to obtain the best features of each of them. In the context of this work, *eligibility traces* are briefly introduced as such a possibility.

2.2.4.1 Dynamic Programming

Dynamic programming is a term that refers to algorithms used to compute optimal policies under the assumption of a perfect model of the environment as a MDP. It is precisely the assumption of a perfect model of the environment and the high computing effort they require that limits their utility. They are, however, of great theoretical importance since they provide an essential foundation for the understanding of the other presented methods. Essentially, these methods can be considered an attempt to reproduce the effect of DP with less computation and without the assumption of a perfect model of the environment (or no model at all).

The main idea behind DP (and RL generally) is the use of value functions to organize and structure the search for good policies. As discussed in the previous subchapter, optimal policies are easily found when the optimal value or action-value function, $V^*(s)$ and $Q^*(s,a)$ respectively, are known.[113] Ultimately, DP algorithms are obtained by turning Bellman equations into update rules for improving approximations of the desired value functions.

---

[112] Sutton / Barto 1998

[113] cp. chapter 2.2.3

The DP process can be reduced to two fundamental, interdependent procedures. The first consists of computing the state-value or action-state value function for a policy. This step is called *policy evaluation*. Once the value function has been computed it is used to find better policies. Thus, the second step is called *policy improvement*. Once the policy has been improved, its value function can be computed and used to improve the policy yet again. This cycle of policy evaluation and policy improvement is known as *policy iteration*.

Policy evaluation

For the policy evaluation, the initial approximation of the value function V_0 is chosen arbitrarily. Every successive approximation is computed using the Bellman equation for V^π as an update rule:

$$V_{k+1}(s) = E_\pi\{r_{t+1} + \gamma V_k(s_{t+1}) | s_t = s\} = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V_k(s')] \qquad \textit{Eq. 2.30}$$

The sequence V_k can be shown to converge to V^π for $k \to \infty$.

Each successive approximation of the value function means iterative policy evaluation is applied to every state, i.e. the value of s is replaced by a new value obtained from the old values of its successor states and the expected immediate rewards. The update is applied to all possible one-step transitions under the policy under evaluation; therefore, this kind of operations is called a *full backup*.

Finally, in order to implement a policy evaluation algorithm it is necessary to introduce a halting condition since iterative policy evaluation would otherwise never stop.

Policy improvement

Once the value function has been computed or approximated, it can be used to find better policies. This can be achieved by deviating from the current policy and selecting an action that is outside of it when in a given state and following the policy again thereafter. The value of this behavior is given by:

$$Q^\pi(s,a) = \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V^\pi(s)\right] \qquad \textit{Eq. 2.31}$$

If this value is greater than $V^\pi(s)$, then it was better to select the new action when in this particular state and thereafter follow the policy all the time. The so called *policy improvement theorem*[114] specifies that, when this is the case, it is better to always

[114] A derivation of the policy improvement theorem is provided by Sutton / Barto 1998. The authors also point out, that even though the presented case is for deterministic policies it can easily be extended to stochastic policies.

choose the new action when in that state and therefore, that the new policy that requires this selection is better than the old one.

1. Initialization

$V(s) \in R$ and $\pi(s) \in A(s)$, arbitrarily for all $s \in S$

2. Policy Evaluation

Repeat

$\Delta \leftarrow 0$

For each $s \in S$:

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V(s')\right]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \beta$ *(β is a small positive number)*

3. Policy Improvement

$policy - stable \leftarrow true$

For each $s \in S$:

$b \leftarrow \pi(s)$

$\pi(s) \leftarrow \arg\max_a \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V(s')\right]$

If $b \neq \pi(s)$, then $policy - stable \leftarrow false$

If $policy - stable$, then stop; else go to 2

Output $V \approx V^{\pi}$

Figure 2.12: Pseudo algorithm of the generalized policy iteration

Generalized policy iteration

Once a policy has been improved with policy improvement, its value function can be evaluated in order to improve it once again. A sequence of monotonically improving policies and value functions is obtained:

$$\pi_0 \xrightarrow{\mathbf{E}} V^{\pi_0} \xrightarrow{\mathbf{I}} \pi_1 \xrightarrow{\mathbf{E}} V^{\pi_1} \xrightarrow{\mathbf{I}} \pi_2 \xrightarrow{\mathbf{E}} \cdots \xrightarrow{\mathbf{I}} \pi^* \xrightarrow{\mathbf{E}} V^*$$

Figure 2.11: Policy iteration. Here, "*E"* stands for a policy evaluation and "*I*" stands for a policy improvement.

A simplified pseudo algorithm for the generalized policy iteration (GPI) procedure can be formulated after SUTTON AND BARTO[115] and is depicted in Fig. 2.12.

[115] Sutton / Barto 1998

A final important characteristic of DP methods is that they update estimates of values of states based on estimates of the successor states. The general idea of estimating on the basis of estimates is called *bootstrapping*.

2.2.4.2 Monte Carlo Methods

On the context of this work, Monte Carlo (MC) methods are only briefly introduced. The fact that value function is updated only at the end of an episode makes it unsuitable for incremental computation. Furthermore, the high variance of the returns has a negative influence on the convergence of MC methods.[116]

MC methods can still be very effective due to the fact that they do not require a model of the environment at all. Instead, they can be used to compute good policies on the base of sample episodes that can be generated quite easily. For example, a set of games of black jack can be used to compute a policy with MC-methods, whereas a DP method would require a comprehensive model of all possible game situations. Furthermore, MC methods do not *bootstrap*, unlike the previously presented DP methods. This makes MC methods less susceptible to violations of the Markov property, since they do not require estimates for the estimation of new values.

The main idea behind MC methods is to estimate the state-value function of a given policy in a given state by simply averaging the returns observed after visiting it. With time, the average should converge to the expected value.

SUTTON AND BARTO[116] differentiate between two main types of MC methods. The first considers only the first visit to a state during an episode for the average of the return for that state. This so called *first-visit MC method* ignores any latter visits to the state during that episode. On the other hand, the *every-visit MC method* averages the returns of all visits to all states during an episode. [116]

In order to find an optimal policy, MC methods rely on a variation of the GPI previously presented for DP methods and described in detail by SUTTON AND BARTO[116].

Analogously to the DP method, SUTTON AND BARTO[116] offer the pseudo algorithm for a first-visit MC method depicted in Fig. 2.13.

[116] Sutton / Barto 1998

1. Initialization

$\pi \leftarrow$ policy to be evaluated

$V \leftarrow$ arbitrary state-value function

$Returns(s) \leftarrow$an empty list, for all $s \in S$

Repeat forever:

(a) Generate an episode using π

(b) For each state s appearing in the episode:

$R \leftarrow$ return following the first occurence of s

Append R to $Returns(s)$

$V(s) \leftarrow average(Returns(s))$

Figure 2.13: Pseudo algorithm for a first-visit MC method[117]

2.2.4.3 Temporal Difference Learning

Temporal difference (TD) learning can be describes as a combination of ideas from DP and MC methods. On the one hand, they can learn from direct experience without a model of the environment and/or exact transition probabilities. On the other, they estimate the value of new estimates on the basis of old estimates, i.e. they bootstrap, and they are suitable for step-by-step incremental computation like DP methods.

In its structure, TD methods are very similar to DP and MC methods. At first, the focus of these three methods lies in estimating the value function V^{π} for a given policy π, often referred to as the *prediction* problem. Afterwards, in order to find a better and eventually an optimal policy, a form of GPI is performed. The main difference between the three lies in the approach used for the prediction problem.

Both MC and TD methods use experience for the prediction problem. The main difference between them is that MC methods wait until the return R_t is known at the end of an episode to update the value function, whereas TD methods need only wait until the reward r_{t+1} is known in the next time step to do so. This becomes apparent when comparing the update rules for an MC method:

$$V(s_t) \leftarrow V(s_t) + \alpha[R_t - V(s_t)] \qquad \textit{Eq. 2.32}$$

and a comparable TD method:

$$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \qquad \textit{Eq. 2.33}$$

In both cases α denotes the *learning rate*, which determines how newly acquired information is weighed against previous estimations. In *Eq. 2.33*, γ denotes the

[117] Sutton / Barto 1998

discount rate introduced in *Eq. 2.14*. These update rules show how the estimation in MC and TD methods is based on single samples rather than a complete distribution of all possible states (or state-action pairs). For this reason they are said to perform a *sample backup*, whereas DP methods perform a *full backup*.

The core of most RL algorithms is the update rule for the estimation of new values. The update rule for the simplest of TD methods, the TD(0) method, is given by *Eq. 2.33*. SUTTON AND BARTO[118] offer the following pseudo algorithm for the TD(0) estimation of the value function:

Initialize V_s arbitrarily,π policy to be evaluated

Repeat (for each episode)

Repeat

$a \leftarrow$ action given by π for s

$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$

$s \leftarrow s'$

until s is terminal

Figure 2.14: Pseudo algorithm for the TD(0) estimation of V_s

Once the prediction problem has been dealt with, TD is used to find an optimal policy. This second part of the procedure is often referred to as the *control* problem. There are two main approaches to implement a TD method: *on-policy* and *off-policy*. A third type of approaches called *actor-critique methods* is an extension of the former two but was not considered in the context of this work.

SARSA: On-Policy TD control

The on-policy implementation of TD methods is commonly known as SARSA[119]. The name is an acronym derived from its update rule, since it involves the current **s**tate and **a**ction, and the **r**eward received for transitioning into the next **s**tate and selecting one of the next available **a**ctions. It is called on-policy because the policy in use is also the one being updated. The changes to the policy after an agent's interaction with the environment immediately influence future decisions.

On- and off-policy TD control methods aim at the estimation of an action-value function and consequent optimization of a policy, i.e. estimate $Q^{\pi}(s,a)$ for all states and actions and afterwards improve it until $Q^{\pi}(s,a) \approx Q^{*}(s,a)$. SUTTON AND BARTO[120] provide the

[118] Sutton / Barto 1998
[119] Rummery / Niranjan 1994
[120] Sutton / Barto 1998

general form of a SARSA-algorithm in the form of the pseudo algorithm depicted in Fig. 2.15.

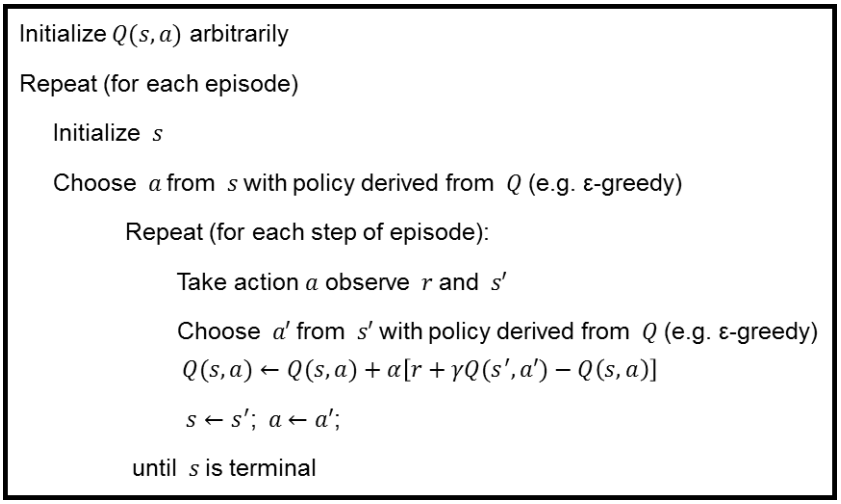

Figure 2.15: Pseudo algorithm for on-policy TD-control[120]

The convergence of SARSA is considered to be guaranteed in the limit for policies like ε-greedy as long as ε decreases with time. SINGH ET AL.[121] provide the convergence results of on-policy algorithms.

Q-learning: Off-policy TD control

The off-policy control algorithm known as Q-learning[122] is considered one of the most important breakthroughs in RL. The main difference to the SARSA algorithm is that in this case the learned value function directly approximates the optimal value function independently of the policy being followed, hence off-policy. Not only does this simplify the analysis of the algorithm but also enables early convergence proofs. However, off-policy algorithms are often found to underperform on-policy methods.[123]

SUTTON AND BARTO[120] provide the pseudo algorithm for off-policy TD control depicted in Fig. 2.16.

[121] Singh et al. 2000
[122] Watkins, C. J. C. H. 1989
[123] Frietsch 2011

Initialize $Q(s,a)$ arbitrarily

Repeat (for each episode)

Initialize s

Repeat (for each step of episode):

Take action a observe r and s'

Choose a' from s' with policy derived from Q (e.g. ε-greedy)

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{s'} Q(s',a') - Q(s,a)\right]$$

$$s \leftarrow s';\ a \leftarrow a'$$

until s is terminal

Figure 2.16: Pseudo algorithm for off-policy TD control[124]

2.2.4.4 Eligibility Traces: TD(λ)-algorithm

Finally, the possibility of extending the one-step TD algorithm, the TD(0), to consider a more farsighted approach is achieved through the implementation of eligibility traces.

Generally, eligibility traces can be understood as a bridge between the TD(0) algorithm at one end and an MC algorithm at the other. A more mechanic view of them would be to describe them as a temporary record of states visited and actions taken. The first option is considered the *forward view*, whilst the second is known as the *backward view*. However, they both describe one and the same method.[124]

The forward view requires that the agent decides how to update each state by looking forward to future rewards and states and is, thus, not practical. The backward view, on the other hand, can be regarded as an incremental mechanism for approximating the forward view.[124]

In the backward view, an additional memory variable for every state, the so called *eligibility trace* (e-trace), is used to determine which states are "eligible" for being updated as a result of an interaction of the agent with the environment. The traces do not have a constant value; they are incremented by 1 when the state they are associated with is visited and decay with time by the constant value $\gamma\lambda$. The value of the e-trace $e_t(s)$ at the time t for the state s is given by the following definition:

$$e_{t(s)} = \begin{cases} \gamma\lambda e_{t-1} + 1 & if \ \ s = s_t \\ \gamma\lambda e_{t-1} & if \ \ s \neq s_t \end{cases} \qquad \textit{Eq. 2.34}$$

whereas γ is again a decay parameter. The trace decay factor, $0 \leq \lambda \leq 1$, determines how farsighted the TD(λ) algorithm is, since it determines how strongly the value of a trace diminishes after a visit to its state. For $\lambda = 0$, the algorithm becomes the one-step TD algorithm, thus the TD(0) algorithm. For $\lambda = 1$ the algorithm becomes the

[124] Sutton / Barto 1998

corresponding MC algorithm. Analogously, the value of $e_t(s,a)$ for action-value functions yields:

$$e_{t(s)} = \begin{cases} \gamma\lambda e_{t-1} + 1 & if \;\; s = s_t, a = a_t \\ \gamma\lambda e_{t-1} & otherwise \end{cases} \qquad \textit{Eq. 2.35}$$

The value of an e-trace depends on the number of visits to its state (or state-action pair), as depicted in Fig. 2.17.

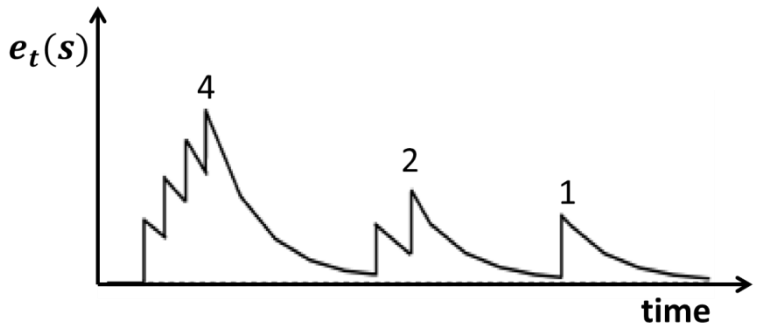

Figure 2.17: Value of e-traces as a function of time and the amount of state visits[125]

Despite the advantages of a more farsighted algorithm and eventually faster learning, eligibility traces require significantly more computation than one step methods such as TD(0). Furthermore, SUTTON AND BARTO[125] point out that implementation of a relatively simple TD(λ) approach is a challenge to implementations on conventional serial computers and might require the use of single-instruction, parallel computers or neural systems. However, they also point out that the value of most of the eligibility traces at a given time is almost always nearly zero for conventional values of γ and λ. Despite this fact, the implementation of eligibility traces still poses a challenge to online implementations due to the restriction on real time communication and data processing.

Further aspects on practical issues regarding TD(λ) algorithms[126] and their application on a game of backgammon (TD-gammon)[127] are offered by TESAURO.

2.2.5 Applications of RL and State of the Art Research

In this section, a selection of RL applications in a diversity of fields of study is presented. Subsequently, an overview of state of the art research in RL algorithms is provided.

[125] Sutton / Barto 1998
[126] Tesauro 1992
[127] Tesauro 1995

2.2.5.1 Applications of Reinforcement Learning Algorithms

The RL framework is of a very flexible character, which in turn allows it to be used as an approach for very diverse tasks, as long as they can be formulated as MDPs and the Markov property can be assumed for the environment. Even if the property is infringed, there are several methods and approaches in order to apply RL algorithms successfully. ALBERS, FRIETSCH and SOMMER OBANDO among others researched the application of RL approaches for control purposes in the context of the development of humanoid robots such as ARMAR. The range of applications was extended to include other optimization tasks, such as the computation of fuel efficient operation strategies for automatic vehicles and eventually the subject of this thesis, which focuses on the reduction of clutch judder through a RL based controller. In this section, these and a selection of other RL applications in diverse fields of study is presented. A more detailed overview is offered by FRIETSCH[128].

In the field of general industrial applications, CREIGHTON AND NAHAVANDI[129] used a RL algorithm to improve the performance in a melt facility. The continuous problem was modelled as a Semi Markov Decision Process (SMDP). The agent was able to improve the operation strategy in use, which was based on expert knowledge, by 14% and was more robust towards random disturbances.

HONG AND PRABHU[130] present an approach that is suitable for Just-In-Time (JIT) production for multi-objective scheduling problems in a dynamically changing shop floor environment. The machine control problem is also modeled as SMDP and solved using Q-learning. According to the authors, results show that distributed learning and control DLC algorithms achieve significant performance improvement over usual dispatching rules in complex real-time shop floor control problems for JIT production.

DALAMAGKIDIS ET AL.[131] use a RL agent to deal with the issue of achieving comfort in buildings with minimal energy consumption. The environment state-signal is provided in the form of the thermal comfort of the building occupants, the indoor air quality and the energy consumption. The controller is then compared with a traditional on/off controller, as well as a Fuzzy-PD controller. The results show that after some simulated years of training, the reinforcement learning controller has equivalent or better performance when compared to the other controllers.

In automotive science, HOWELL ET AL.[132] propose a RL algorithm for the control of a semi-active suspension system on a road going, four wheeled, passenger vehicle. The

[128] Frietsch 2011
[129] Creighton; Nahavandi 2005
[130] Hong / Prabhu 2004
[131] Dalamagkidis et al. 2007
[132] Howell et al. 1997

algorithm is used to minimize the mean square acceleration of the vehicle body and thus improve its ride isolation qualities. The learning algorithm operates over a bounded continuous action set and the authors consider it robust to high levels of noise and ideally suited to operating in a parallel computing environment.

The application of RL algorithms is also found in the field of medical research. FAKIH[133] applied a RL agent to the task of determining the most efficient use of diagnostic tests on different patients. The algorithm is implemented on a sample problem of diagnosing Solitary Pulmonary Nodule. The author considers the RL based methodology holds significant promise in improving the performance of diagnostic process.

MAGALHAES BARROS NETTO, S. ET AL.[134] use an application of RL for diagnosis based on medical image purposes. Specifically, application of an off-policy reinforcement learning algorithm, Q-Learning, is used to solve the problem of lung nodules classification by analyzing the 3D geometric nodules characteristics to guide their classification. The authors consider their results encouraging, stating that using characteristics of the nodules' geometry can effectively classify benign from malignant lung nodules based on computer tomography CT images.

GUEZ ET AL.[135] and BUSH AND PINEAU[136] offer an approach using RL for the neurostimulation treatment of epilepsy. The agent chooses which stimulation action to apply, as a function of the observed Electroencephalography signal (EEG), so as to minimize the frequency and duration of epileptic seizures using labeled training data of animal tissue. The results show the application to be an effective way of reducing the incidence of seizures, while also minimizing the amount of stimulation applied. The application of the algorithm on a model of human epilepsy is pending.

Furthermore, RL applications have been used for the optimization of traffic control. BALAKRISHNA ET AL.[137] describe a method for estimating average taxi-out times at the airport in 15 minute intervals of the day and at least 15 minutes in advance of aircraft scheduled gate push-back time. The RL algorithm is trained and tested using historic data from the Federal Aviation Administration's (FAA) Aviation System Performance Metrics (ASPM) database. The algorithm was tested on John F. Kennedy International airport (JFK), whereas the predicted average taxi-out times matched the actual average taxi-out times within ±5 minutes for about 65 % of the time (for the period

[133] Fakih 2004
[134] Magalhaes Barros Netto, S. et al. 2008
[135] Guez et al. 2008
[136] Bush / Pineau 2010
[137] Balakrishna et al. 2010

before 4:00 P.M) and 53 % of the time (for the period after 4:00 P.M) on average across 15 days.

DESJARDINS AND CHAIB-DRAA[138] investigate cooperative adaptive cruise control CACC by proposing a novel approach for the design of autonomous vehicle controllers based on RL. Specifically, the agent is used to develop controllers for the secure longitudinal following of a front vehicle. According to the authors, the experimental results show the approach to be promising.

ALBERS ET AL.[139] propose an on-policy TD(0) algorithm, a SARSA algorithm, for the purpose of computing fuel efficient vehicle operation strategies. The RL-algorithm is used to compute optimal gear and gas pedal trajectories in order to minimize the fuel consumption during a driving maneuver consisting of “acceleration” and “cruise” phases on a simulated vehicle and environment. The results were compared to those achieved with a common reference optimization process and validated with the commercial simulation tool AVL-cruise. The RL-algorithm was able to outperform the reference optimization both in computation time and resulting fuel consumption during the maneuver.

Finally, a considerable number of RL applications are also found in the field of robotics. MORIMOTO AND DOYA[140] consider a RL for the standup task of a three-linked planar robot. Particularly, the focus of the work is the application of TD-methods to solve the task. Later, MORIMOTO ET AL.[141] applied a model based RL algorithm for the task of robotic biped walking. The algorithm is used to appropriately modulate an observed walking pattern. It modulates via-points from observed walking trajectories using the minimum jerk criterion to improve the walking trajectories. The approach is applied to a simulated and an actual biped robot.

KORMUSHEV ET AL.[142] used a RL-based approach for the minimization of the electric energy consumption during walking of a passively-compliant bipedal robot. This is achieved by learning a varying-height center-of-mass trajectory which uses efficiently the robot’s passive compliance. Also, the RL algorithm evolves the policy parameterization dynamically during the learning process and thus manages to find better policies faster than by using fixed parameterization. The algorithm is applied to

[138] Desjardins / Chaib-draa 2011
[139] Albers et al. 2011d
[140] Morimoto / Doya 1998a
[141] Morimoto et al. 2005
[142] Kormushev et al. 2011

the humanoid robot COMAN[143], where the authors confirm a significant reduction in energy consumption.

KOHL AND STONE[144] achieved a similar improvement of the walking pattern of a quadrupedal robot as MORIMOTO ET AL. did for the bipedal robot. KOHL AND STONE present a RL algorithm for optimizing a quadrupedal trot gait for forward speed. The algorithm searches for the best possible set of parameters to increase the forward velocity of the commercially available robot Aibo[145]. After approx. three hours of learning on the physical robot, the achieved gait was faster than any previously known gait for the Aibo, significantly outperforming a variety of existing hand-coded and learned solutions.

Other applications of RL algorithms can be found in the fields of economics and finance[146], computer science[147] or entertainment[148] amongst other.

2.2.5.2 State of the Art Research

Although there is no real consensus about the structure of the research field, this section leans on the work by BARTO AND MAHADEVAN[149] and FRIETSCH[150]. According to their work, there are two fundamental aspects that are the focus of recent research: the *optimization of the learning process* and the development of *generalizing approaches*. However, FRIETSCH[150] points out that it is often difficult to categorize approaches into strictly one of these two aspects.

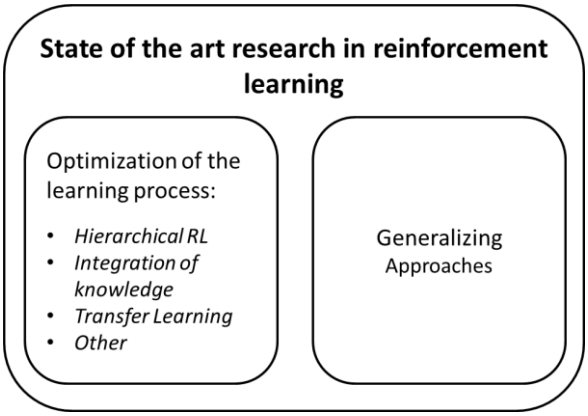

Figure 2.18: Overview of state of the art research field in RL according to FRIETSCH[150]

[143] Istituto Italiano di Tecnologia (IIT) 2014
[144] Kohl / Stone 2004
[145] Sony 2014
[146] Dempster / Leemans 2006
[147] Zhang / Huang 2006
[148] Andersen et al. 2009
[149] Barto / Mahadevan 2003
[150] Frietsch 2011

Optimization of the learning process

Most implementations of RL algorithms have been proven to converge towards an optimal policy mathematically, however, they normally do so in the limit, i.e. after every action in every state has been taken an infinite number of times. This is evidently unpractical and is not adequate to compare different algorithms. For this reason, practical implementations of RL learning consider the learning process finished when the changes to the value or action-value function are below a low predefined threshold or a time limit is reached.[151] Furthermore, different benchmark environments and tasks exist, in order to compare different solutions such as the “mountain car task”[152], the “Acrobot”[153] or the “pole balancing task”[154].

The optimization of the learning process targets both the *convergence* of a policy as well as its *quality* after convergence, i.e. how good is the best policy the algorithm can find and how fast is it found. This is of particular importance in high dimensional problem due to the previously mentioned curse of dimensionality.

The most prominent method to optimize the learning process in high dimensional problems is the implementation of *hierarchical learning*. A complex problem is analyzed and decomposed into subproblems in line with the maxim of “divide and conquer”.[155] Accordingly, a normal MDP is specified to be a “*flat”* or *“monolithical”* MDP. [156]In the context of this work, two examples of hierarchical reinforcement learning implementations are presented. For further examples refer e.g. to FRIETSCH[157]. First, BAKKER ET AL.[156] implement a hierarchical RL algorithm for the task of robot path planning. In their work, the authors use maps with different grades of resolution and assign them a hierarchy. The implementation consists of a DP algorithm that the authors claim is more efficient than standard DP for "flat" MDPs, because it reduces the state space for all levels in its hierarchy and it allows reuse of previously computed partial policies. However, they also point out that computational advantage comes at the cost of some extra memory and overhead to represent and coordinate the hierarchical system and in some cases somewhat longer paths to target locations. The algorithm is tested on artificially generated MDP data, and on real robot data from a vision-controlled robot in an office environment. Similarly, MORIMOTO AND DOYA[158] propose a hierarchical reinforcement learning architecture on a three-link, two-joint

[151] Sutton / Barto 1998
[152] Moore 1991
[153] Yoshimoto et al. 1999
[154] Behsaz; Safabakhsh 2007
[155] Jonsson / Barto 2005 and Jonsson 2008
[156] Bakker et al. 2005
[157] Frietsch 2011
[158] Morimoto / Doya 2001b

robot for the task of learning to stand up. The authors introduce a low-dimensional representation of the state of the robot for higher-level planning, whereas the upper level learns a discrete sequence of sub-goals in a low-dimensional state space for achieving the main goal of the task. The upper-level learning was implemented by Q-learning, whereas the lower-level learning was implemented by a continuous actor–critic method. The robot successfully learned to stand up within 750 trials in simulation. The real hardware was able to complete the task after an additional 170 trials.

A further principle to optimize the learning process consists of the *integration of previous knowledge*. The use of this knowledge can be used to boost the speed with which good policies can be found in large state spaces, where the agent might seldom experience a favorable reward. The knowledge might be found in the form of predetermined behavior patterns and/or rules, as implicit knowledge defined by the programmer or as predetermined structures.[159] Also, a distinction is made about whether the knowledge is a result of past learning or introduced by the programmer. An example of the principle of integration of previous knowledge is known as reward shaping and consists of a reward-signal designed with a higher amount of information than is usual, thus allowing the agent to experience more about its environment from a single interaction. PERKINS AND HAYES[160] and COLOMBETTI ET AL.[161] consider reward shaping to be an optimal combination of classic engineering with its use of models and formulae and autonomous learning with its inherent flexibility and unplanned behavior. For a selection of implementations of reward shaping refer to FRIETSCH[159].

The last approach to the optimization of the learning process presented in this work consists of the transfer of information resulting from previous (similar) tasks, also known as *transfer learning (TL)*. TAYLOR[162] offers an overview on TL and its application. The author also differentiates between the following characteristics of learning tasks for transfer:

- Step response to performing an action
- Design of the reward function
- Initial state and terminal state or goal
- Design of the state space (continuous, discrete, amount of state variables, …)
- Design of the action space (continuous, discrete, range, …)

[159] Frietsch 2011
[160] Perkins / Hayes 1996
[161] Colombetti et al. 1996
[162] Taylor 2009

An example of TL consists of the initialization of the value function with values of previous learning instead of an initialization with zero or random values. The values can be a result of previous, similar tasks or the result of simulation data or a guess by the programmer. SOMMER OBANDO ET AL.[163] propose the use of averaged Q-Tables for the application on a two-linked robot arm. The initialization with Q-Tables averaged from earlier tasks can be used to boost the learning speed for new tasks; in fact, a lot of new tasks can be completed without the need for new learning since it also has a generalizing effect on the task[164]. Later, in ALBERS ET AL.[165] the averaged Q-Tables are computed in simulation and applied to a physical robot. The learning speed and the number of failed episodes are reduced considerably.

Generalizing approaches

Finally, a major scope of research in RL consists of the generalization of different tasks that can be clustered to a certain type. LAUER AND RIEDMILLER[166] describe the internal generalizing of tasks on the example of a football playing robot. They state that even though there are virtually an infinite number of different dribbling moves the robot could perform differently, the learning effort can be reduced if a general concept of dribbling as a move is used and deployed in different situations. This is akin to the idea of PETERS[167], who relies on learning to perform elemental tasks as “building blocks” of movement generation. Applied to the football example, an example would be the robot performing the action sequence “locate and get the ball – dribble – shoot”, instead of having to learn all the required movement for all of its limbs individually. PETERS also states that the future steps in motor skill learning in robotics are the collection of skills into libraries, the learning of appropriate skill selection, as well as the sequencing and parallelization of the motor primitives. All of these steps require a high degree of generalization of tasks. A selection of examples of generalizing approaches in RL is presented subsequently. For a more extensive selection of applications refer to FRIETSCH[168], whereas a deeper look into the application of generalizing approaches in robotics is provided by the work of Jan Peters[169].

The concept of applying function approximation methods for the approximation of the value function in RL is also widely popular. An application of function approximation is

[163] Sommer Obando et al. 2010

[164] The generalizing effect is achieved through the implementation of both the averaged Q-tables and a relative approach.

[165] Albers et al. 2011c

[166] Lauer / Riedmiller 2007

[167] Peters 2008

[168] Frietsch 2011

[169] Some examples include Peters et al. 2003a, Peters et al. 2003b and Peters / Schaal 2006

found in this work and the necessary fundamentals according to ZELL[170] and BUSONIU ET AL.[171] are provided in 6.3.3.

2.3 X-in-the-Loop Framework (XiL)

The XiL-Framework is based on long-term research at IPEK – Institute of Product Engineering dating back to 1996 and integrates consequently simulation and experiment in the product engineering process. It is considered a framework suitable for the continuous, holistic, customer oriented synthesis and validation of modern vehicles with complex functionality with respect to environmental interactions.[172] Therefore, the physical test bench employed in this work was conceived in accordance with the XiL-framework. In this section, the placement of the framework in the engineering process, as well as a description of its elements, is presented.

2.3.1 The Product Engineering Process

According to ALBERS[173], the product engineering process can be described as a continuous interaction of three systems: the operation system, the system of objectives and the system of objects, also called the system triple of product engineering. Furthermore, ALBERS[174] also states that discussing human knowledge and process aspects of the system triple approach reveals the necessity of specifying the role of the operation system within the co-evolutionary and iterative process of complex product engineering. Therefore, the Advanced System Triple Approach[175] (c.f. Fig. 2.19) is introduced in order to describe the two central activities of product engineering: the combination of analyzing objectives and synthesizing objects (creation) and the combination of analyzing objects and synthesizing objectives (validation).

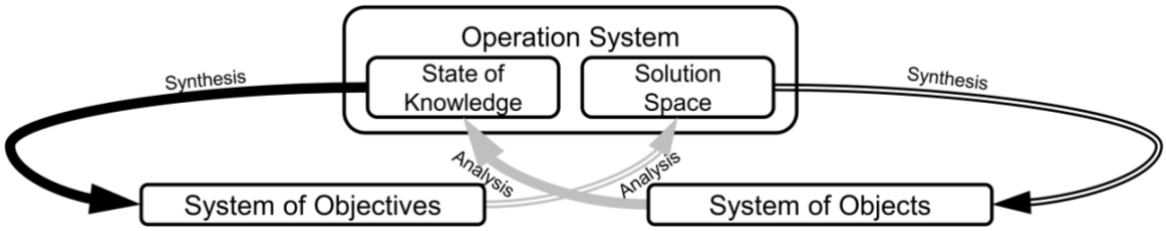

Figure 2.19: Advanced System Triple Approach (from ALBERS ET AL.[176])

[170] Zell 1997
[171] Busoniu et al. 2010
[172] Albers et al. 2008, Düser 2010 and Albers / Düser 2011
[173] Albers 2010
[174] Albers et al. 2010
[175] Albers et al. 2012b
[176] Albers et al. 2013e

ALBERS[177] proposes the integrated product engineering Model – iPeM as a suitable meta-model of the product engineering process mainly because of its high flexibility and generality. The meta-model is divided into “Activities of Product Engineering” (also called macro-activities), the “Activities of Problem Solving” (also called micro-activities), the “Phase Model”, the “System of Resources” and the system triple of objects, objectives and the operation system. Within the iPeM, the balance between the system of objectives and the system of objects is achieved through validation, which is considered the central macro-activity in the product engineering process. This balance, thus validation, is of great importance for the success of the engineering process as only thereby can knowledge be generated, as opposed to the simple comparison between objects and objectives. This knowledge is used to solidify and expand the system of objectives and allows a successful synthesis in case of a goal-oriented return of knowledge in other activities. A graphical representation of the iPeM is depicted in Fig. 2.20.

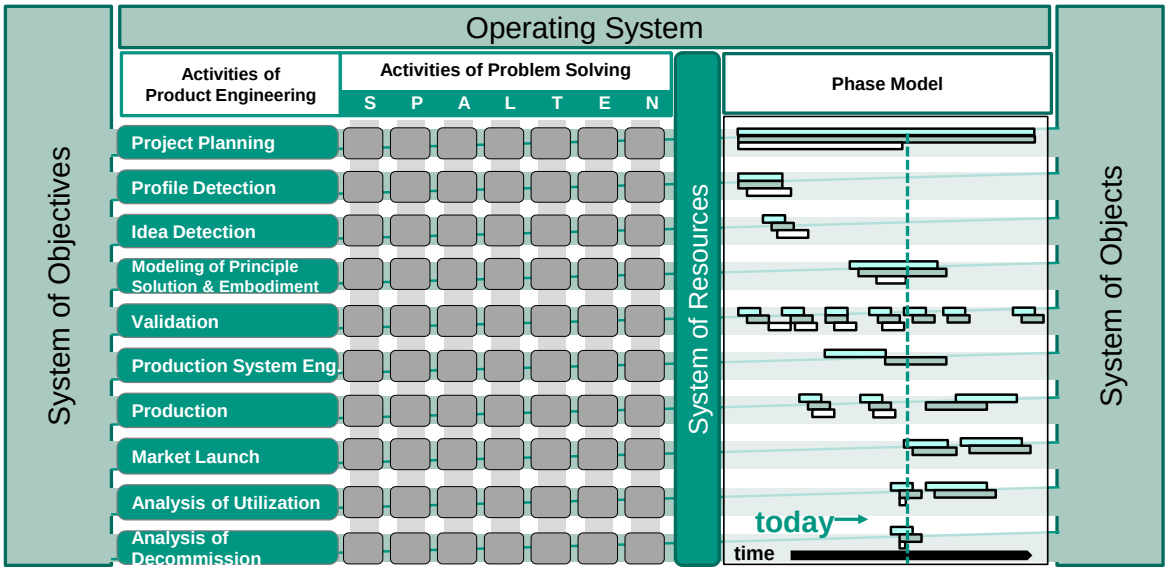

Figure 2.20: Integrated Product Engineering Model – iPeM (ALBERS[177])

2.3.2 XiL-Framework in the Context of Product Engineering

Within the development process, the application of the XiL-Framework mainly addresses the interaction of the activities “Modeling of principle solution & embodiment” and “Validation”. Furthermore, the activity „Project planning” allows an application-related realization of the framework.[176] The framework represents a continuously useable and process applicable approach as part of the “Operation

[177] Albers 2010

System" of product engineering, which allows the analyzing of the three interacting systems "Driver", "Vehicle" and "Environment" with a changing focus.[178]Therefore, the XiL-Framework constitutes a holistic and integrated development and validation framework for powertrain systems. The "X" represents the "System Under Development (SUD)" and can be located in any of the systems "Driver", "Vehicle" or "Environment".[179] The SUD can be of a physical, virtual or a combined physical/virtual nature and can be conceived from a very abstract or highly concrete manner.[180] For this reason, different layers in the XiL-Framework are introduced.

For the system "vehicle" these layers are the element-in-the-loop-layer, the subsystem and the vehicle-in-the-loop-layer (c.f. Fig. 2.21). On each layer the behavior of the rest vehicle is reproduced, either virtually or physically, and connected to the "driver" and the "environment" systems in order to reproduce their effects on the SUD as closely to reality as possible.

The "Driver" and "Environment" systems and the rest vehicle simulation can also be implemented on different scaling levels, which have to be determined according to the application at hand.

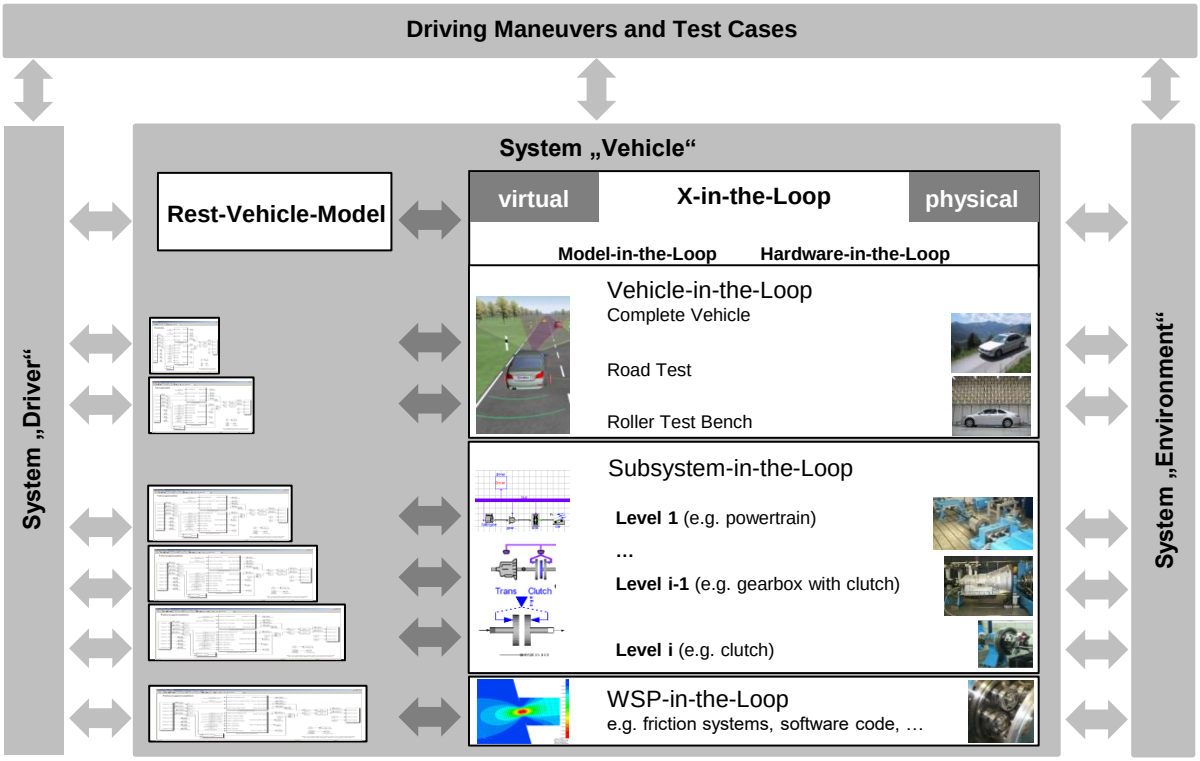

Figure 2.21: IPEK X-in-the-Loop-Framework[181]

[178] Albers et al. 2012a

[179] In the context of this work the SUD is considered to be in the system "Vehicle". Albers et al. 2013e consider the cases where the SUD is located in any of the remaining systems.

[180] Albers et al. 2014

[181] Albers et al. 2008, Albers et al. 2012a and Albers et al. 2013e

3 Motivation and Research Objectives

In this chapter, the motivation for a RL based active damping of clutch judder vibrations and the definition of the research objectives of this work are presented. First, the research gap regarding the suppression of clutch judder is derived from the disadvantages and shortcomings of state of the art in the field and the suitability of a RL based solution to close this gap is evaluated.

3.1 Motivation

The presence of clutch judder vibrations in automotive drive trains has a negative effect on the durability of its components, however, it most importantly constitutes a major detraction on the perceived comfort of the vehicle by the driver regarding noise, vibration and harshness (NVH), which plays a determinant role in regards of his purchase decision. For this reason, efficient countermeasures are ever sought after. In summary, countermeasures against clutch judder can be categorized as either of constructive, tribological or mechatronic nature. Even though most of them can prove to effectively reduce clutch judder, they also entail disadvantageous characteristics.

Constructive countermeasures[182] can address each of the different forms and sources of judder vibrations separately or as a whole. They are realized by both stricter tolerances on components and their assembly, the inclusion of additional vibration-damping components, or by increasing the damping in the drive train or its components. However, stricter tolerances cause an increase in production and assembly costs, whereas additional components inflict space and weight constraints and generate additional production, material, and assembly costs. Lastly, an increase in the damping of the drive train or its individual components generally causes a decrease in energy efficiency because of higher dissipation.

Tribological countermeasures[183] aim to mitigate clutch judder vibrations by means of the friction characteristics of the material parings in the clutch. A negative friction coefficient gradient has an exciting effect on self-induced vibrations, thus materials with such a gradient are generally avoided. However, high performance materials like engineering ceramics, which make for smaller and lighter clutch systems, possess such a gradient. Therefore, their use would be very desirable if the previously mentioned negative effect could be offset. On the other hand, a positive friction coefficient gradient has a dampening effect on drive train vibrations; however, no

[182] c.f. 2.1.4.1
[183] c.f. 2.1.4.2

pairing materials with a positive friction coefficient gradient over the whole relevant temperature range have been developed yet.

Finally, mechatronic solutions[184] aim primarily at the active damping of judder vibrations by means of a control of the clamping force between clutch disks. These measures present an effective alternative provided an electromechanical actuation of the clutch is available and would allow the use of highly efficient materials as friction pairings, which represents an immense untapped potential. The main drawback of these measures is the relative complexity of the solutions, most of the time involving detailed and accurate models of the drive train and its environment and their dynamics during life-time, and the high dynamic demands on the electromechanical actuators, which at the present time might prove to involve high costs. Furthermore, relatively advanced and abstract mathematical operations are often involved for the controller design, whereas its tuning requires additional empirical or optimization effort. Furthermore, the necessary models might prove too specific, so that the successful implementation of a mechatronic countermeasure is only guaranteed to a given individual system (test bench, drive train, vehicle, etc.), unless further adaption efforts are undertaken.

At this point, the need for a mechatronic countermeasure that relies as little as possible on the accuracy and detail of the models it needs and that is able to adapt to changes of these without compromising its effectiveness becomes clear. A possibility to achieve such a countermeasure is the application of a RL control algorithm for the active damping of clutch judder vibrations. RL methods (specifically TD and MC methods) do not require a model of the environment and can learn and optimize a control strategy autonomously.

3.2 Research Objectives

The main research objective of this work is the development and analysis of a RL clamping force control algorithm for the active damping of clutch judder vibrations. In the scope of this work, its implementation on a reduced physical drive train test bench is considered appropriate, since the behavior of a vehicle can be reproduced accurately enough without having to employ prototypes and performing costly experiments. The algorithm should effectively reduce clutch judder without compromising the function of the system or otherwise negatively affecting the comfort perception of passengers. Therefore a successfully implemented RL algorithm should fulfill the following requirements:

- Effective reduction of clutch judder

[184] c.f. 2.1.5

The selected method for quantifying judder vibrations in this work is based on the consideration of the amplitude difference between the gearbox entry shaft rotational speed and the gearbox output.[185] Since the amplitude of the latter is always lower, the amplitude of the gearbox entry shaft rotational speed has to be reduced effectively. Furthermore, the function of the clutch should not be compromised. Therefore, the flux of torque is to be closed within a reasonable time limit. This way, the most important factors regarding clutch comfort during engagement according to DOLCINI ET AL.[186], which are the total duration of the engagement and the amplitude of drive train oscillations, are addressed. However, these factors are physically interdependent, since shorter engagements imply a higher torque transmission by the clutch.

- High learning speed

  Even though in the context of this work it is not clear at what precise point learning will take place in the life time of the drive train, it is safe to assume that a short learning time (fewer episodes), in which comfort might be harmed due to explorative behavior, is to be preferred. The RL algorithm should therefore aim to learn an effective strategy in as few episodes as possible. As a reference value, LuK GmbH & Co. KG[187] consider a minimum of 2000 hours or $60\,000$ km to $120\,000$ km of operation acceptable for a passenger car. The learning time should be substantially shorter.

- Balance between learning time and effectiveness of judder reduction

  A tradeoff between the previous two requirements exists, since accurate, high quality solutions are linked with longer learning time and high computing effort, whereas fast learning solutions usually rely on some form of simplification or generalization that causes the quality of the solution to decrease. A solution that offers a favorable compromise between learning speed and effectiveness is to be favored.

- Model free computation of the control strategy

  The computation of the control strategy should not rely on simulation models other than to reproduce the response of the physical environment. The use of models of some kind is indispensable in this work, since the physical test bench itself is a model of the drive train where some components are conceived

[185] For the concrete formulation of the quantification method c.f. 5.4
[186] Dolcini et al. 2010
[187] LuK GmbH & Co. KG 1982

virtually, whereas others are conceived physically in accordance to the XiL-Framework. However, models in this work are limited to the description of the environment as a “black box” to the agent.[188] Any developed method should be applicable to any given physical or virtual “black box” of the agent’s environment.

[188] The use and the computation of a model of the environment are not entirely restricted in RL. Amongst others, Sutton / Barto 1998 present methods to compute models of the environment with RL algorithms and then use them for the purpose of planning.

4 Research Design

In order to achieve the objectives presented in the previous chapter, a combination of implementations of a RL framework on two different simulated and one physical environment is proposed.

Initially, the formulation of the active damping control task as a RL problem is addressed in a preliminary stage. The elements of the RL framework are adapted to the clutch judder reduction task in order to present it as an MDP. This includes determining an appropriate action-state space representation, a suitable reward-signal and the representation of the agent's environment, the drive train.

In this work different RL based approaches are presented, which have some elements in common, but possess unique features that determine their suitability for the task of suppressing clutch judder. Even though a simulation model of the environment is not necessary for the realization of a RL based controller, it serves the purpose of reducing the development effort and makes it possible to produce significant results without the need of costly prototypes. The structure of this work considers one theoretical stage and two experimental stages. The latter are derived from different conceptions of the environment of the RL framework, i.e. different models of the automotive drive train. These models were conceived in a way that allows the study of the application of the RL algorithms from a relatively simple and abstract representation of the drive train to a test bench of the reduced drive train, which itself is a physical model of the drivetrain.

The theoretical stage aims at the representation of the clutch judder task as a RL problem. All elements of the RL framework are applied to the clutch reduction task and then defined for the latter stages of this work. All propositions and definitions within this chapter are of a highly abstract nature and its elements are often later adapted in order to suit the requirements of the following experimental stages. However, the abstract RL framework conceived in this stage serves as a reference throughout this work. First, the definition of the environment and the task are provided. Afterwards, a definition of the state-signal on whose basis the agent perceives the environment is proposed. Analogously, the action-signal is defined subsequently. Then, the reward-signal that will be used by the agent to evaluate the quality of its actions and which it will strive to maximize is defined. Finally, the characteristics and elements of the agent itself are presented.

In the second stage, the first of the experimental stages, a basic, highly abstract simulation model of the drive train is implemented as the framework environment. This stage can be regarded as a preliminary study of the RL based control of the clamping force in order to reduce clutch judder. The main focus in this stage lies in assessing

the general suitability of the proposed RL framework as a means to successfully perform the judder reduction task. Initially, a classic RL agent (SARSA) learns an adequate policy in a strictly discrete state space with the smoothest conceivable mesh. The high smoothness of the discretization and the strictly discrete nature of the action-state space of state variables deliver the best results at the highest computing effort and set a benchmark value for the rest of the approaches in this stage. These approaches aim at reducing the learning speed whilst retaining as much of the effectiveness of judder suppression. In order to do so, different action and state space representations and approaches to the update of the value function are proposed. First, an adjusted action space is proposed in order to reduce the size of the action space and therefore computing effort. Afterwards, different approaches for the conception of the state space and the update of the value function are proposed. The first consists of a novel conception of a dynamic state space, in which an initially roughly discretized state space is locally refined in order to accommodate the computing effort more efficiently. In the second approach, a custom application of radial basis functions (RBFs) for the purpose of the approximation of the value function in a semi-continuous state space is introduced. Finally, a novel approach involving a Gaussian filter for the approximation of the value function is proposed.

The second experimental stage aims at the application of a RL algorithm for the suppression of clutch judder on the scaled down IPEK Mini Hardware-in-the-Loop test bench. After introducing the test bench itself, an appropriate simulation model to reproduce its behavior is proposed. This simulation model includes new features that offer a detailed description of its physical counterpart, such as a differentiation between closed- and slipping-clutch modes, as well as the consideration of dead times in the transmission of signals, e.g. in the CAN bus. This simulation model allows the analysis and evaluation of different RL approaches, including the most promising approach of the previous stage, in an environment that better mirrors the expected behavior of the physical test bench. After comparing the results of the different implementations of the approaches, the most promising is determined for the application on the physical test bench. In the last part of the stage, the selected RL approach is implemented on the Mini-HiL physical test bench. The performance of the selected approach is presented and evaluated. Finally, a discussion of the results and a conclusion on the application of the approach take place.

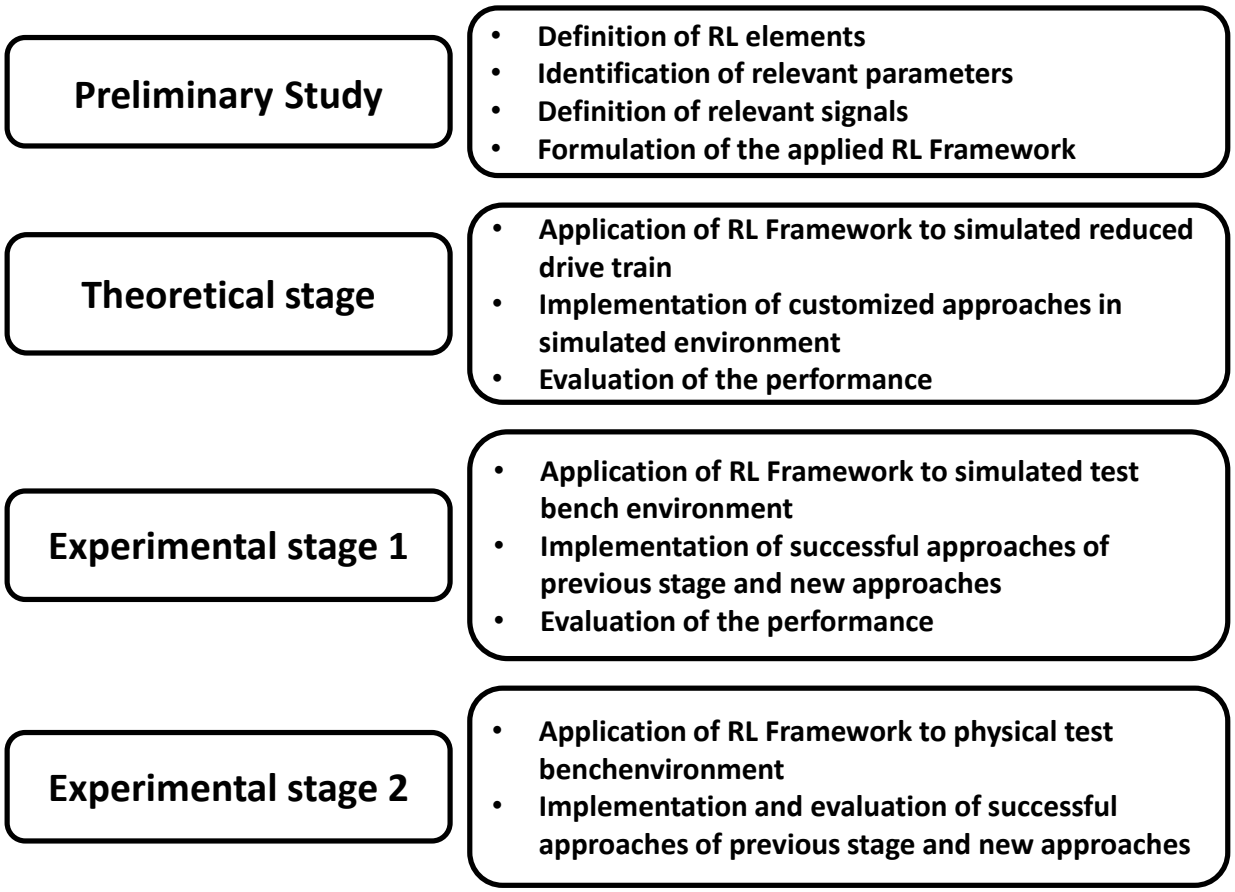

Figure 4.1: Graphical overview of the proposed research design

5 Clutch Judder Reduction as a RL Task

The first step towards the implementation of a RL based control algorithm for the active damping of judder vibrations is the description of the task as a RL problem, i.e. as a MDP. Violations of the Markov property will occur due to discretization or approximation of the continuous environment, however, their effect can be held to a low enough level so that satisfying results can be achieved nonetheless. This is achieved by the selection of proper state and action variables and an adequate discretization or approximation of the value function in the state-action space. Furthermore, the definition of the agent's environment is provided.

5.1 Definition of the Environment

Throughout this work, different models of the dynamically separated, vibration capable drive train are regarded as the agent's environment. The clutch engagement and synchronization maneuver is described as an episodic task with a set of terminal states derived from the maneuver itself. In this section, the realization of both the environment and the maneuver are described.

5.1.1 Description of the Drive Train

Over the course of this work, different models of the drive train are used to model the agent's environment. The models vary in their degrees of abstraction and in the presence of hardware, i.e. physical elements. However, all different models are based on the powertrain model as a three-mass rotational oscillator as suggested by MAUCHER[189] and depicted again in Fig. 5.1. It is considered that this elemental description of the system suffices as a means to reproduce the response of a physical drivetrain.

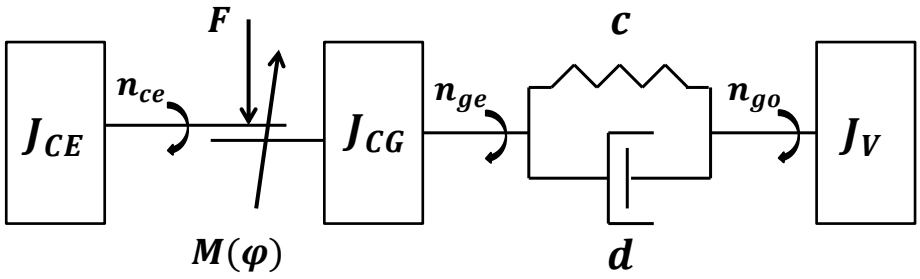

Figure 5.1: Three-mass model of the reduced drive train according to MAUCHER[189]

The detailed mathematical description of the environment during each of the research stages proposed in chapter 4 will be provided in each of the respective chapters. The

[189] Maucher 1990

decisive differences between them lie in the absence or presence of physical elements and the abstraction level of the simulated components. The presence of physical elements inevitably introduces a stochastic element to the response of the environment to a given action that is not known well enough to be accounted for in simulation.

5.1.2 Description of the Maneuver

The last part of the description of the environment is the definition of its terminal states. Although the environment itself is of a continuous nature and therefore the state-signal needs necessarily be discretized or approximated, it is possible to describe the clutch engagement and synchronization maneuver as an episodic RL learning task.

In the context of this work, an episode of the task consists of a start-off maneuver in which the clutch is actuated every $0.1\,s$ [190] and the gearbox entry shaft speed is synchronized with the combustion engine speed. The initial value of the engine speed is $900\,1/min$, whereas the gearbox input and gearbox output speeds are both $0\,1/min$. When the condition is met, that both the engine speed and the gearbox input shaft speed are synchronized, the current state is considered terminal and the episode is concluded. The exact definition of this stop criterion for the three implemented frameworks is provided in the corresponding chapter.

A sample of a simulated synchronization maneuver with strong judder vibrations is depicted in Fig. 5.2.

5.2 Definition of the State-Signal

In order to reduce the computing effort and avoid the “curse of dimensionality” mentioned in 2.2 it is sensible to reduce the state-signal to as few state variables as possible. If the signal is discrete, then it should also be as roughly discretized as possible. However, leaving important state variables out of the state-signal or roughly discretizing them entails a loss of information known as “perceptual aliasing”[191], by which the agent perceives at least two different states of the environment as the same. This inability to discern between different states of the environment leads to potential errors in the mapping of states and actions, since an action taken during one perceived state might not necessarily be the best action for all the states the agent perceives as

[190] The frequency necessary to produce a force impulse that is exactly opposite to the judder vibrations is the same than the vibrations themselves, thus the eigenfrequency of the drive train at around *8Hz - 12Hz*. This translates to an actuation of the clutch every approx. *0.083s – 0.125s*, preferably even higher (cf. Krüger 2003). However, this lies beyond or close to the maximum response speed of the test bench actuator in this work and can present a challenge to real time communication systems. In this work, an actuation time of *0.1s* is implemented (cf. 7.2.1.3), but the different algorithms would retain their functionality if it is changed at a later time. However, the results achieved with different actuation times may of course vary.

[191] Crook / Hayes 2003 and Whitehead / Ballard 1990

one and the same. Perceptual aliasing constitutes a violation of the Markov property of the state-signal. Hence, the design of state-signal should be determined in a way that the property is upheld as far as possible while it keeps the computing effort low.

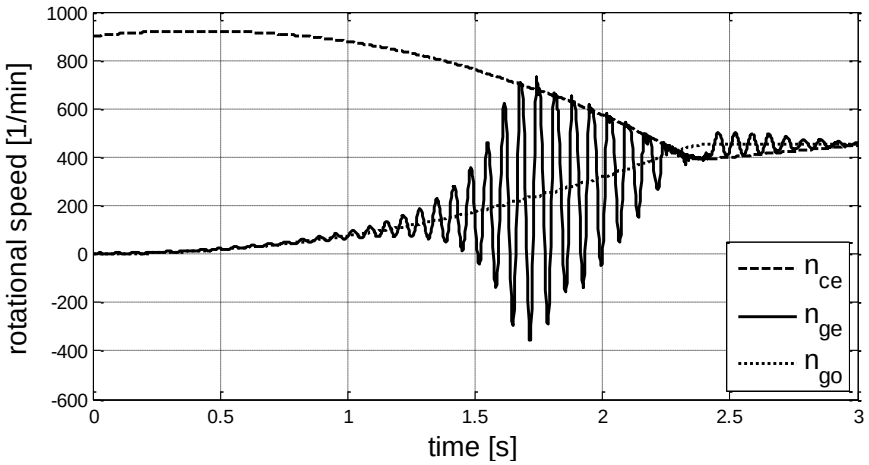

Figure 5.2: Clutch synchronization maneuver. Here, n_{ce} is the combustion engine speed, whereas n_{ge} and n_{go} are the speeds at the gearbox input and output respectively.

Furthermore, the state variables in the state-signal have to be computable or measurable at every time step of the task. Keeping in mind the application on a physical test bench or actual drive train, this means that the necessary sensors for the acquisition of the value of the signal variable should be considered.

Throughout this work, the conditions of the drive train's environment[192] are not considered for the design of the state-signal. Even though variables such as outside temperature, humidity, or air pressure likely have an effect on the behavior of the drive train, they are considered to be either stationary or their effect is considered to be negligible in comparison to other variables.

Furthermore, external clutch judder exciting deviations and misalignments of drive train components are also considered to either be stationary or simply unavailable for measurement during operation. Therefore, they are not considered for the state-signal either.

For this reason, the status of the output side of the clutch, as a vibration capable system, is considered for the selection of the state variables.

[192] This is not to be confused with the agent's environment. The drive train's environment refers to the system "Environment" of the system triple "Vehicle", "Driver" and "Environment". It is only a part of the agent's environment, which includes the drive train itself.

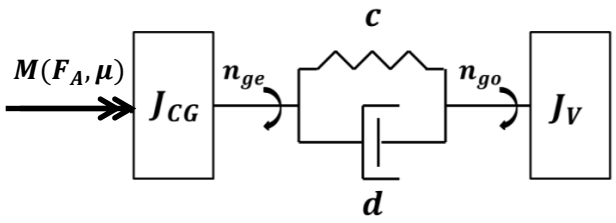

Figure 5.2: Dynamically separated, vibration capable drivetrain

Therefore, the following physical parameters remain as viable variables for the state-signal:

- the added inertias of the clutch and the gearbox J_{CG}
- the inertia of the vehicle J_V
- the stiffness c of the drive train
- and the damping d of the drive train
- the rotational speed of the gearbox input shaft n_{ge}
- the rotational speed of the gearbox output shaft n_{go}
- the clamping force F_A exerted by the clutch actuator
- the friction coefficient μ
- the clutch torque $M(F_A, \mu)$ as a result of the clamping force and the friction coefficient

The inertias J_{CG} and J_V as well as the drive train's stiffness c and damping d are considered to have a constant value throughout the test maneuver and are therefore not suitable variables for the state-signal.

The two first variables in the state-signal are the rotational speeds n_{ge} and n_{go}. They are the most important values for the characterization of the oscillation behavior of the system and sensors for their measurement are readily available in modern series production drive trains. These variables are also important for two further purposes. First, the third state variable is the result of the calculated value of the speeds at different time steps, as will be addressed shortly. Secondly, the reward-signal perceived by the agent after each interaction with the environment is derived from the difference in value between n_{ge} and the comparatively strongly dampened n_{go}.

The clamping force is not considered as an appropriate element of the state-signal since it is the only possibility of actuation and therefore the only parameter suitable as the action-signal for a RL agent.[193]

The value of the friction coefficient consists of a static and a dynamic part. Although the static part can be regarded as constant, the dynamic part can be derived from a characteristic curve (friction gradient) which requires the use of empiric data and a suitable model, e.g. assumption of linearity in order to realize a so called friction coefficient observer[194]. Furthermore, the parameter that best suits the characterization of the state of the vibration capable system is the clutch torque. Knowledge about the status of the friction coefficient is superfluous, if the value of the torque is known.

For this reason, the last state variable proposed in this work is the clutch torque. However, the availability of torque measurement units in modern, series production vehicles is restricted, because of the added cost it represents. Therefore, a method to calculate or approximate the clutch torque is necessary for the completion of the state-signal. SCHWENGER[195] proposes such a method and determines that the clutch torque correlates nearly linearly with the torsion angle in the drive train, which in turn can be calculated from the rotational speeds n_{ge} and n_{go} at the gearbox input and output respectively as follows:

$$\varphi(t) = \int_{t-1}^{t} (n_{ge}(\tau) - n_{go}(\tau))d\tau \qquad \textit{Eq. 5.1}$$

The linear interrelationship between the clutch torque M and the torsion angle θ can be taken from in Fig. 5.2.

The state-signal including the three proposed state variables provides a Markov description of the environment for the agent, if all assumptions are met.

Thus, the resulting state-signal of the environment at the time t consists of the value triple:

$$s_t = (n_{ge}, n_{go}, \varphi)_t \qquad \textit{Eq. 5.2}$$

[193] In the next section, two different possibilities for designing the action space are proposed, one of which requires the inclusion of the current value of the clamping force, since the actions selectable by the agent are given as an offset to be added to this value.

[194] Dolcini et al. 2010 use such a friction coefficient observer for the design of an adaptive controller.

[195] Schwenger 2005

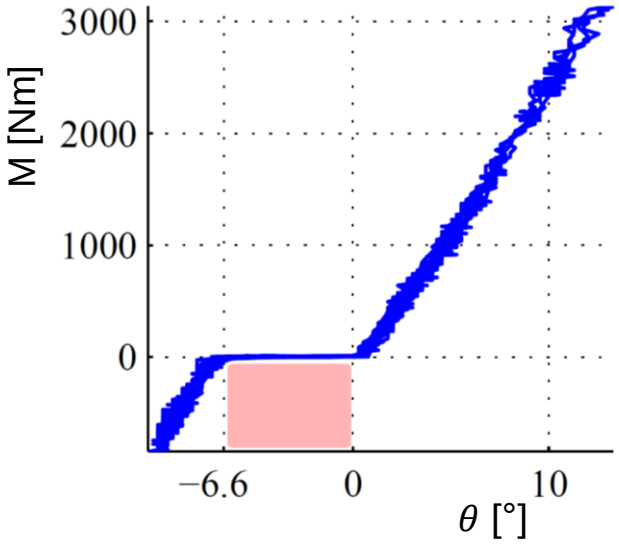

Figure 5.2: Linear interrelationship between torque and torsion angle according to SCHWENGER[196]. The pink rectangle signals the clearance in the drive train.

The discretization of the state space and /or the approximation of the value function for different state-action space configurations are a central issue throughout this work, as they strongly affect the tradeoff between computing effort and quality of the solution. Different methods that favorably address the tradeoff are proposed in chapter 6.

5.3 Definition of the Action-Signal

The requirements on the action-signal in terms of its variables and their discretization are not different from those for the state-signal. On the one hand, more variables discretized more smoothly cause an increase in computing effort but deliver the best results. On the other hand, an action-signal with fewer variables discretized more roughly reduces the ability of the agent to affect its environment and might lead him to either bad solutions or could even make the task impossible to solve for him.

Furthermore, the action-signal is of function relevance for the task and maneuver at hand. An inappropriate design of the action-signal might not only lead to no reduction of judder but might also cause the function of the clutch to be lost. If the necessary clamping force for the transmission of torque is not applied between the clutch disks, the vehicle is not accelerated since synchronization between the engine and the drivetrain is not achieved. Therefore, an action-signal that allows the agent to affect its environment enough to reduce clutch judder without compromising the functionality of the clutch is sought after.

The action-signal can be formulated more easily than the state-signal, since the only parameter that can actively be set by the agent is the clamping force F_A. However, the design of the action space is worth addressing.

[196] Schwenger 2005

Generally, the first distinction that needs to be made is whether the action space is to be continuous or discrete. Due to the discrete nature of the digital control of physical actuators, a discrete implementation of the action space is proposed.[197]

In chapter 6.2, three different approaches are proposed and analyzed and the most promising is identified. This approach is implemented in all further experiments.

5.4 Definition of the Reward-Signal

The definition of a reward-signal that is available for the agent after each interaction with its environment and accurately evaluates the result of said interaction is a key element in the RL framework. The agent uses the short term reward he acquires to learn the best behavior in order to maximize the long term reward he expects to perceive.

The requirements on the reward-signal can be reduced to two essential qualities. First, it needs to reward the actions by the agent that lead to the desired behavior, i.e. reduction of judder vibrations with focus on passenger perception. Secondly, the value of the reward-signal needs to be available after each agent-environment interaction, since the methods implemented in this work aim to exploit the advantages of incremental computation and therefore cannot wait until the end of an episode to update the values in the value function.

The methods for the quantification of clutch judder presented in 2.1.3 comply well with the first requirement but are not suitable for an incremental implementation. For example, the method proposed by ALBERS AND KRÜGER[198] and KARRAR[199] requires the value of the upper and lower envelope curves before the area index can be calculated. However, the value of the envelope curve at a given time during the maneuver is not necessarily available. Furthermore, vibrations with high amplitude but a short duration can be expected to cause a jerk strongly perceived by a passenger despite having only a moderate effect on the area index. On the other hand, the effect of low amplitude vibrations persistent throughout the maneuver is usually negligible regarding passenger perception because of the damping in the drive train, chassis, seats, etc. The VDV introduced by GRIFFIN[200] offers a potential value for the reward-signal since it can be computed for very short durations and it accounts for the effect of high amplitude vibrations. In this work, different approaches similar to the VDV are analyzed by means of the two exemplary vibrations depicted in Fig. 5.4. The first exemplary vibration represents the case of persistent low amplitude vibrations unlikely to have an

[197] For an insight into RL in continuous action spaces refer to e.g. van Hasselt / Wiering 2007.
[198] Albers / Krüger 2003b
[199] Karrar 2009
[200] Griffin 2007

adverse effect on passenger perception of comfort, whereas in the second case a high amplitude vibration of short duration is used to simulate undesired jerk.

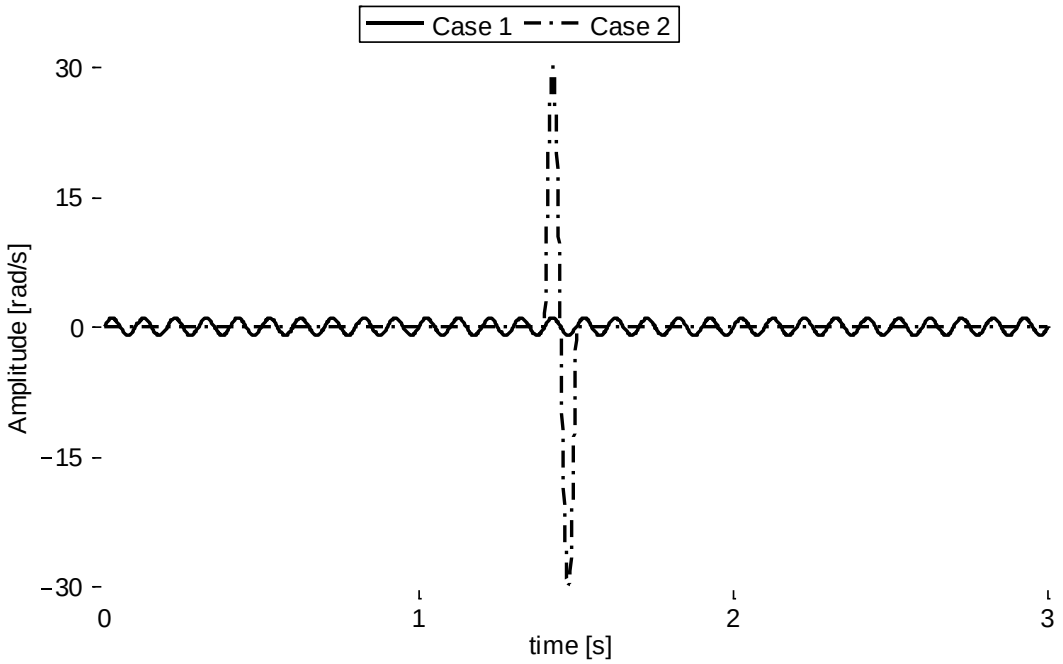

Figure 5.4: Exemplary cases for rotational speed vibrations

Throughout this work, the amplitude relevant to the task is the difference speed between gearbox input and output n_{dif}:

$$n_{dif} = n_{ge} - n_{go} \qquad \textit{Eq. 5.3}$$

On order to determine an appropriate definition of the reward-signal, the exemplary cases are analyzed using different established methods. The formula that defines the value of the vibration according to the specific method and the ratio between their values for the presented exemplary cases is contained in Table 5.1.

Method	**Formula**	**Ratio (case 2 to case 1)**
Amplitude	$\max\lvert n_{dif}\rvert$	30
Area	$\int_0^T n_{dif}(t)dt$	1
Mean Square	$\frac{1}{T}\int_0^T n_{dif}(t)^2 dt$	30
Mean Cube	$\frac{1}{T}\int_0^T n_{dif}(t)^3 dt$	900
Mean Quad	$\frac{1}{T}\int_0^T n_{dif}(t)^4 dt$	27000
Root Mean Square	$\sqrt{\frac{1}{T}\int_0^T n_{dif}(t)^2 dt}$	5,5
Root Mean Cube	$\sqrt[3]{\frac{1}{T}\int_0^T n_{dif}(t)^3 dt}$	9,7
Root Mean Quad	$\sqrt[4]{\frac{1}{T}\int_0^T n_{dif}(t)^4 dt}$	12,8
Vibration Dose	$\int_0^T \dot{n}_{dif}(t)^4\, dt$	27000
Vibration Dose Value (VDV)	$\sqrt[4]{\int_0^T \dot{n}_{dif}(t)^4\, dt}$	12,8

Table 5.1: Vibration evaluation methods

The main drawback of an area based measurement of the judder vibrations can be taken from the ratio column. The area for both vibrations is the same and they are therefore rated equally. The cubic and quadratic values cause the range of values to explode, so very big numbers arise, whereas the “root” methods reduce this range, but only through an additional mathematical operation. As a compromise between the size of the value range and the simplicity of the method the mean square method is selected for the design of the reward-signal in this method. Due to its incremental implementation, a sum is formulated instead of an integral and is subsequently negated since high amplitude vibrations should be negatively perceived:

$$r_n = -\frac{1}{n}\sum_{i=0}^{n}(n_{dif})^2 \qquad \textit{Eq. 5.4}$$

The term *n* refers to the simulation steps or measurements used to calculate the reward. If the reward is calculated after every interaction step only, then $n = 2$ and the reward becomes the difference of n_{dif} :

$$r_{t+1} = {n_{dif}}_{t+1}^{\ 2} - {n_{dif}}_{t}^{\ 2} = \Delta n_{dif} \qquad \textit{Eq. 5.5}$$

By maximizing the long term reward presented in this chapter, the agent learns to actuate the clamping force in order to synchronize the output and input sides of the clutch while avoiding high amplitude vibrations and treating low amplitude vibrations more leniently.

Given the nature of the task and the definition of the reward-signal, the highest possible reward equals 0, as a result of a perfect synchronization of the input and output speeds. Since this reward is nearly unattainable in practice, all values that are updated as a result of an interaction are likely to be < 0.Thus, every unexplored state-action pair has a higher value than those already tried out. Therefore, a strong explorative behavior is forced at the beginning of a learning process, extending its duration but increasing its chances of finding an optimal solution in turn. SUTTON AND BARTO[201] refer to this kind of problem as one with *"Optimistic Initial Values"*.

5.5 Definition of the Agent

As stated in 2.2.2, the agent is the entity considered to be learner and decision maker. In a simplistic sense, the agent can be regarded as a set of lines of code used for the tasks of updating the value (or action-value) function after an interaction with the environment and determining the action to be taken next.

The first part of the task is addressed through the selection of the update rule according to the RL method being followed. The second part is achieved through the definition of a policy, e.g., ε-greedy.

Throughout this work, tabular TD methods are favored, since they are suitable for the discrete problem at hand. Implementations of both SARSA and Q-Learning are proposed. The corresponding update rules were introduced in 2.2.4.3. Therefore, the information about the interactions between the agent and the environment is stored in a so-called Q-Table. The Q-Table is the tabular value function in which the desirability of the actions at every state, thus the value of the actions at those states, is mapped. It can also be considered the solution space itself, since it contains all possible combinations of actions and states discernable by the agent.

Having provided the definition of the state and action-signals, the structure of the Q-Table is that of a 4-dimensional matrix:

$$Q(s,a) = \langle n_{ge}, n_{go}, \varphi, F_A \rangle \qquad \textit{Eq. 5.3}$$

[201] Sutton / Barto 1998

The simplest form of a discrete Q-Table consists of the discretization and listing of the different state variables into a state list and repeating the same procedure with the action space. An example for the structure of a small tabular Q-Table is presented in Fig. 5.5.

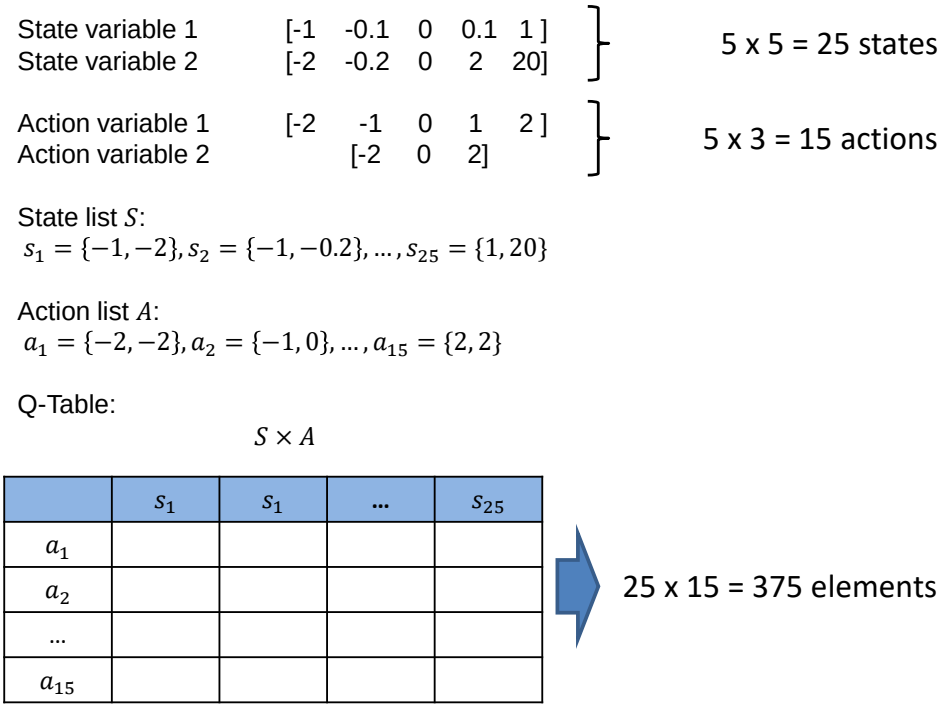

Figure 5.5: Example of the structure of a tabular Q-Table for discrete state and action variables

Even though the Q-Table in the provided example has “only” 375 elements, RL applications generally struggle with the “curse of dimensionality” mentioned in earlier chapters. As an example, the smallest Q-Table in this work consists of 21600 elements.

In the following chapters, different approaches are presented for the structure of the Q-Table depending on the definitions and implementations of the state and action spaces and the corresponding state and action lists.

Finally, the action selection by the agent throughout this work is implemented using the ε-greedy method introduced in 2.2.1. For certain cases, however, the range of selectable actions or the behavior of the agent during explorative actions is altered to either improve the explorative behavior, boost the learning speed or exert the necessary clamping force by the end of an episode.

5.6 The RL Framework for the Clutch Judder Reduction Task

Over the course of this chapter all the elements of the reinforcement learning task have been applied to the clutch judder reduction task. The different applications and

experiments presented in the following chapters share the same general framework, albeit the different individual elements are implemented differently.

Thus, the RL framework for the judder reduction task through the active suppression of vibrations can be taken from Fig. 5.6.

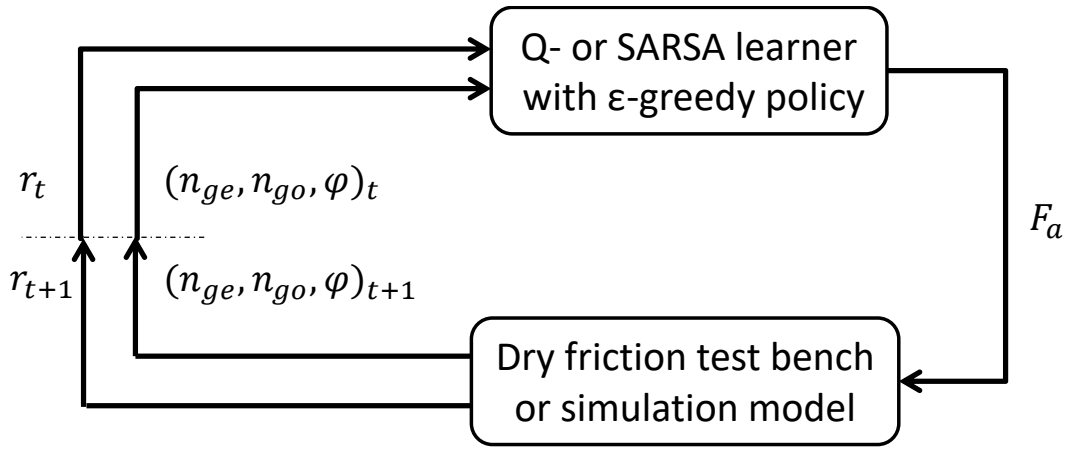

Figure 5.6: Generalized RL framework for the clutch reduction task

The framework can be described as follows: a Q- or SARSA algorithm uses an ε-greedy policy to calculate a strategy for the actuation of the clutch via the clamping force F_a in dependency of the input and output velocities of the clutch and the torque angle in order to maximize the reward r_t, i.e. to minimize the difference in speed between the gearbox input and output. The task is episodical and consists of a start-up maneuver, in which a terminal state is reached when the velocities of the engine and the vehicle are synchronized and have reached the target speed, or a given time limit for the synchronization is surpassed.

The realization of the individual elements of the RL framework for the different implementations in this work is presented in the following chapters.

6 Implementation of the RL Framework on an abstract Simulation Model of the Drive Train

In the first stage of the research, the focus lies in the development of adequate state and action space designs. The state variables have been introduced in the previous chapter, however, the concrete implementation of the state and action spaces that determine the structure of the Q-Table remain unspecified.

First, a highly abstract three-mass model is introduced as the agent's environment as a means to provide a preliminary research platform that allows the study of different approaches at low implementation and computing effort.

Afterwards, three different approaches for the conception of the action space are introduced on a RL algorithm with a standardized discretization of the state space (introduced shortly after in 6.3.1).

One key element of this work is the way in which the conception of the state space and the updating of the value function are implemented in order to achieve satisfying results. In 6.3 the standard approach for the discretization of the state space and the estimation of the value function is implemented for the RL task described in the previous chapter. Additionally, three novel approaches developed in this work that aim at the reduction of computing effort are presented. The first consists of a dynamic state list in which an initially roughly discretized state list is refined locally in order to allocate computing efforts more efficiently. The second approach consists of the implementation of a function approximation method with radial basis functions in order to estimate the value function in a semi-continuous state space. Finally, in the third approach, a Gauss filter-function is applied to approximate the value function in an entirely discrete state space.

To conclude this chapter, an analysis of the response of a selected RL controller to changes in its environment is performed. The focus herein lies in the ability of the agent to learn a new behavior that suits a permanently changed environment better, hence its adaptivity, and its performance under fluctuating system parameters, hence its robustness.

6.1 Simulation Model of the Environment

The three-mass rotational oscillator used to model the environment in this chapter is implemented by HORNUNG[202] after NAUS ET AL.[203] and is depicted in Fig. 6.1.

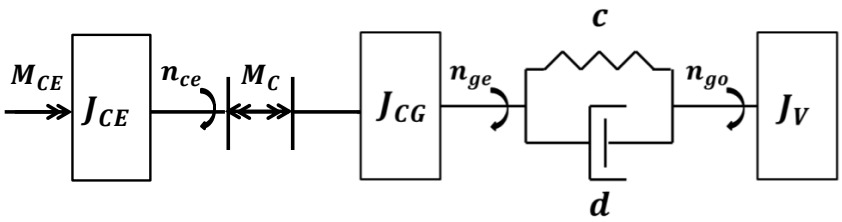

Figure 6.1: Three mass oscillation model after NAUS ET AL.[203]

The reduction of the physical drive train yields the set of equations *Eq.6.1* for the vibration system:

$$\begin{aligned} J_{CE}\dot{\omega}_{ce} &= M_{CE} - M_C \\ J_{CG}\omega_{ge} &= M_C + c\theta + d\dot{\theta} \\ J_V\omega_{go} &= -c\theta - d\dot{\theta} \\ \dot{\theta} &= \omega_{go} - \omega_{ge} \end{aligned} \qquad \textit{Eq. 6.1}$$

where θ is the torsion angle in the drive train. J_{CE}, J_{CG}, J_V and $\omega_{ce}, \omega_{ge}, \omega_{go}$ are the inertias and angular speeds of the combustion engine, the gearbox input shaft and the gearbox output, respectively. Finally, M_{ce} and M_C are the torque of the combustion engine and the clutch. Analogously to the definitions in chapter 2.1.1, the clutch torque yields:

$$M_C = R_M * n * F_a * \mu \qquad \textit{Eq. 6.2}$$

where n is the amount and R_M the mean radius of friction pads. For this simulation, the friction coefficient was modelled as follows:

$$\mu = \mu_{st} + \mu' * R_M * (\omega_{ce} - \omega_{ge}) \qquad \textit{Eq. 6.3}$$

The values of the parameters can be taken from Appendix A.

The task is modelled as a discrete episodic task in steps of $0.1\,s$ that finishes either after synchronization of the engine and gearbox output (vehicle) speed or after a time limit of $3\,s$ is surpassed. Even if the synchronization happens before the given time limit, the simulations are run for $3\,s$ but no further reward is given.

---

[202] Hornung 2012

[203] Naus et al. 2008 and Naus et al. 2010

A sample synchronization process with the drive train model presented in this chapter is depicted in Fig 5.2.

The reward-signal is defined in accordance to 5.4 and the agent uses an ε-greedy policy for the selection of actions and the on-policy TD-method SARSA to update the Q-Table.

The difference between the different approaches presented in this chapter lies in the conception of the state and action spaces presented in the following sections.

6.2 Design of the Action Space

In this section, the three approaches for the realization of the action space are introduced. They are exemplified on a clamping range of $0\ kN \leq F_A \leq 12\ kN$ of the actuation force of the clutch in the unscaled simulation model introduced in this chapter. However, the value range can be effortlessly applied to the scaled force range of the models in the following chapters. Since the most promising approach is used for all simulations in the following sections, it is introduced prior to the design of the state space. For comparison purposes, all approaches found in this section were tested with the standard, homogenously discretized state space described later in 6.3.1.

6.2.1 Standard Action Space: Homogenous, unrestricted Force Range

The approach referred to as “standard” throughout this work consists of a homogeneous discretization of the full range of the clamping force range. Thus, the agent can select a discrete, absolute force from the full range at every interaction with the environment. Thus, after every interaction the agent can select an action from the entire action space that is at the same time the actuation force applied to the clutch:

$$F_A = a_t(a_t \in \mathcal{A}) \qquad \textit{Eq. 6.4}$$

In this example, the force range of $12\ kN$ was discretized in constant steps of $1\ kN$. Thus the action space yields:

$$\mathcal{A} = \{1\ kN, 2\ kN, 3\ kN, \dots, 12\ kN\} \qquad \textit{Eq. 6.5}$$

An example of the progression of action selections during a startup maneuver with the standard action space is depicted in Figure 6.2.

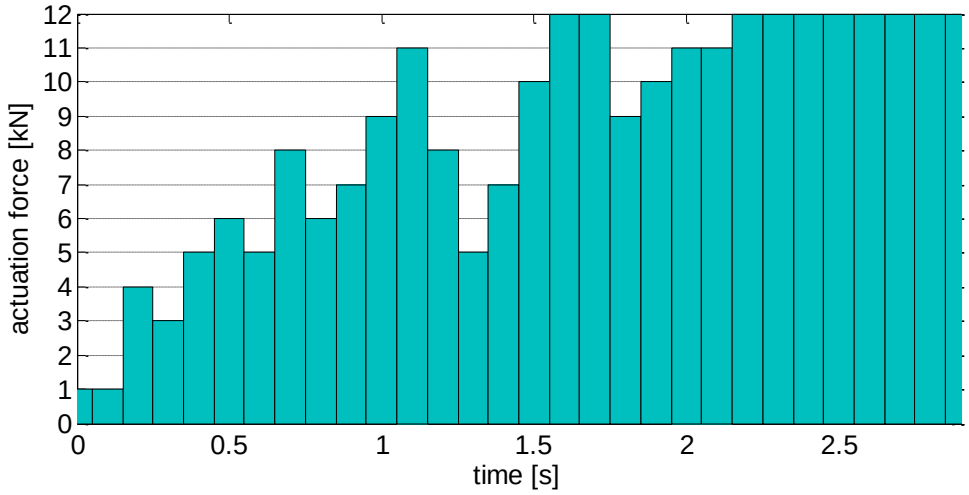

Figure 6.2: Progression of actions in a standard action space

The evident advantage of this approach is its simplicity, since the only necessary parameter to set is the step size of the discrete forces. However, the availability of all discrete forces in the range at every interaction is a major flaw in this approach that contributes to its unpracticality in two ways.

First, it is likely to result in an unachievable setpoint value of the force for the physical actuator. If in three successive interactions the actions selected by the agent result in large force differences, the physical actuator may not be able to set the demanded force. As an example, if the maximal force difference that can be set in a time step is $8\,kN$, then the result of the environment to the agent selecting the actions $F_A = 12\,kN$ or $F_A = 9\,kN$ when the current force is $F_A = 1\,kN$ is the same. If the history of actions is not accounted for additionally, this case would lead to violations of the Markov property and therefore equivocal mappings of actions to states.

Secondly, the availability of low force values near the end of an episode, where high forces are required in order to achieve synchronization, is unnecessary since the agent would need to experience their effect first, before learning that they are indeed unsuitable at that point of the episode, therefore increasing the computing effort for an adequate strategy. This makes it necessary to impose the selection of actions that lead to the required torque for synchronization late in the episode.

In the following two sections, these negative effects are addressed by means of alternative state space designs.

6.2.2 Alternative Action Space: Force Modulation

An approach to neutralize the negative effects of the standard design of the state space consists in the realization of a force modulating agent. Instead of having a set of different clamping forces to choose from, the actions available to the agent are a set of modulating factors that can be applied to a standardized force ramp that ensures

the required progression of the clamping force. Therefore, the agent learns to alter the given progression of the force slightly to reduce vibrations without compromising or delaying the synchronization process significantly.

In this implementation, a set of five modulation elements were defined. The value of the action selected is added to the current value of the standard force ramp. The clutch actuation force modulation action space can be visualized as depicted in Fig. 6.3.

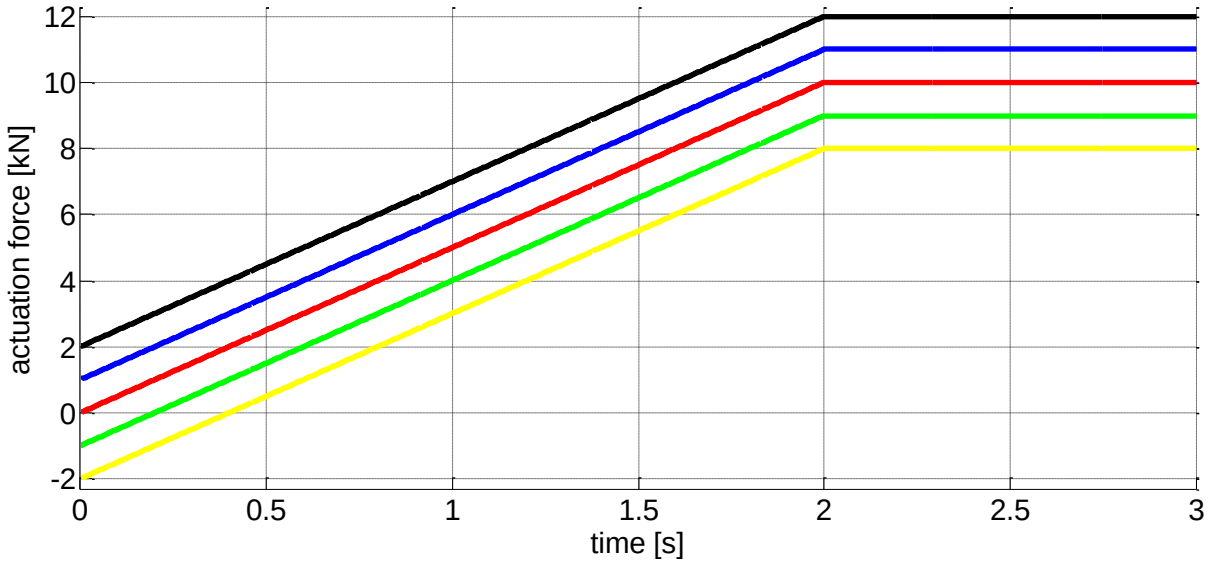

Fig. 6.3: Action space for the force modulation design.

The curve in the middle corresponds to the standard progression. The agent can choose one of the following values to add to this progression, resulting in the other five curves depicted in the figure:

$$\mathcal{A} = \{-2\ kN, -1\ kN, 0\ kN, 1\ kN, 2\ kN\} \qquad \textit{Eq. 6.6}$$

Thus, the actuation force at the clutch at a given time step t yields:

$$F_A(t) = F_{standard}(t) + a_t(a_t \in \mathcal{A}) \qquad \textit{Eq. 6.7}$$

An exemplary progression of the clamping force with a force modulating agent for interaction steps of $0.5\ s$ can be taken from Fig. 6.4. The red curve signals an exemplary clamping force setpoint determined by the agent.

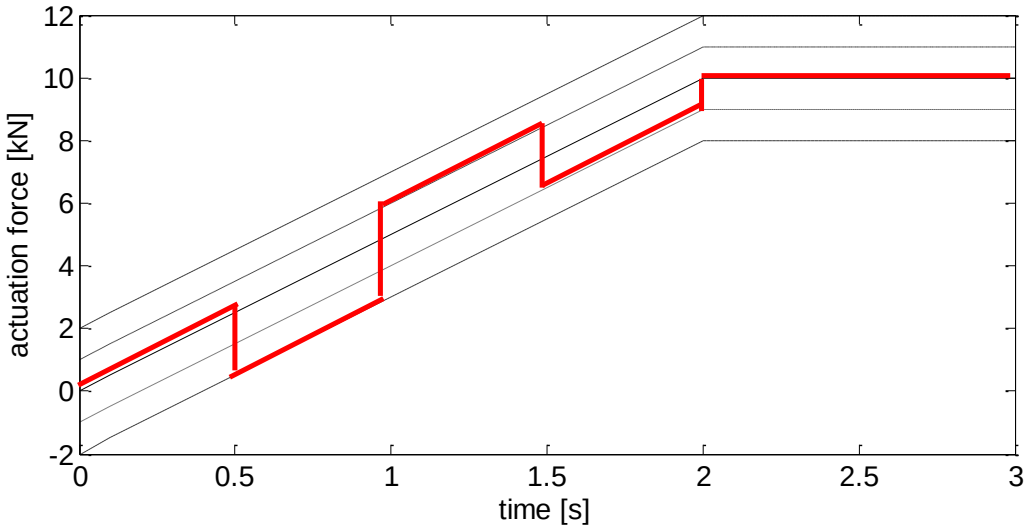

Figure 6.4: Exemplary progression of the actuation force with a force modulating agent

The force modulation alternative provides an effective solution to the problems that arise from the implementation of a standard action space. However, the positive features of this approach come at a high cost. Even though the action space itself can be designed with less discrete elements, it has a negative effect on the computing effort since it requires the extension of the state space by at least one additional state variable. This problem arises from the fact that the value of the standard force ramp is a function of time, as can be taken from *Eq. 6.7*. The selection of the same action at different points in time results in different actuation force. Therefore, either the value of the standard ramp at the time, or the time itself must be known to the agent as additional information from the environment, or else the Markov property is violated. This undesired expansion of the state space is considered a criterion for exclusion which is why the force modulation approach is not further considered in this work.

In order to avoid the expansion of the state space and the increase in computing effort it entails but retain the positive aspects of this action space design, a third approach to the implementation of the action space is introduced and compared to the standard action space in the next section.

6.2.3 Alternative Action Space: Restricted Force Range

In this work, an implementation of the state space is introduced that restricts the actions available to the agent for selection. Therefore, only a part of the action space is available at every step. In principle, the action space is also homogenously discretized in the same fashion as the standard implementation presented in 6.2.1. However, instead of permitting any action in the state space to be selected at any time, the agent is given only a segment of the state space to select actions from. Early in the episode lower values for the clamping force are available and as the episode proceeds, higher values for the force are made available gradually. Formally, the agent chooses an

action from the time dependent segment of the action space. Formally this is described as follows:

$$F_A = a_t(a_t \in \mathcal{A}_t) \qquad \textit{Eq. 6.8}$$

The time dependent action spaces available for selection at an early time t_1 and at a late time t_2 in the episode are exemplified in the following equations:

$$\mathcal{A}_{t_1} = \{1\ kN, 2\ kN, 3kN\} \qquad \textit{Eq. 6.9}$$

$$\mathcal{A}_{t_2} = \{8\ kN, 9\ kN, 10\ kN, 11\ kN\} \qquad \textit{Eq. 6.10}$$

The progressive increase in value of the available actions guarantees synchronization by the end of the episode. Furthermore, the predefined availability of selectable actions actually reduces computing effort, since it limits the solution space. However, the time dependency of the action space entails the risk that the action that could lead to highest reward is not available at a given time. A graphic visualization of the design of the restricted state space can be taken from Fig. 6.5.

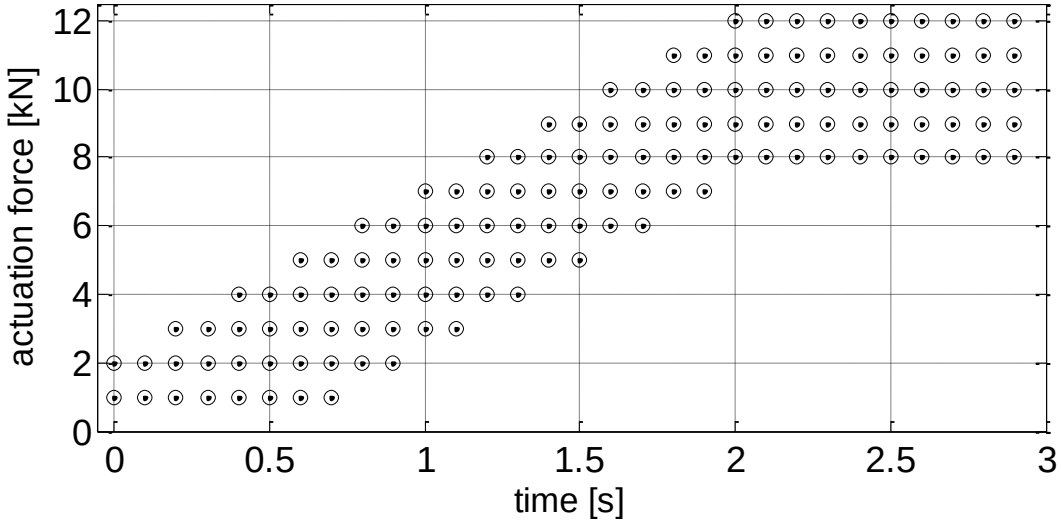

Figure 6.5: Example of available actions during an episode with a restricted action space

Lastly, the performance of the agent in the different action spaces is analyzed. For this purpose a comparison of the learning curves, i.e. the graph containing the progression of the reward gained throughout several episodes, is performed. Every aspect of the algorithm except for the design of the action space is identical. The results can be taken from Fig. 6.6.

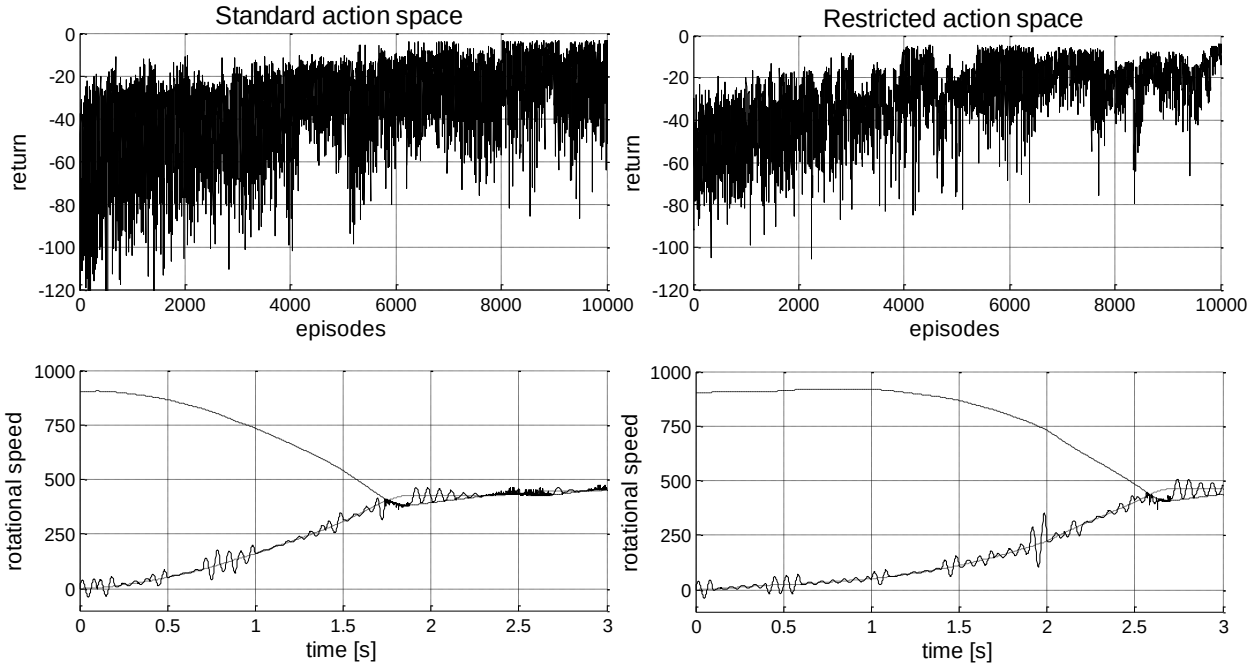

Figure 6.6: Comparison between the standard and the restricted action spaces over 10,000 episodes[204]

The figures in the upper half of Fig. 6.6 depict the progression of the accumulated reward, i.e. the return, over the course of synchronization maneuvers, i.e. episodes. The comparison shows how the standard and the restricted action spaces from Fig. 6.5 achieve similar returns after 10,000 repetitions of the synchronization maneuver. However, the converged solution of the standard action space completes the synchronization after approximately $2.0\,s$, whereas with a restricted action space the synchronization takes approximately $2.5\,s$. This is due to the previously mentioned problem that the restriction on the action space often makes the best actions unavailable to the agent. This is evidenced by the progression of the converged action selection in the unrestricted state space. In Fig. 6.7, the comparison of the action sequence for both approaches is depicted.

[204] The damping of the drive train in the simulation model is very low, thus exacerbating the vibrations. Since the agent's sole purpose is to reduce the amplitude of vibrations, it does so by extending the duration of the maneuver and allowing the engine to be slowed down further than it would be acceptable in practice.

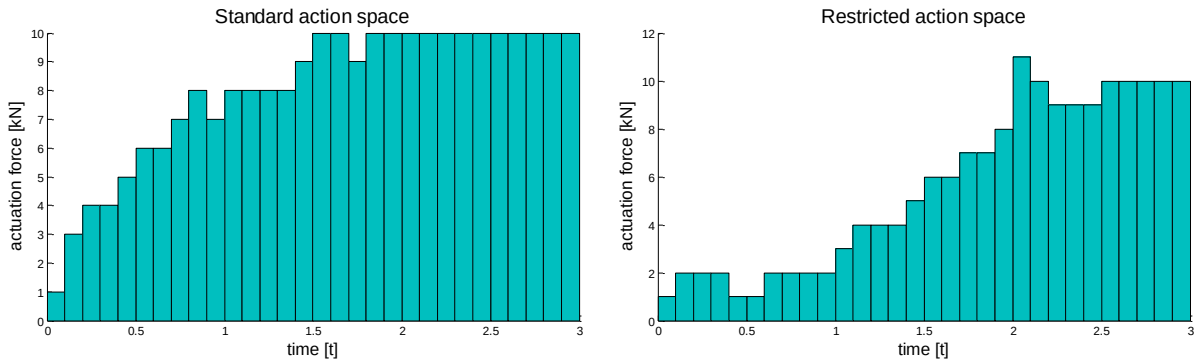

Fig. 6.7: Comparison of the converged action sequences for the standard and restricted action spaces

However, the performance of the agent in a restricted action space can be improved once the best action sequence is known. In order to do so, the restriction of the action space is adjusted using the results of the unrestricted space, by making higher forces available earlier. The adapted restricted action space is depicted in Fig. 6.8.

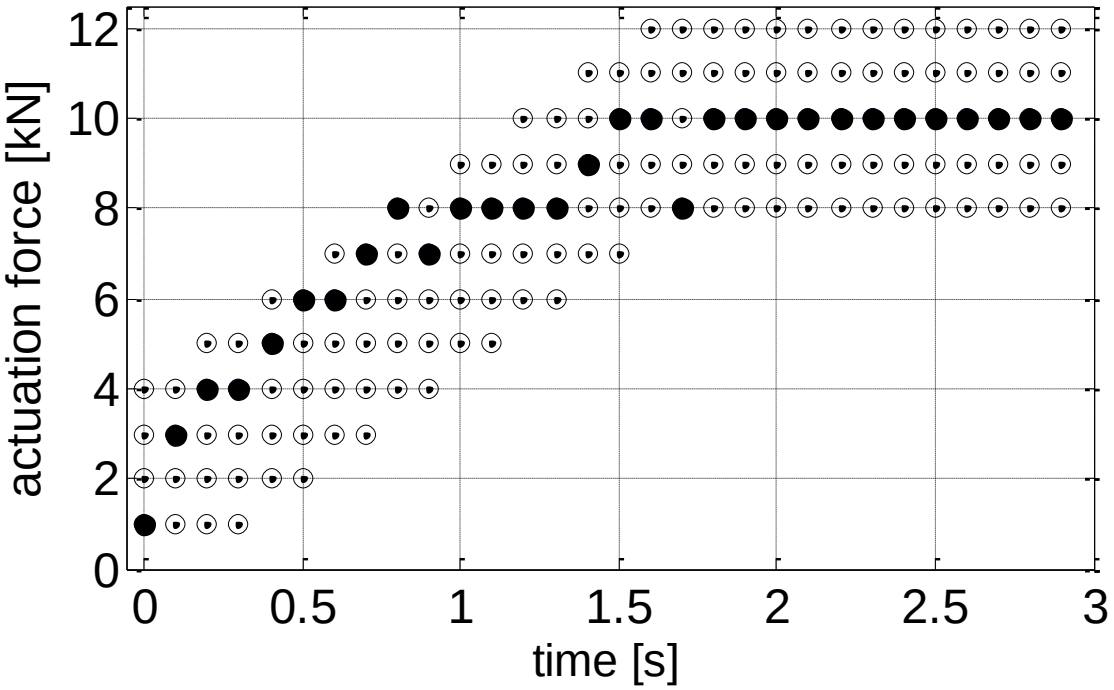

Figure 6.8: Learnt action sequence in the adapted restricted action space. The dots represent the available actions at different times. The dark spots are the selected actions at the corresponding time.

There was no significant difference between the quality of judder suppression in a standard and a restricted action space. However, the advantage the restricted action space provides is a considerable reduction in the number of episodes required to achieve this result. This is mainly due to the previous knowledge provided by the agent in the form of a preselection of actions. In comparison, the adapted restricted action

space leads to convergence[205] after approx. 4500 episodes, which is less than half the number required previously. This is evidenced by its learning curve, depicted in Fig. 6.9. In order to facilitate the recognition of the learning progression, a second degree polynomial fitting of the learning curve is depicted in red as well.

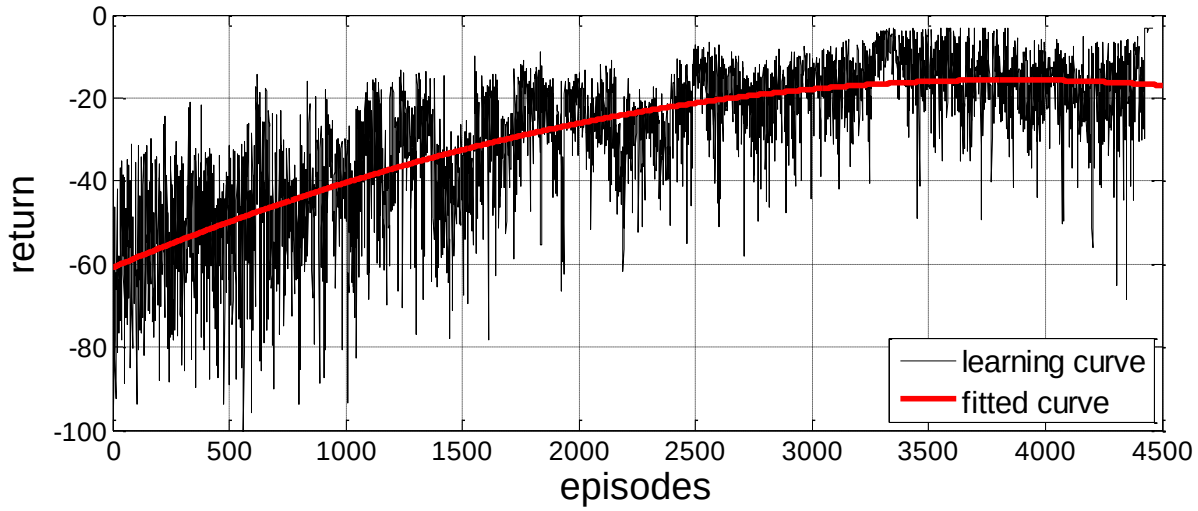

Figure 6.9: Learning curve of the adapted restricted action space (in homogenously discretized state space)

For this reason, in all further experiments in this chapter the adapted action space as depicted in Fig 6.8 is implemented. In the following section, the design of the state space is addressed.

6.3 Design of the State Space

The only element of the RL framework provided in 5.6 not addressed thus far is the design of the state space. In this section different implementations including simple discretization approaches and an approximation of the value function in a partially continuous state space are provided.

All the presented approaches are developed for the state space with three state variables presented in 5.2. Furthermore, the adapted restricted action space from 6.2.3 is implemented in all further simulations in this chapter.

Discretization has the purpose of transferring continuous models, equations or parameters into discrete counterparts usually as a means toward making them suitable for numerical evaluation and implementation on digital computers. As was mentioned in 5.2, discretization entails a loss of information known as discretization error that in the case of RL leads to violations of the Markov property. However, this loss can be reduced far enough so that the learning of appropriate solutions can be achieved.

[205] Convergence of the learning process is defined by 20 successive episodes in which the change in the return is smaller than 0.0001.

The simplest approach at discretizing the state space spanning three state variables is to create a homogenous mesh in which the value range of an individual variable is reduced to a finite number of elements with a constant step size. The permutation of all the discrete values for every variable constitutes a list of states of the environment:

$$\mathcal{S} = \{s_1, s_2, \dots, s_N\} \qquad \textit{Eq. 6.11}$$

Following a given rule, every continuous state of the environment can be assigned a discrete value of this list:

$$s_t \leftarrow s_i \ , \ s_i \in \mathcal{S} \qquad \textit{Eq. 6.12}$$

The most common way to implement this rule consists of searching in the list of discrete states of each variable for the state with the lowest Euclidean distance to the continuous value. Such a rule assigns discrete states to continuous values as exemplified in the following application to the state space in this work:

$$s_t \leftarrow s_i \ , \qquad \forall i = \underset{i}{\operatorname{argmin}}[\left(n_{ge,t} - n_{ge,i}\right)^2 + \left(n_{go,t} - n_{go,i}\right)^2 + (\theta_t - \theta_i)^2] \qquad \textit{Eq. 6.13}$$

However, the use of this rule requires the computation of the Euclidean distance between the continuous value and every discrete state before it can determine the lowest. In order to simplify computation, which is particularly important in regards to future real time implementation of the algorithm, an alternative formulation is proposed. It consists of defining areas of validity for the discrete values of parameters. This means that a certain combination of ranges of the three state variables constitutes a discrete state:

$$s_i = (\Omega_{ge,i}, \Omega_{go,i}, \Theta_i) \qquad \textit{Eq. 6.14}$$

Therefore, instead of calculating the distance to every discrete state in the state list the assignment is determined by the belonging of the continuous value to the different validity ranges:

$$s_t \leftarrow s_i \ , \qquad \forall i = \ n_{ge,t} \in \Omega_{ge,i} \ , n_{go,t} \in \Omega_{go,i} \ , n_{ge,t} \ , \varphi_t \in \Theta_i \qquad \textit{Eq. 6.15}$$

The only remaining features to define are the size of the value ranges that make up the mesh of the discrete state space and the way the values in the Q-Table, consisting of the state and action space, are updated as a result. The way these features are implemented is what distinguishes the four approaches presented in the following sections.

6.3.1 Homogenously discretized State Space

The first and simplest method for the discretization of the state space consists of defining a constant size for the value ranges of each state variable. Therefore, the values of $\Omega_{ge,i}$, $\Omega_{go,i}$ and Θ_i are constant but not necessarily the same. An example for the homogenous discretization of a two-dimensional state space can be taken from Fig. 6.11.

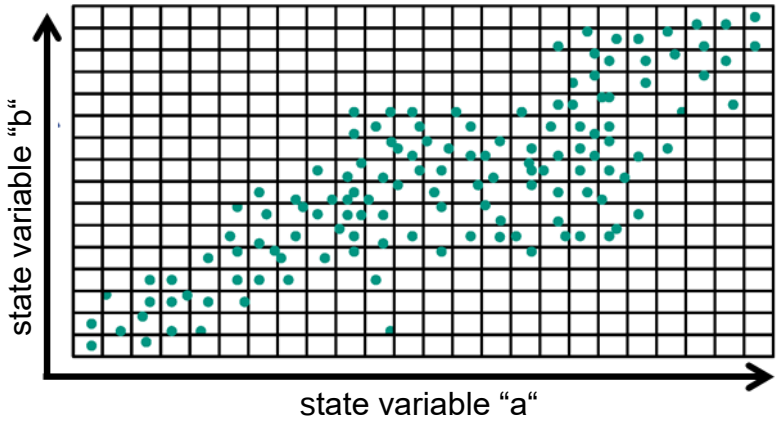

Figure 6.11: Exemplary homogenous discretization of a two-dimensional state space. The green dots represent a continuous state, whereas the grid represents the discrete state it is assigned.

The values of these parameters should be defined keeping in mind their influence on the learning process. Therefore, the size of the discrete ranges that determines the smoothness of the mesh should offer a favorable compromise for the tradeoff between computation effort and quality of the solution. However, in this work, this approach to the discretization of the state space is also meant to offer a reference for the quality that can be expected of the implementation of the RL algorithm. For this reason, the highest available smoothness of the homogenous mesh is targeted. Unless other elements of the framework are changed, the quality of the solution that results of the learning process in the homogenous discrete state with the smoothest mesh has the highest quality. However, it also causes the highest computing effort, since it results in the most extensive state space and, therefore, the most extensive solution space.

In order to determine a sensible size for the discrete mesh, several iterations of the learning process are simulated with decreasing values for Ω_{go} , Ω_{ge} and Θ. The results of this simulation are contained in Fig. 6.12.

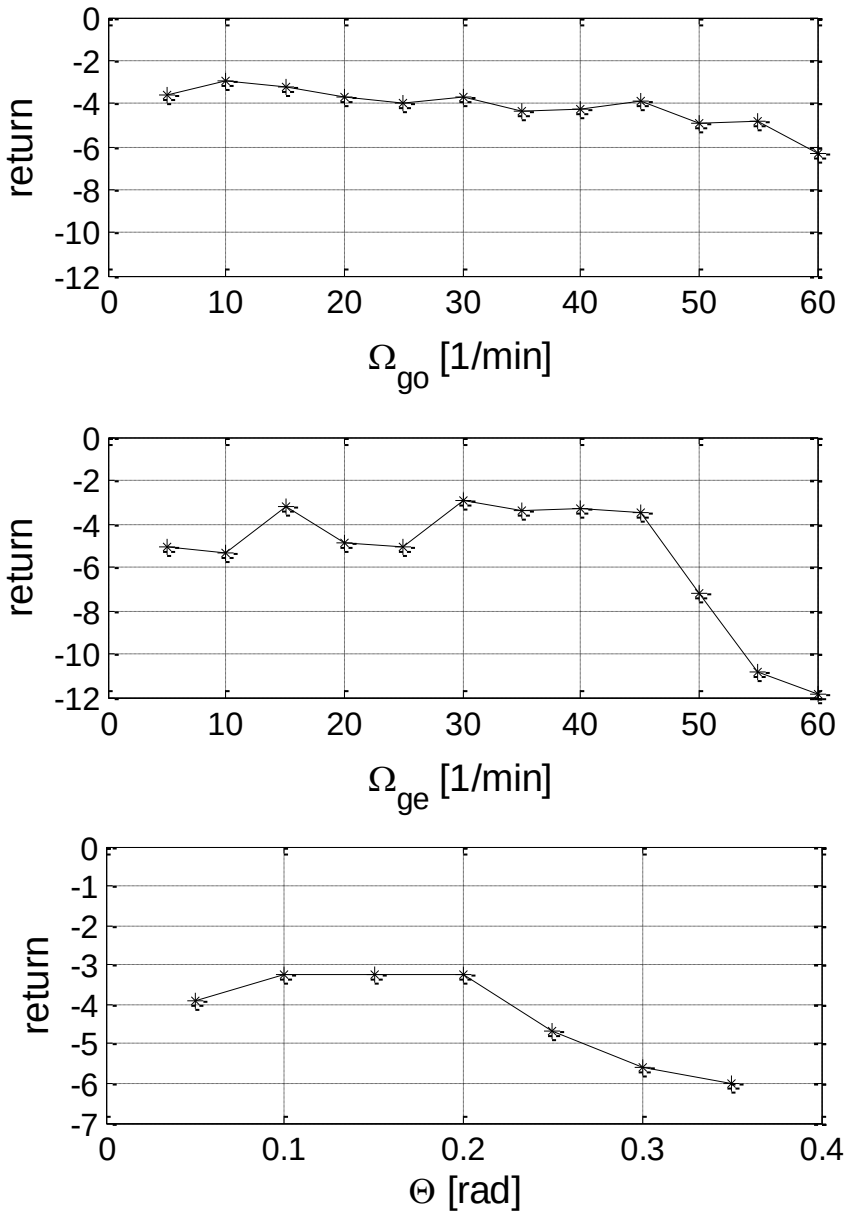

Figure 6.12: Value of the converged return dependent on the size of range increments for the three state variables. Top: gearbox output speed. Middle: gearbox input speed. Bottom: torsion angle.

For Ω_{ge}, a considerable drop in the return is observed for an increment in size of the range $\Omega_{ge} > 40$. For Ω_{go} and Θ, the return drops seemingly proportionally with the increase in size of the incremental ranges, though not as strongly. On the other hand, the best results were achieved for $\Omega_{go}, \Omega_{ge} \approx 15 - 30\ 1/min$ and $\Theta \approx 0.1 - 0.15\ rad$.

Taking into consideration that in a future implementation on a physical environment the resolution of sensors represents a constraint on the maximum smoothness of the discrete mesh (i.e. the smallest possible size of the value ranges of each variable), the values contained in Table 6.1 were selected for the implementation of the homogenously discretized state space.

Range	Value
Ω_{go}	$15\ 1/min$
Ω_{ge}	$15\ 1/min$
Θ	$0.1\ rad$

Table 6.1: Values for the ranges Ω_{go} , Ω_{ge} and Θ

The resulting learning curve is depicted on Fig. 6.9. A greedy episode after convergence can be seen in Fig. 6.13. The convergence is reached after approx. 4500 episodes and judder vibrations are reduced by 85%. The percentage is based on the value of the return of a converged episode in comparison to the return acquired during the standard maneuver introduced in 5.1.2.

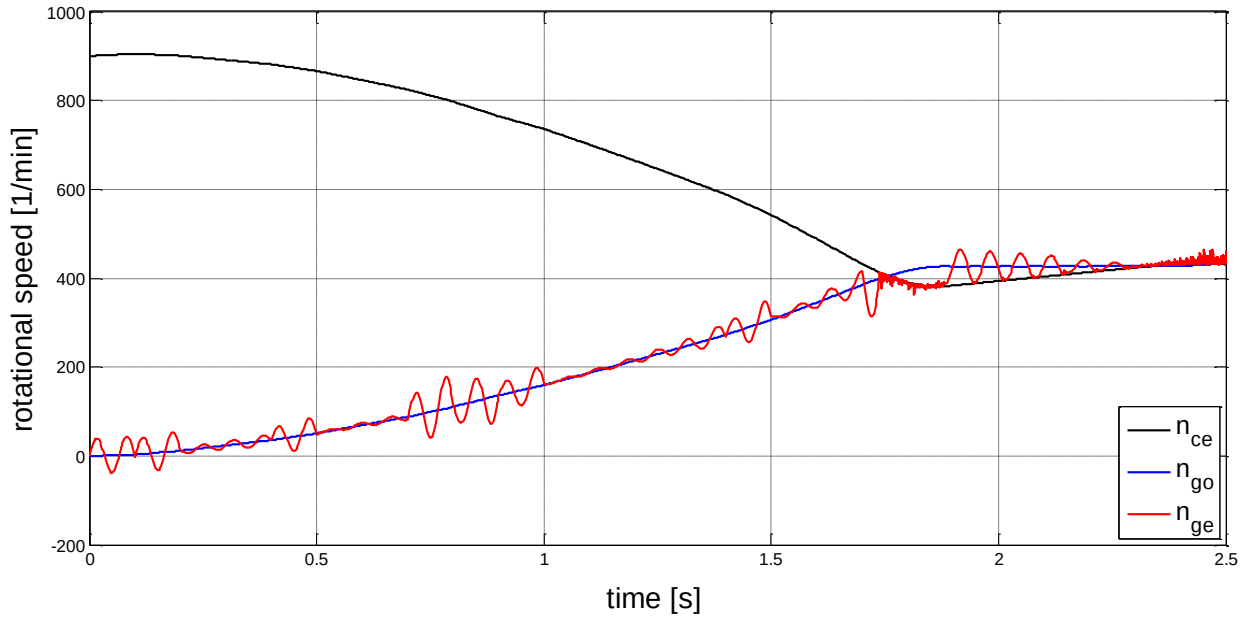

Figure 6.13: Greedy episode with a homogenously discretized state space

As stated previously, this is the best result possible but it requires the maximal learning effort. The next three approaches presented in this chapter aim at the highest possible reduction of effort at the expense of as low a decrease in the solution quality as possible. In the language of RL, the approaches aim at highest possible reduction of the episodes needed for convergence at the expense of as little return as possible.

6.3.2 Dynamically discretized State Space

The first approach for the optimization of the learning task pursues the smart allocation of computing effort in order to downsize the state space, and thus, the solution space.

For this purpose, an initially homogenous, roughly discretized state space is implemented, in which the values for Ω_{go} , Ω_{ge} and Θ are substantially higher than

they were in the previously presented approach. In order to refine the mesh locally, a discrete state is partitioned into smoother elements according to the number of times the agent visits this state. The idea behind this procedure is that providing the agent with a more differentiated perception of the state space around the states it visits more often will lead to an improved behavior in these states, whereas for states seldom visited, the behavior is generalized.

The implementation of this approach requires the definition of three parameters: the number of visits to state required before partitioning it, the initial value for the range of the discrete states of all state variables, and the smoothest value after which no further partition takes place. The values of these parameters were defined so that up to three partitions of a state can take place before reaching the highest smoothness determined in the previous section. The values can be taken from Table 6.2.

State variable	Visits before partition	Max. Range	Min. Range
Ω_{go}	5	$120\ 1/min$	$15\ 1/min$
Ω_{ge}	5	$120\ 1/min$	$15\ 1/min$
Θ	5	$0.8\ 1/min$	$0.1\ rad$

Table 6.2: Parameters for the dynamically discretized state space

After each partition, the range is halved in value for each variable. A partitioned state is called parent state and the resulting states are its children. A single discrete parent state is thus divided into 4 children states after the first, into 16 after the second and into 64 after the third and last possible partition. Therefore, it takes three partitions, or 15 visits to a state for the dynamically discretized state space to locally reach the smoothness of the homogenous state space.

An example for a two-dimensional dynamically discretized state space where the number of visits before partition is set to two is depicted in Fig. 6.14.

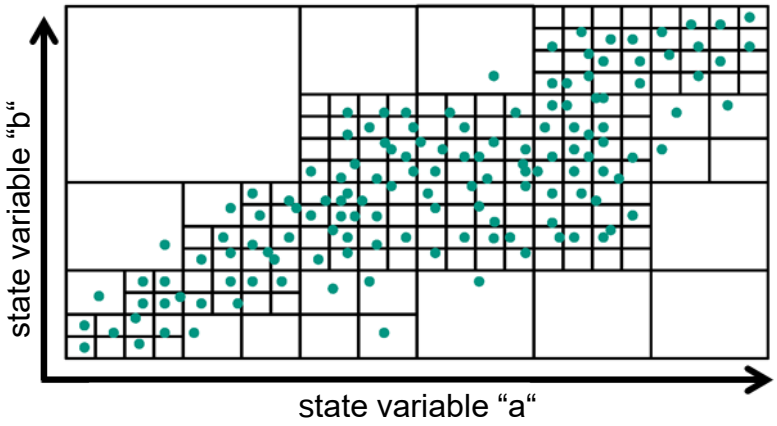

Figure 6.14: Exemplary dynamically discretized two-dimensional state space. The green dots represent a continuous state, whereas the grid represents the discrete state it is assigned. Rougher cells in the mesh are partitioned after the second visit.

The result of the implementation of the dynamically discretized state space can be taken from Fig. 6.15 und Fig. 6.16.

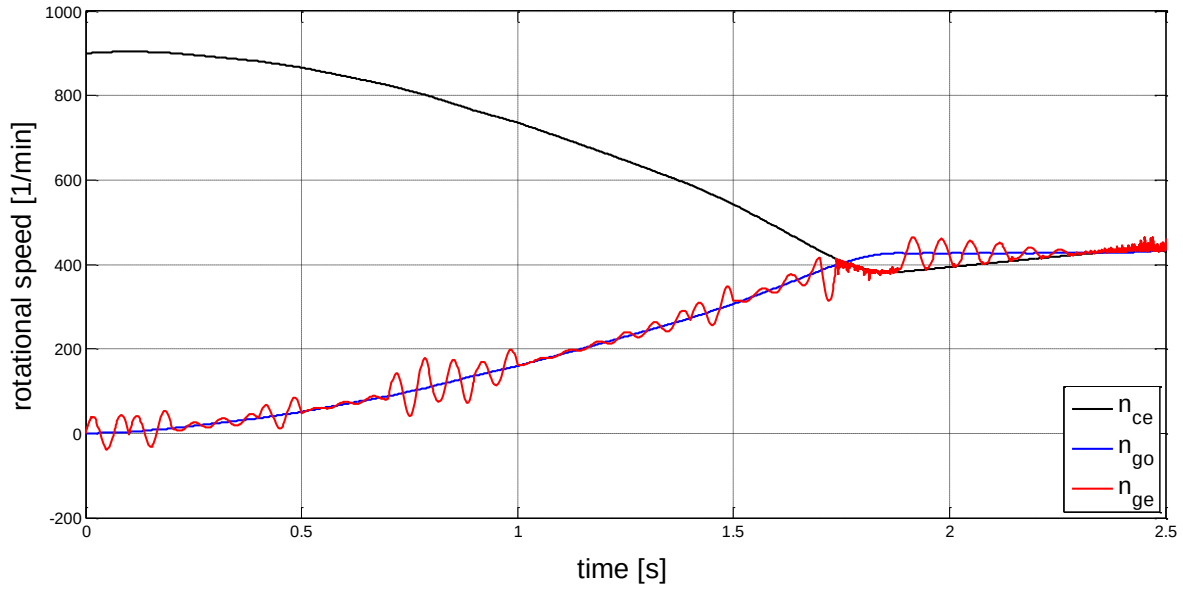

Figure 6.15: Greedy episode in the dynamically discretized state space implementation after convergence

The new setup converges to the exact same solution, where judder vibrations could also be reduced by 85% of the standard synchronization maneuver. Some minor differences could be observed after synchronization was achieved. However, due to the simplifications applied to the simulation model they are not relevant.

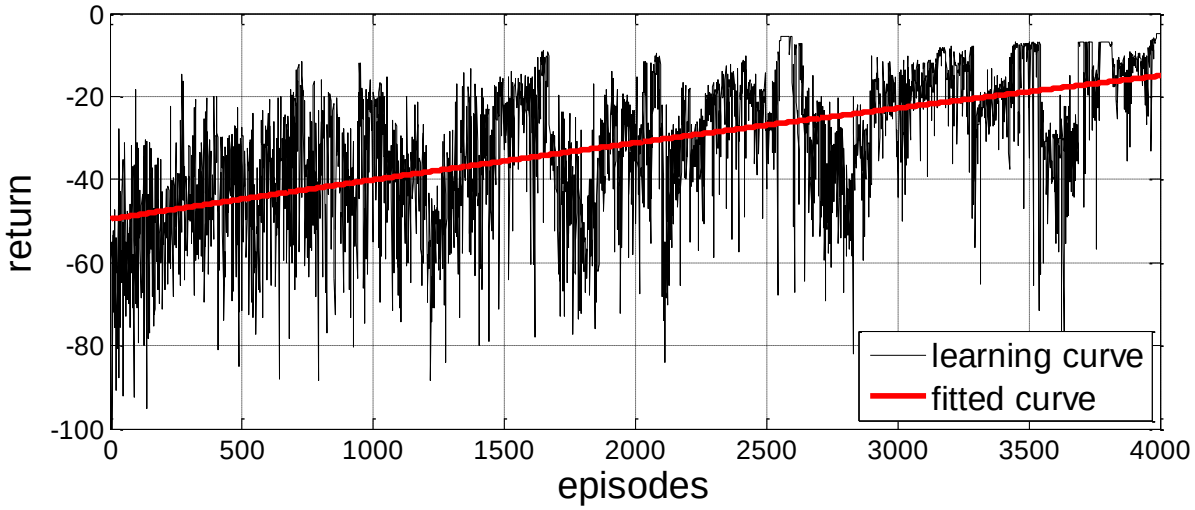

Figure 6.16: Learning curve for the dynamically discretized state space implementation

The main advantage of the dynamic discretization model is observable in the learning curve. Convergence of the learning process is faster, achieved 500 episodes earlier than under the homogenous discretization and evidenced by the steeper progression of the polynomial fitting of the curve.

An optimization of the parameters for the dynamic discretization of the state space bears the potential to deliver better results.

6.3.3 Approximation with Radial Basis Functions (RBF)

As was mentioned in earlier chapters, the clutch synchronization and judder reduction tasks are of a continuous nature. Therefore, the discretization of both the state-signal of the environment and the actions the agent can select necessarily leads to a loss in the Markov property when the tasks are formulated as MDPs. However, the Markov property cannot be fully maintained even without discretization, since the sensors and actuators also sample and actuate the environment in discrete intervals.

An approach to limit this loss of information relies on the approximation of the value function in a continuous state space or state-action space. There are numerous approaches to achieve this that range from relatively simple to sophisticated and complex approximation methods. A detailed overview along with simulated examples of such methods can be taken from BUSONIU ET AL.[206].

In the context of this work, a solution is proposed in which the torsion angle remains a discrete variable and for each discrete angle the value function in the continuous two dimensional space spanned by the two speed variables is approximated. Thus, the

[206] Busoniu et al. 2010

state space is a semi-continuous space. The selected method is the approximation of the three dimensional function (input speed, output speed and corresponding Q-value) with radial basis functions (RBF). Hence, a brief theoretical introduction of the method after ZELL[207] is introduced.

A RBF is a radially symmetrical function whose value is dependent on the distance from its center c, where $h(x,c) = h(\|x-c\|)$. Its norm $\|\cdot\|$ is commonly defined as the Euclidean distance, although other distances are possible. A special form of RBF is the Gaussian RBF selected for this work:

$$h(x) = e^{-(a\|x-c\|)^2} \qquad \textit{Eq. 6.16}$$

where a is a free parameter to be calibrated. A function $f: \mathbb{R}^n \to \mathbb{R}$ as a mapping from $\mathbb{R}^n$ to $\mathbb{R}$ is given by N nodes. Each node i consists of an n-dimensional point $X_i = (x_{i1}, x_{i2}, \dots, x_{in})$ and a corresponding real value $y_i \in \mathbb{R}$. The interpolation rule for all nodes $i = 1,2,\dots N$ yields:

$$f(X_i) = y_i \qquad \textit{Eq. 6.17}$$

Furthermore, the function f can be approximated using K RBFs as follows:

$$f(X) \approx \sum_{j=1}^{K} w_j * h(\|X - C_j\|) \qquad \textit{Eq. 6.18}$$

where C_j is the center of the j-th RBF. The weights w_j can be calculated by solving a linear system of equations after introducing the weights and value vectors W and Y and the Matrix H, where:

$$C = \begin{bmatrix} c_1 \\ \dots \\ c_j \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ \dots \\ y_N \end{bmatrix}, H = \begin{bmatrix} h(\|X_1 - W_1\|) & \dots & h(\|X_1 - W_j\|) \\ \vdots & \ddots & \vdots \\ h(\|X_N - W_1\|) & \cdots & h(\|X_N - W_j\|) \end{bmatrix} \qquad \textit{Eq. 6.19}$$

Hence, the interpolation problem can be formulated as follows:

$$H \cdot C = Y \qquad \textit{Eq. 6.20}$$

The system of N equations and K unknowns can be solved for $N = K$ with the help of the inverse of the Matrix H:

$$C = Y \cdot H^{-1} \qquad \textit{Eq. 6.21}$$

[207] Zell 1997

However, the common case is that of $N \neq K$. This over-determined system of equations cannot be solved exactly. Nevertheless, the best fit around the nodes can be computed with the pseudoinverse of H and the least square method:

$$C = Y \cdot (H^T H)^{-1} H^T \qquad \textit{Eq. 6.22}$$

Having introduced the approximation method, the location of the nodes for the task at hand is yet to be determined. One possibility is to consider the state resulting for an interaction as the position of a node. Each new state visited results in a new node. However, if two nodes are too close to each other, a considerable interpolation error due to singularities is to be expected. This case is exemplified in Fig. 6.17.

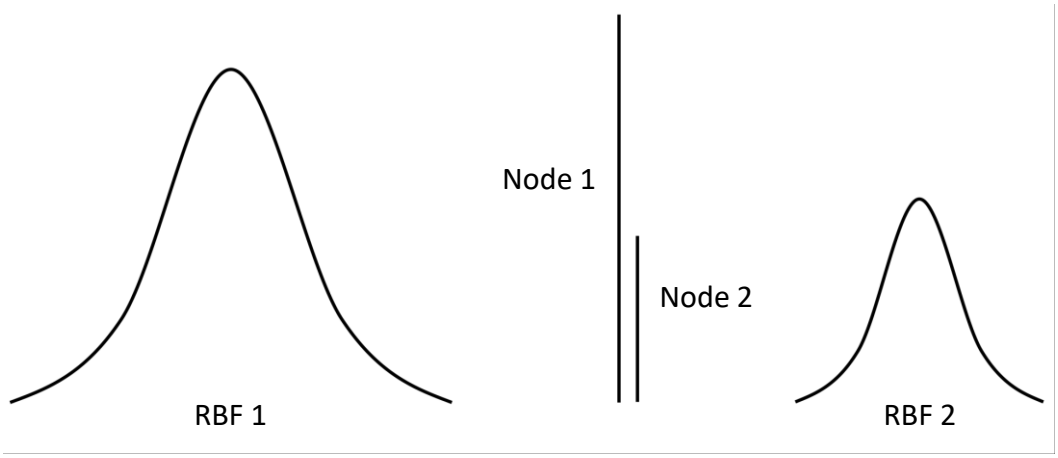

Figure 6.17: One dimensional singularity in the RBF interpolation mesh due to close nodes

The RBFs assigned to each Node 1 and Node 2 cannot be interpolated appropriately due the closeness of the nodes. For the task at hand involving strictly negative values of the value function, false positives result from the interpolation attempt. The nature of the mechanical problem rules out the appearance of positive values, hence positive rewards, since the lowest possible magnitude of the reward is zero. An example of the appearance of false positives as a result of interpolation errors after one and after 10 episodes is depicted in Fig. 6.18.

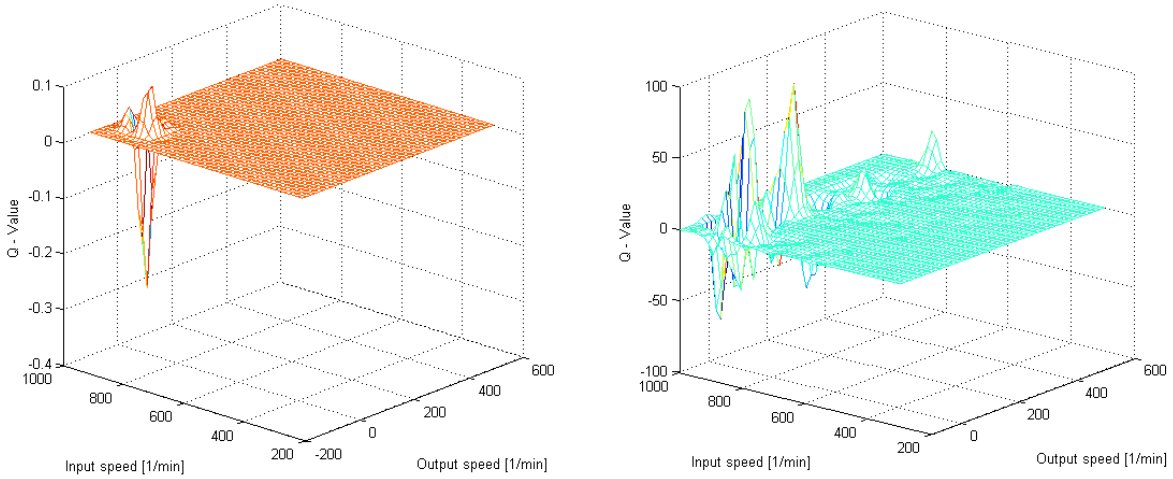

Figure 6.18: False positive Q-Values as a result of interpolation error after one (left) and ten episodes (right)

Initially, the interpolation error is relatively small. However, given the false positive values are higher than any value the agent can achieve, their effect is self-reinforcing and their value rises quickly since the agent perceives actions that lead to higher false positive values as the best. Thus, the agent learns to maximize the interpolation error instead of minimizing clutch judder.

A simple way to avoid interpolation error pursued in this work consists of defining fixed homogeneous meshes of both nodes and RBFs. The number of RBFs is smaller than the number of nodes in order to reduce the computing effort. A newly visited state is assigned a node and the values of the RBFs using their pseudoinverse are computed accordingly. A smoother mesh of nodes reduces the errors due to assignment, but increases the risk of interpolation errors, and thus, of false positive Q-Values. The following grid values were determined empirically for the node and RBF mesh, respectively.

	Nodes	**RBFs**
n_{ge}	15 $1/min$	45 $1/min$
n_{go}	15 $1/min$	45 $1/min$

Table 6.3: Grid values for the node and RBF meshes

After implementing the suggested measure, the appearance of false positive values in the approximated value function can be almost entirely suppressed, as can be taken from Fig. 6.19.

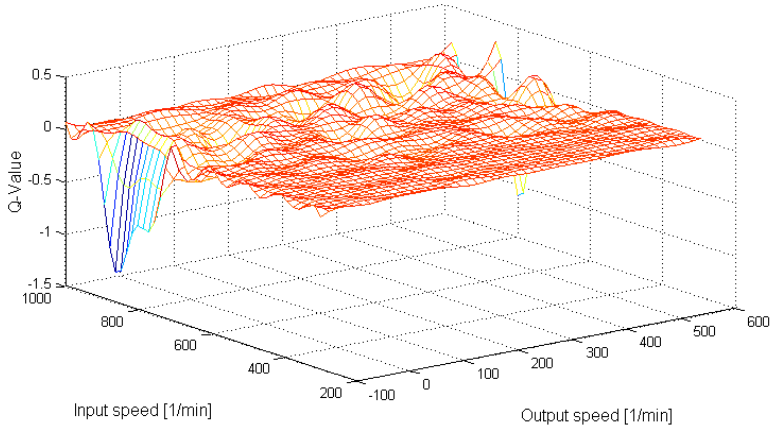

Figure 6.19: Suppressed false positives in the approximated value function through the implementation of fixed meshes of nodes and RBFs

Finally, a last step in order to reduce the effect of the approximation error is introduced. It consists of the introduction of a reference value $Q_{ref}(s_i^*)$ for every node s_i^* in the mesh in order to stabilize its approximated value $\hat{Q}_{ref}(s_i^*)$. The reference value is defined as the peak value of an RBF and it is intended as a means to limit the propagation of approximation errors along the mesh of RBFs. Theoretically, all RBFs are updated as a result of an agent-environment interaction. However, the further away the RBF is from the node to which the continuous state was assigned, the greater the approximation error becomes. For this reason, only the reference value of the closest RBF to the node currently active is considered for the approximation.

Initialize $\mathrm{Q_{ref}}(s^*, a), C$ arbitrarily

Repeat (for each episode)

Initialize s , discretize to s^*

Approximate $\hat{Q}$ for s

Choose a from s using policy derived from $\hat{Q}$ (e.g. ϵ-greedy)

Repeat (for each step in episode)

Take action a, observe r, s'

Choose a' from s' using policy derived from $\hat{Q}$ (e.g. ϵ-greedy)

$Q_{ref}(s^*, a) \leftarrow Q_{ref}(s^*, a) + \alpha\left[r + \gamma\hat{Q}\,(s', a') - Q_{ref}(s^*, a)\right]$

Update C from Q_{ref}

$s^* \leftarrow s'^*; a \leftarrow a';$

Until s is terminal

Figure 6.20: Pseudocode for the RBF based approximation of the value function in a semi-continuous state space

Figure 6.20 depicts the pseudocode for the implementation of the RBF approximation of the semi-continuous state space. A greedy episode of the result of the learning process with the presented approximation of the value function in the semi-continuous state space can be taken from Fig 6.21.

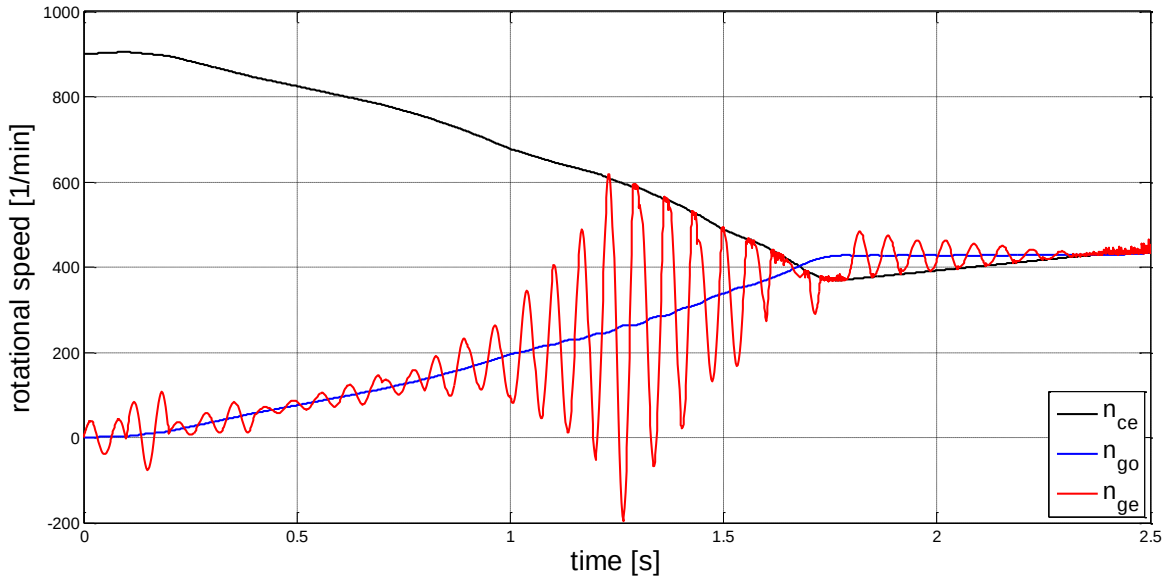

Figure 6.21: Greedy episode of the RBF based approximation of the value function in a semi-continuous state space

It is evident that the judder vibrations are not reduced at the same rate as the previously presented approaches, mainly due to the approximation error. In comparison, the agent in the dynamically discretized state space was able to lead to a reduction of judder of approx. 85%, whereas the reduction with the approach presented in this section only amounts to approx. 42%.

However, the effect of the approximation of the semi-continuous state space on the learning speed of the agent becomes evident upon observation of the learning curve depicted in Fig. 6.22.

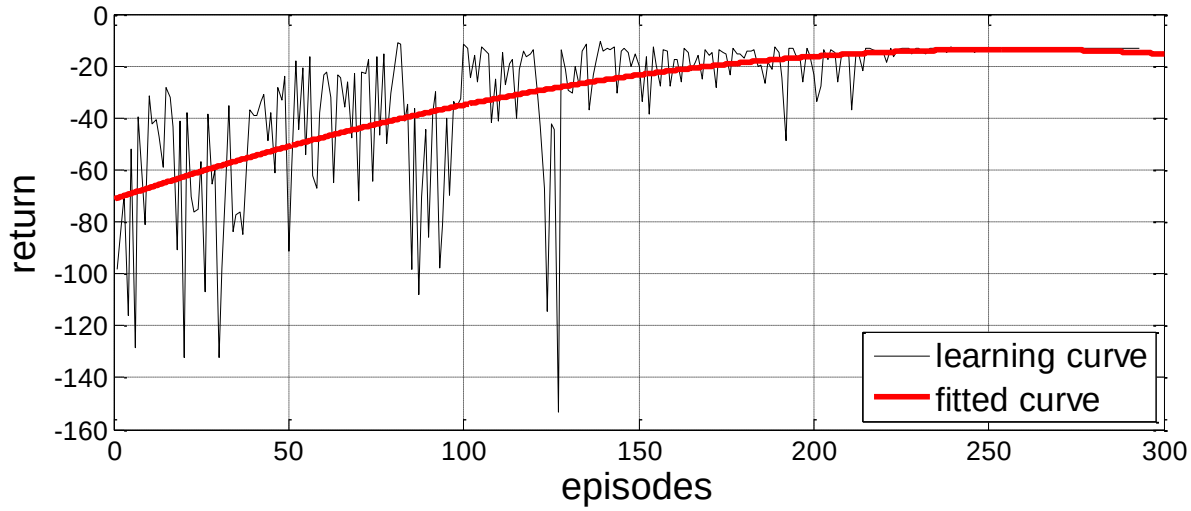

Figure 6.22: Learning curve of the RBF based approximation of the value function in a semi-continuous state space

The approximation of the continuous two-dimensional spaces for each of the discrete torque angles has a generalizing effect, in which an action is assumed to have a similar value for states close to that which has been recently visited. Thus, the speed of the learning process is boosted considerably. Convergence of the return is achieved after only approx. 250 episodes, which is only approx. 5.6% of the episodes required for convergence with the fastest approach presented thus far. Also, the fluctuation of the return is greatly reduced as is evident from the smoother course of the learning curve. This is due to the approximation of the value for actions that the agent had not taken before. In previous approaches, the agent would encounter new states even at late stages of the learning process, which caused him to take “wild guesses” regarding the best course of action, therefore causing considerable fluctuations of the return. The approximation of the value of such actions provides the agent with a choice that is less likely to be counterproductive, even if it is not necessarily optimal, thus improving its behavior considerably.

6.3.4 Gaussian-Update Method for the Value Function

The last approach presented in this chapter represents an attempt to realize a compromise between the effectiveness of the judder suppression of the discrete solutions and the generalizing effect of the approximation of the value function in the semi-continuous state space.

In order to achieve this compromise, a new method for the approximation of the value function in entirely discrete state and action spaces is proposed along with a new method to handle the approximation error.

Instead of defining meshes of nodes and RBFs, the new proposed approach shares the discrete homogenous mesh of 6.3.1.

Initially, the interaction between the agent and its environment and the updating of the value function are analogous to the classic RL approach, in which only the value of the state-action pair currently visited is updated. However, in this approach the implementation of a Gaussian filter as a means to approximate the value of nearby states is introduced.

Two parameters are deemed necessary for the successful implementation of the Gaussian filter for the approximation of the value function. The first parameter assesses the probability of a state to have the same value as the state currently visited, and is named $belief$. The probability of neighboring states to have the same value as the state currently visited is considered higher than that of farther states. In this work the measure for the distance between two states consists of the number of states in the discrete mesh lying between the state s currently visited and the state s_i whose value is currently being approximated. Thus the distance yields $\|s_i - s\|$. An example for the calculation of the distance in a simplified two-dimensional example can be taken from Fig. 6.23.

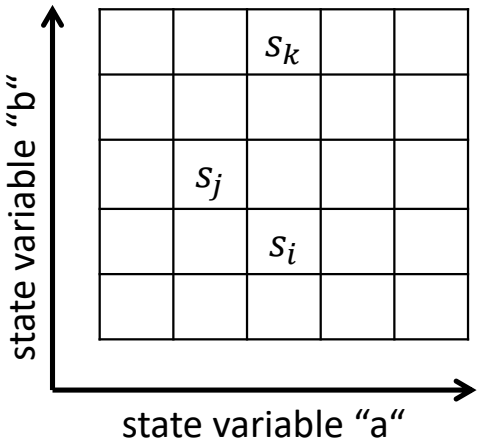

Figure 6.23: In this two-dimensional grid each cell represents a state. The distance between states for the exemplified states yields: $\|s_i - s_j\| = \sqrt{2}$ and $\|s_i - s_k\| = 3$.

The value of $belief$ for a state s_i decreases following a Gauss-function with increasing distance to state s :

$$belief = e^{-\|s_i - s\|^2 / 2\sigma^2} \qquad \textit{Eq. 6.23}$$

The second factor is introduced as a means to establish the necessity of an approximation. It is deemed more reasonable to trust the values resulting from direct interactions involving a particular state than those of approximated values. The idea behind this parameter is to determine how well "known" the value of a state is, before determining if its value should be updated using information resulting from an approximation. In simpler words, the value of a state that has been visited often should

not be altered as a result of visits to nearby states. Thus, the $need$ factor is introduced and defined as follows:

$$need = \begin{cases} 1 - sig[v(s_i) - k], & for\ s_i \neq s \\ 1, & for\ s_i = s \end{cases} \qquad \textit{Eq. 6.24}$$

where $v(s_i)$ is the number of visits to s_i and k is an empirical parameter that determines after how many visits the value of a state can be considered to be free of error. A graphical exemplification of the $need$ factor is contained in Fig. 6.24.

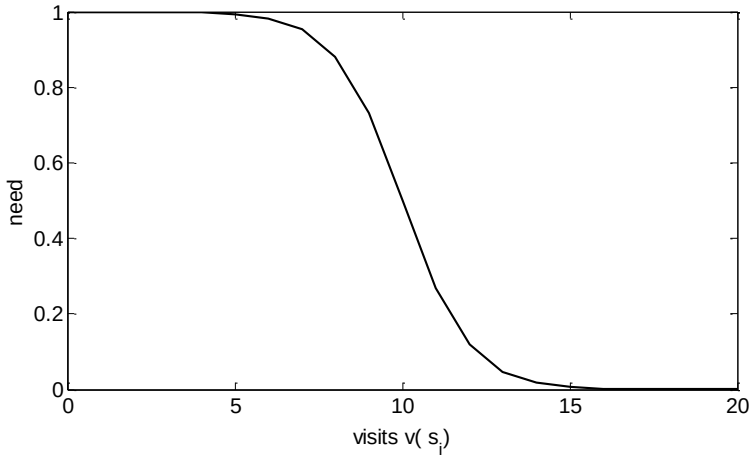

Figure 6.24: Example for the parameter $need$. In this example $k = 10$, thus the value of the state after 5 visits is considered to be faulty enough to be influenced by the approximation as a result of visits to nearby states, whereas the values remain almost unaffected by approximation after they have been visited around 15 times.

Thus far, only the approximation of nearby states has been addressed. However, this approach can be easily extended to approximate the space spanned by states and actions. Instead, the approximation takes place for nearby state-action pairs.

The SARSA-algorithm for a RL problem with three state variables x, y and z and one action variable a, can be reformulated in order to perform the proposed Gaussian filter approximation. The introduced factors for all $s_i = (X_i, Y_i, Z_i) \in \mathcal{S}$ and all $a_i \in \mathcal{A}$ after a visit to the state-action pair (s, a) yield:

$$belief(s_i, a_i) = e^{-\|X_i - X\|^2 / 2\sigma_x^2} \cdot e^{-\|Y_i - Y\|^2 / 2\sigma_y^2} \cdot e^{-\|Z_i - Z\|^2 / 2\sigma_z^2} \cdot e^{-\|a_i - a\|^2 / 2\sigma_a^2} \qquad \textit{Eq. 6.25}$$

$$need = \begin{cases} 1 - sig[v(s_i, a_i) - k], & for\ (s_i, a_i) \neq (s, a) \\ 1, & for\ (s_i, a_i) = (s, a) \end{cases} \qquad \textit{Eq. 6.26}$$

The value $\hat{Q}(s, a)$ for the transition from the state-action pair (s, a) to the pair (s', a') is calculated following the SARSA-update rule:

$$\hat{Q}(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma Q(s', a') - Q(s, a)] \qquad \textit{Eq. 6.27}$$

This value is then used for the approximation of the values $Q(s_i, a_i)$ of all $s_i \in \mathcal{S}$ and $a_i \in \mathcal{A}$ with the following Gaussian-update rule:

$$Q(s_i, a_i) \leftarrow [1 - need(s_i, a_i) \cdot belief(s_i, a_i)] \cdot Q(s_i, a_i) + need(s_i, a_i) \cdot belief(s_i, a_i) \cdot \hat{Q} \qquad \textit{Eq. 6.28}$$

Therefore, an update of all state-action pairs takes place after each interaction according to the presented rules, even though some states are barely affected according to the distance to the state currently visited.

An example of the value function update procedure for the classic SARSA-update and the Gaussian-update approach are depicted in Fig. 6.25 and 6.26 accordingly.

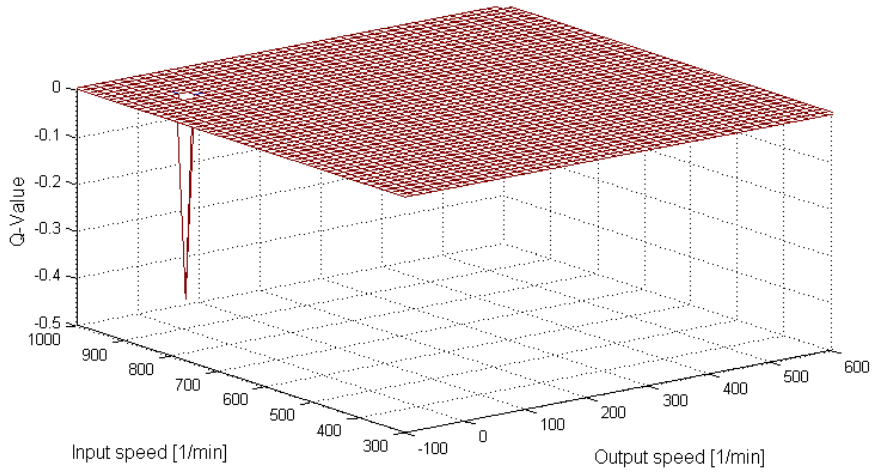

Figure 6.25: Classic SARSA-update of the value function in an exemplary two-dimensional state space

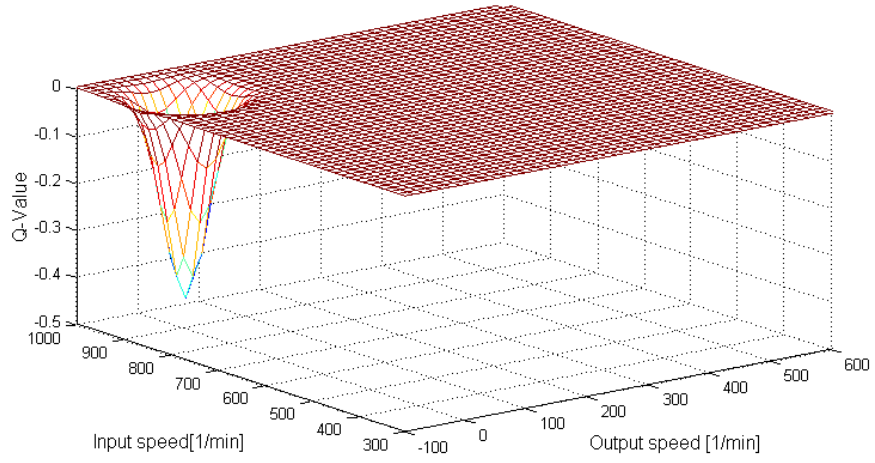

Figure 6.26: Gaussian-update of the value function in an exemplary two-dimensional state space

The Gaussian-update method for the approximation of the value function was implemented for the clutch judder reduction task using the discrete homogeneous mesh of 6.3.1 and with the empirically determined parameters contained in Table 6.4.

Parameter	Value	Used for
$\sigma_{n_{ge}}$	2	$belief$
$\sigma_{n_{go}}$	2	$belief$
σ_{θ}	0.4	$belief$
σ_{a}	0.4	$belief$
k	10	$need$

Table 6.4: Parameters of the Gaussian-update method for the clutch judder reduction task

A greedy episode of the learning process and the resulting learning curve are depicted in Fig.6.27 and Fig 6.28, respectively.

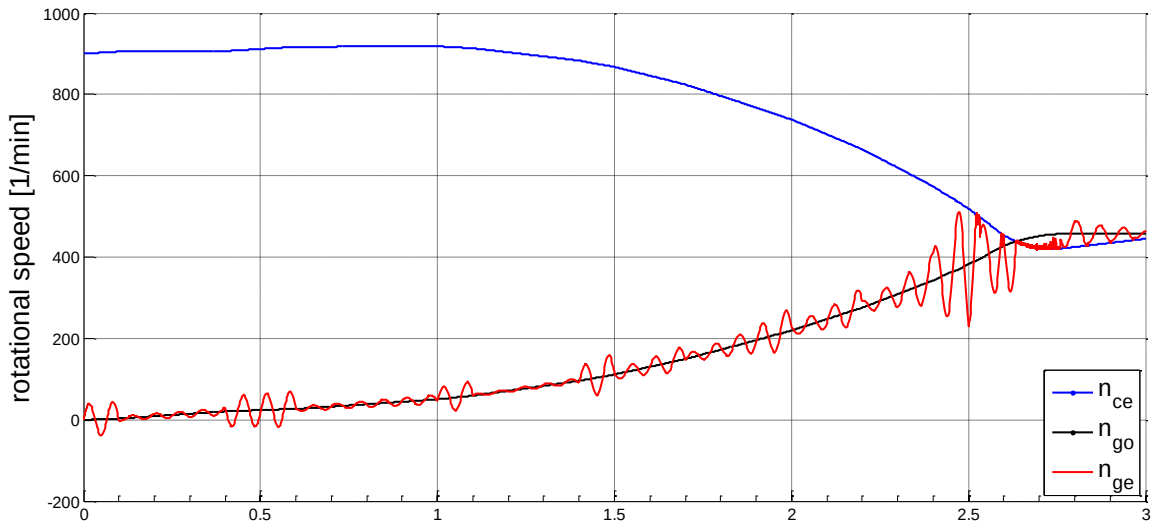

Figure 6.27: Greedy episode of the SARSA-algorithm with implemented Gaussian-update method

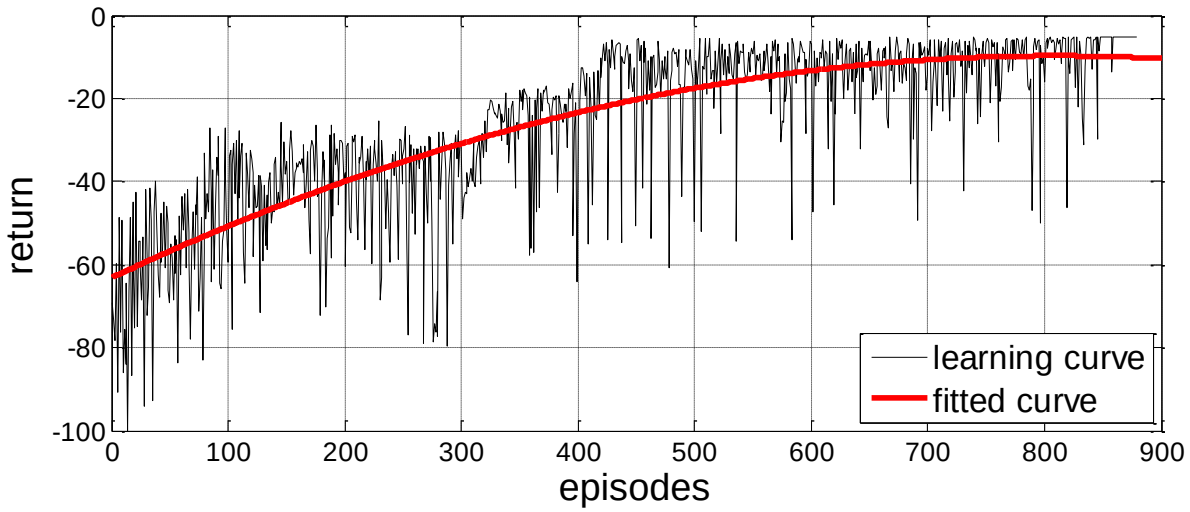

Figure 6.28: Learning curve of the SARSA-algorithm with implemented Gaussian-update method

The result of the implementation of the Gaussian-update method is the realization of a compromise between the enhancement in learning speed through the implementation of approximation methods and the effectiveness of the solutions involving a homogenously discretized state space. Through the implementation of the $belief$ and $need$ factors, the approximation of values that are not currently being visited becomes more selective.

The SARSA-algorithm with the Gaussian-update method converges approx. five times faster (<900 episodes) than the classic approach in the same homogenously discretized state space suppressing 80% of judder vibrations; a mere 5% less than the classic approach. Although the approach presented in this section needs four times the number of episodes to converge than the method involving the approximation of the value function in a semi-continuous state space, it is 38% more effective. However, a negative aspect of the implementation is the extended time required for synchronization. At approx. $3\ s$ it is comparatively slow and takes around 20% longer to achieve synchronization. A comparison of the relationship between the learning speed and the effectiveness of judder suppression of the four approaches presented in this chapter is contained in Table 6.5.

Approach	Time to convergence[208]	Effectiveness of suppression[209]
Standard approach	100%	85%
Dynamic discretization approach	88.9%	85%
Function approx. in semi-continuous state space	0.56%	42%
Gaussian-update approach	18%	80%

Table 6.5: Learning speed and effectiveness of judder suppression of the presented approaches

Furthermore, it is worth mentioning that the results presented in this work were achieved using parameter values that resulted from short empirical study. A comprehensive parameter study and optimization bears the potential to yield even better results.

6.4 Adaptivity and Robustness

As a final element of this chapter, an analysis of the performance of the RL algorithm to fluctuations and permanent changes in its environment is observed.

Perceived fluctuations in system parameters in physical environments are always present to a certain extent. This is due to actual changes in environmental parameters such as temperature, humidity or to fluctuations in the measured values of parameters within the tolerances of sensor devices. On the other hand, physical environments tend to experience permanent changes as a result of deterioration and wear in their components. In this section, a RL agent is presented with different scenarios intended to reproduce each of the mentioned changes to its environment in order to analyze its behavior.

The agent implemented for this purpose uses the SARSA-algorithm with the Gaussian-update method introduced in 6.3.4 to estimate the value function.

In the first scenario, the agent has already completed the learning task with the standard parameters of the drive train model. Afterwards, a fluctuation of the mass of the vehicle, e.g. as a result of more passengers or a load being transported, and thus of its inertia J_V, within $\pm 10\%$ of its original value is simulated. As long as the fluctuation of one parameter is simulated, all other parameters are held at a constant value. The agent then performs ten greedy runs, i.e. without taking random actions, with a

[208] The duration of the standard SARSA-agent in the homogenously discretized state space is used as benchmark.

[209] As stated previously, judder is quantified in terms of the return computed after a synchronization maneuver. The return for a standard engagement without RL-control is regarded to generate 100% judder and serves as base of comparison.

fluctuating mass and otherwise constant system parameters. The results of the performance of the agent under the described circumstances can be taken from Fig. 6.29.

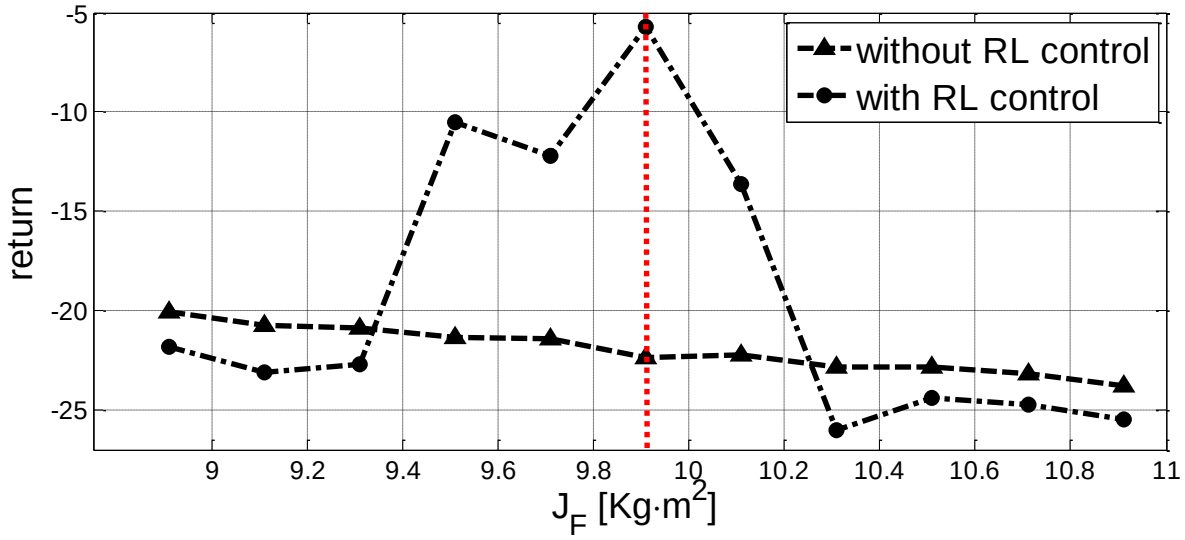

Figure 6.29: Performance of the RL-agent under fluctuating vehicle mass J_V in comparison to actuation with a standard force ramp without RL-control. The original value of J_V is marked in red.

It can be observed that the agent is able to reduce judder vibrations in comparison to a standard engagement ramp only when the fluctuations are relatively small and the inertia of the vehicle does not change substantially. The greater the offset to the original value, the less effective the agent is at suppressing the vibrations. In fact, for larger deviations from the standard value the RL-control is actually counter-productive.

A similar experiment is performed for a decreasing static friction coefficient μ_{st}, e.g. as a result of oil or dirt in the friction pairing. Again, an agent who has already completed a learning task with the standard parameters performs four greedy runs in an environment where the static friction has decreased. The result of the experiment can be observed in Fig. 6.30.

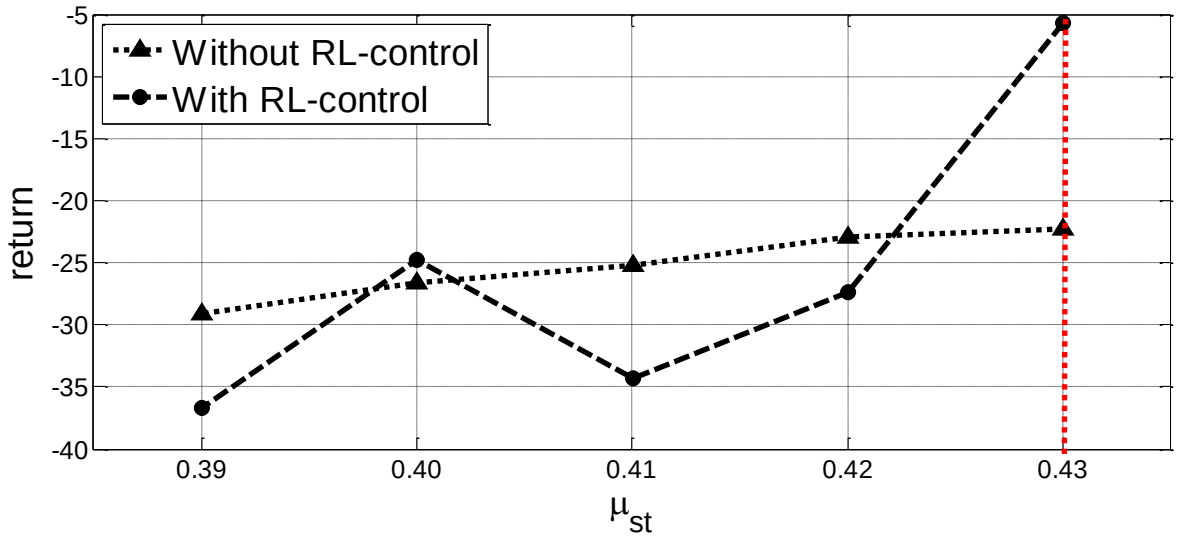

Figure 6.30: Performance of the RL-agent for a decreasing static friction coefficient μ_{st}. The original value of μ_{st} is marked in red.

The loss in effectiveness of judder suppression by the agent is observable. As was the case in the previous experiment, the greater the deviation of the original value of the fluctuating parameter, the less effective the agent becomes. In fact, the RL-control is likely to actually cause judder instead of reducing it, when the value it had learned is altered. It should be mentioned that the presence of e.g. oil in the friction pairing usually leads to a much more dramatic decrease of the friction coefficient, as considered for this experiment.

The second scenario regards the case in which the changes to the environment are permanent. Again, changes in the mass of the vehicle are considered first. A considerable permanent change in the mass of the vehicle is rather unlikely, however, the weight of the vehicle could be notably higher as a result of more passengers over an extended period of time, which could cause a more difficult or extended start up. Another option would consider the use of an existing implementation of the RL-controller for another type of vehicle with a different mass inertia. Therefore, for the purpose of this investigation the ability of the RL-agent to learn a successful strategy for the altered weight is analyzed. The result of the adaptivity experiment is contained in Fig. 6.31.

The agent retains its ability to learn to suppress judder vibrations under altered vehicle inertia. It is worth noting that both the number of episodes to reach new convergence and the achieved return after it is reached remains almost unaffected with a decrease in the inertia, whereas even a relatively small increase affects both negatively.

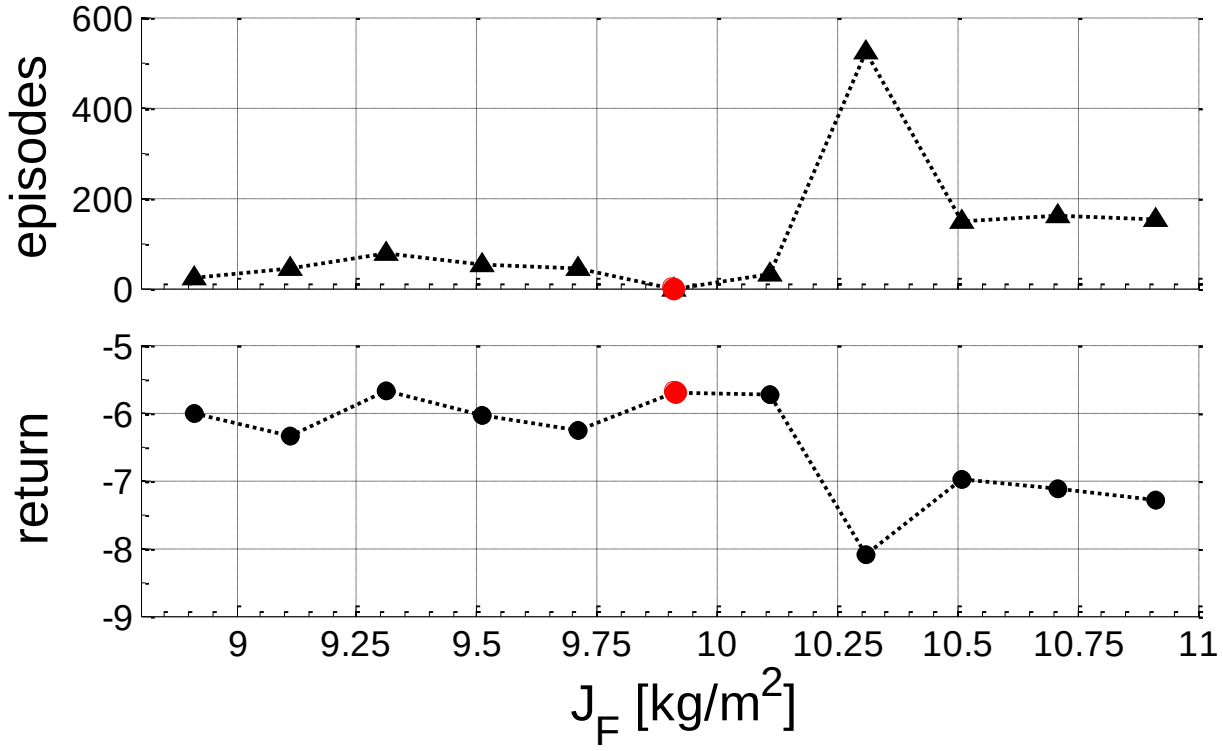

Figure 6.31: Adaptation of the RL-agent to a permanent change in the inertia J_m of the vehicle. Top: episodes to reach convergence in second learning process. Bottom: Achieved return after convergence of second learning process. The return of original learning process is marked in red.

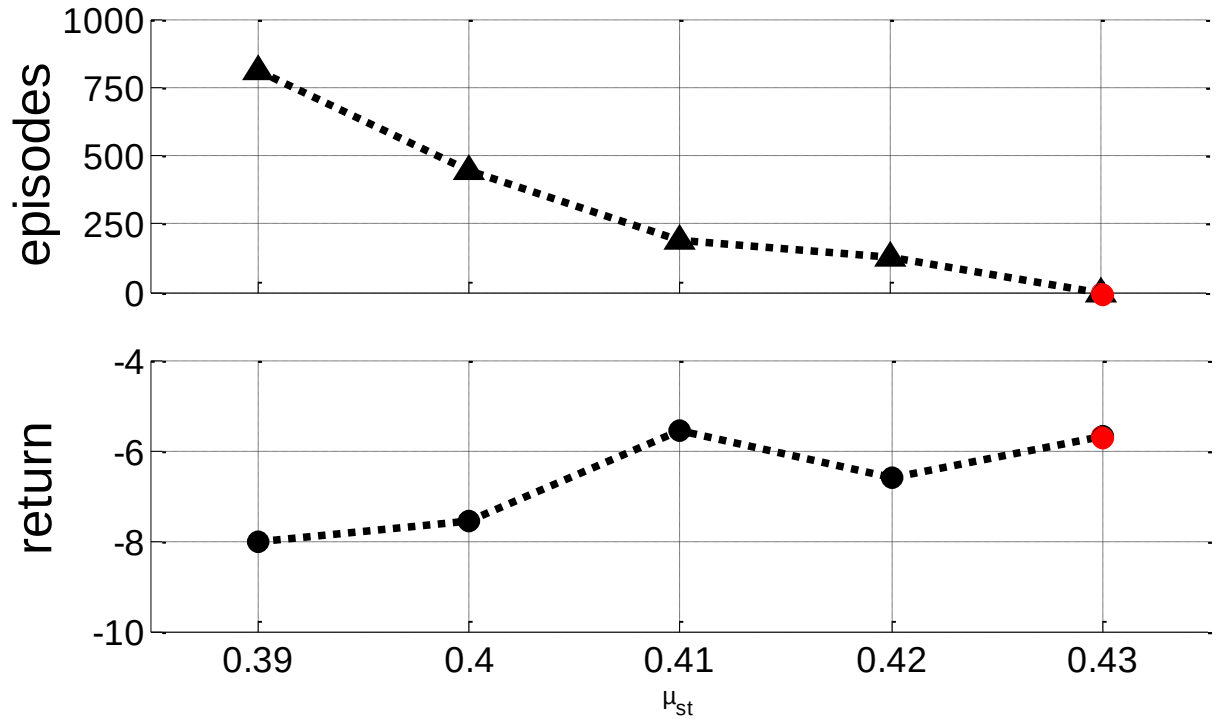

Figure 6.32: Adaption of the RL-agent to a permanent change in the static friction coefficient μ_{st} of the vehicle. Top: episodes to reach convergence in second learning process. Bottom: Achieved return after convergence of second learning process. The return of original learning process is marked in red.

Lastly, the RL-agent is confronted with a permanently altered static friction coefficient, e.g. as a result of wear. The results of the experiment are contained in Fig. 6.32. The agent is able to learn a new strategy to suppress judder vibrations. However, the effectiveness of the algorithm is reduced by lower friction coefficients.

In summary, the ability of the agent to suppress judder vibrations under fluctuations and permanent changes in the environment is retained. However, its effectiveness is reduced for both.

In the first scenario, fluctuations of system parameters lead to a temporary loss of the correct mapping of actions to states. Unless these fluctuations are accounted for, e.g. through an adequate model, it represents a partial loss of the Markov property. The smaller the fluctuations, the “more Markov” the environment becomes and the better the performance of the agent remains.

In the second scenario, permanent changes to the environment do not prevent the agent from learning a new strategy to reduce judder vibrations. However, the effectiveness of the new strategy and the additional effort to learn it can suffer considerably. A possible explanation lies in the change in the vibration behavior of the system and the need for an adaptation of the RL framework to these changes. For example, a change in the mass of the vehicle leads to a change in its eigenfrequency, which might make it necessary to adjust the rate at which the agent sets actions in order to counteract judder vibrations.

Therefore, it is of great importance that fluctuations and changes in the environment are avoided as far as possible for the agent to learn an effective strategy in as few episodes as possible. The importance of this requirement becomes more evident in the following chapter, when the RL framework is applied to a physical environment in which the Markov property can never be fully ensured.

7 Implementation of the RL Framework on the IPEK Mini Hardware-in-the-Loop scaled down Test Bench

The IPEK Mini-Hardware-in-the-Loop test bench (Mini-HiL) is designed to provide a scaled down experimental environment for drive trains and drive train components. Due to its flexible architecture, different levels of the XiL-framework introduced in 2.3 can be realized in order to provide an adequate physical model of the drive train with the desired levels of partitioning, maturity and abstraction.[210]

In this chapter, the conception of the physical test bench and a simulation model of it as new environments for the RL framework are presented. Afterwards, the Gaussian-update algorithm is implemented in this new environment and the results are discussed. Furthermore, new RL-algorithms are introduced for this new environment, in order to assess if they are better suited for the implementation on the physical test bench. After a discussion of the results of all implemented approaches, the one considered the most promising is implemented on the physical test bench and the results are discussed.

7.1 Description of the Environment

In this section the configuration and architecture of the Mini-HiL test bench for the purpose of clutch judder studies is presented according to GWOSCH[211].

In general, the Mini-HiL consists of a machine bed on which different physical models can be assembled and two highly dynamic direct current motors (DC motors) with which the different input and output speeds of components or subsystems can be reproduced. The test bench consists of a real time system for the control of the DC motors and the processing of different input and output signals through a modular interface system (MIS). The architecture is completed by a target and a host pc on which the different test and control programs can be generated and executed.

In the next section, the realization of the physical model of the drive train for the reproduction of judder vibrations is introduced.

7.1.1 Physical Drive Train Model

The physical drive train model used throughout this investigation corresponds to a scaled down reduced drive train of an Opel Corsa®. The scale down factor is defined so that an energy-equivalent model, where the vibrations lie in the desired judder

[210] cf. 2.3.2

[211] Gwosch 2011 and Gwosch et al. 2013

frequency region, is achieved. Due to restrictions in the geometry and the mechanical properties of the materials, this can only be accomplished through modification of some parameters like the stiffness and strength of the side shaft and the values of the inertia of certain elements.

CAD-schematics of the mechanical configuration of the test bench can be observed in Figure 7.1.

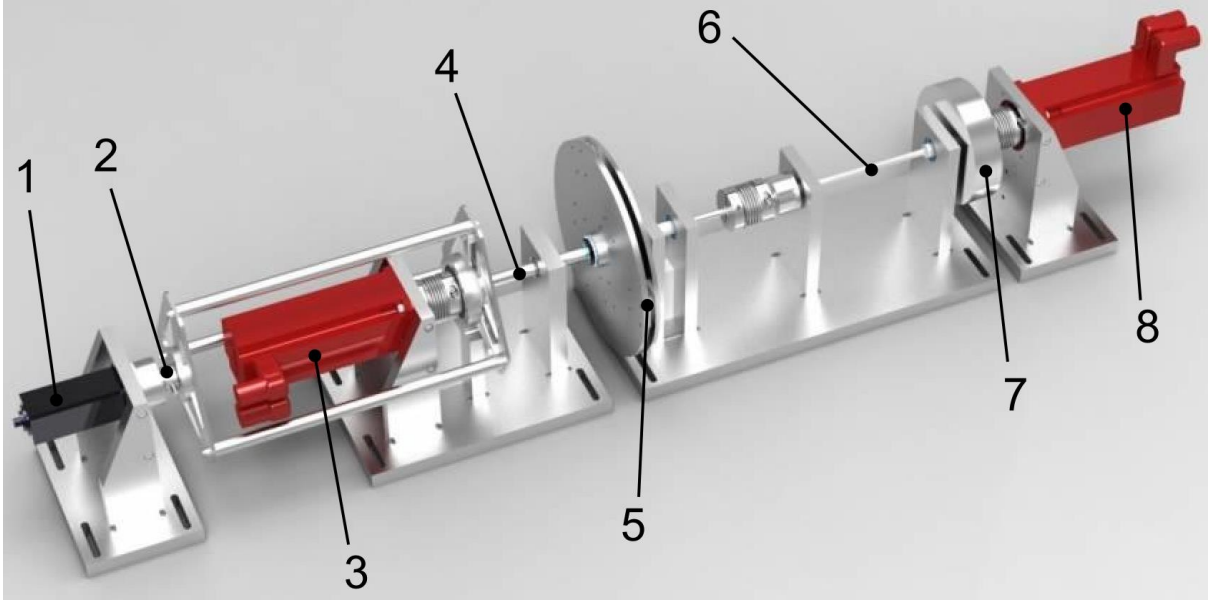

Figure 7.1: Scaled down physical model of the reduced drive train

The components of the model are numbered from left to right. The first component is the linear actuator (1) responsible for the application of the clamping force in the clutch (5). The force is applied through the bar kinematics around the input dc motor (3). The value of the force is measured by the force sensor (2) located before the input DC motor (3) responsible for the reproduction of the combustion engine speed and torque. The latter is measured by a sensor that relies on the magnetorestriction effect of ferromagnetic materials of a permanently magnetized shaft (4). Afterwards, the clutch discs (5) are tested. The friction pads are mounted on the output disc of the clutch. Together with the side shaft (6) and the vehicle mass (7) (modeled as a disc with the equivalent inertia), they form the physical two-mass oscillation model of the drive train. Finally, the output DC motor (8) is found at the right end of Fig. 7.1.

The friction contact in the clutch consists of three pellets of the ceramic material Al_2O_3, whereas the counter surface on the input side is made of the unalloyed steel C45.

The speed of the input and output DC motors of the manufacturer SEW[212] is measured with their integrated resolvers. However, direct measurement of the absolute rotation

[212] SEW-EURODRIVE 2015

angle is not possible with the same device. The third speed of relevance for the purpose of this investigation is that of the clutch output disc, which is measured with an external laser surface vibrometer (LSV).

The clutch linear actuator is conceived as a stepping motor with a spindle that translates the rotational into a linear motion. The integrated rotary encoder enables a resolution of the axial travel path of $1\mu m$. However, the actuator is only able to work in steps of $10\mu m$. Target positions are defined in increments, where $1\ incr. = 1\mu m$.

The control algorithms are programmed in a Matlab/Simulink[213] environment and the real time system ADwin Pro II[214]. The output parameters of the DC motors and the clutch actuator are set via a CAN-bus connection. The input values, thus the speed of the DC motors, are processed through their resolvers but their value is converted into an incremental signal in their inverters, in order to reproduce the sensor signal usually available in commercial drive trains. The incremental signal is converted back to a speed value in the real time program. All relevant sensor input signals are read as analog signals.

The test bench represents the drive train following the XiL-framework introduced in 2.3. The “Vehicle” system is implemented in its Level 1 subsystem layer: the “powertrain-in-the-Loop” layer. The only virtual element of the system “vehicle” consists of the combustion engine, whose output and input signals are computed in real-time. All other elements are modeled as a physical two-mass oscillator, as described in earlier chapters. Thus, the entire output side of the power train is reduced to physical models of the clutch, the side shaft and the vehicle mass.

The system “Environment” is conceived in a manner that would allow an abstract virtual environment consisting of a set of resistances (e.g. air resistance) and a torque load resulting from the vehicle mass and the inclination of the road to be defined. For simplicity reasons no additional loads are defined and thus a virtual environment is not present. The environment of the physical elements of the test bench is ever present but generally neglected.

The system “Driver” consists of a real-time virtual driver that sets a constant speed controller for the engine and a force ramp for the clutch actuator whenever the RL-agent is not active. Furthermore, even though the reward is a key element of the RL framework, it is the driver’s expected behavior to judder vibrations, which was used to define this signal. Thus, the driver is responsible for the feedback that the agent receives regarding judder vibrations.

[213] MathWorks 2015

[214] Jäger Messtechnik GmbH 2015

Finally, the system under development (SuD) consists of the RL framework introduced in earlier chapters. It controls the clutch actuator in real-time in order to optimize the perception of clutch judder through the driver.

The graphical overview of the application of the XiL-framework to the Mini-HiL test bench setup for the investigations in the context of this work can be taken from Figure 7.2.

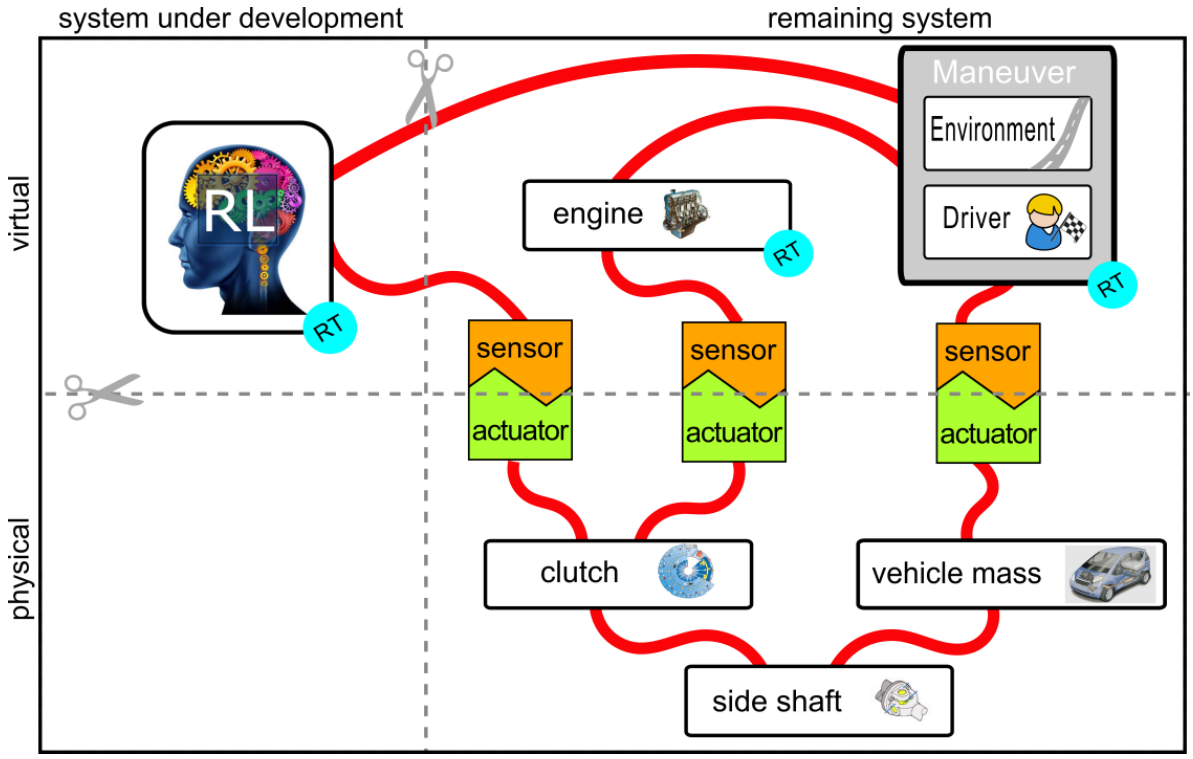

Figure 7.2: Application of XiL-approach to the development of the RL framework for judder suppression on the Mini-HiL test bench setup[215]

7.1.2 Simulation Model of the physical Drive Train

The ultimate goal of this chapter is the implementation of a RL agent on the physical drive train that reduces judder vibrations during the synchronization of the input and output side of the power train. However, there are several reasons for the use of a simulation model before the agent is implemented on the physical test bench.

The first advantage of a simulated environment lies within the time it takes to simulate the test maneuver in comparison to the time it takes to physically reproduce it on the test bench. As a simple example, the duration of an episode consisting of the synchronization maneuver and the following deceleration of the drive train to a full stop takes approx. 10 seconds. A learning task of around 10 000 episodes can therefore be

[215] The clutch, vehicle mass and side shaft are implemented as physical elements. However, it should be noted that they are models of the actual elements, thus they are physical models.

projected to take somewhere between two and a half and three hours. The same learning task can be simulated in approx. 10 minutes.

Another aspect to be considered is the increase in flexibility regarding different experimental configurations gained through the use of a simulated environment. Experimental parameters can easily be adjusted in a simulation model, whereas retooling and even the production of new components is necessary in order to adjust the physical environment to a change in the experimental setup. Furthermore, such changes bear the potential of causing unexpected behavior to arise. Through a preliminary simulation a diagnosis of potential undesired or even dangerous behavior of the physical setup beforehand is possible.

Last but not least, the use of an adequate simulation environment leads to a reduction of the operating time of the test bench, which is a crucial aspect regarding the inherently lengthy RL learning processes. This reduction in operating time is of great importance since it leads to reduced costs of operation, such as wear and energy costs, and more flexibility regarding the available machine time of highly demanded test benches.

In order to model an automotive drive train as a two-mass-oscillator the following assumptions need be made:

- the two branches of the drive train after the differential are perfectly symmetric
- the wheels adhere perfectly to the road (no slipping)
- high stiffnesses can be neglected

Through the assumption of the wheels' perfect adhesion the velocity of the vehicle can be represented as rotational speed and the mass of the vehicle can be represented as a rotational inertia. Neglecting high stiffnesses, e.g. between crank shaft and clutch disc, allows the use of added inertias to form an equivalent reduced model. The procedure of reducing the automotive drive train to a two-mass oscillator is described in several sources[216].

For the purpose of this investigation, the model proposed by GWOSCH ET AL.[217] is enhanced in accordance to SEITENBRECHER[218] in order to describe the drive train in two different modes: slipping-clutch mode and closed-clutch mode.

7.1.2.1 Simulation Model in slipping Clutch Mode

In this section a new simplified model of the reduced drive train is introduced. This model is used as an environment for the RL agent as long as there is a relative speed

[216] Maucher 1990, Naus et al. 2008 and Dolcini et al. 2010
[217] Gwosch et al. 2013
[218] Seitenbrecher 1997

between the clutch and the engine. The simplified model in slipping-clutch mode is depicted in Fig. 7.3.

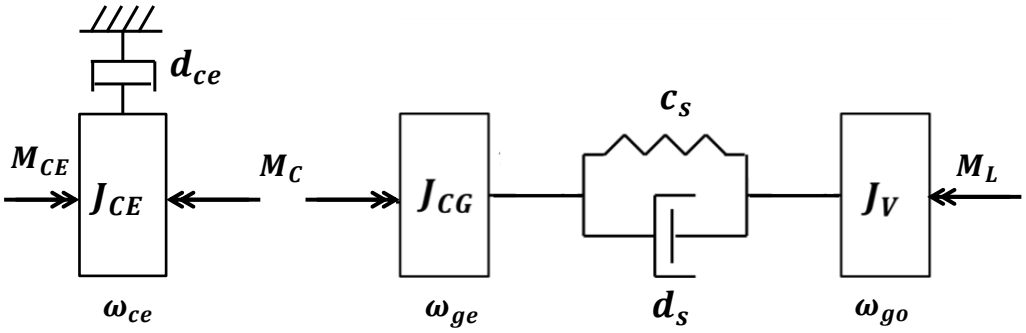

Figure 7.3: Simplified model of the reduced drive train in slipping-clutch mode

The model is fairly similar to the model presented in previous chapters, however, it has certain distinctive features. The powertrain is divided into an input and an output side separated by the clutch. All inertias before the clutch, such as flywheel and crank shaft, are added to the combustion engine inertia J_{CE} rotating with the engine speed ω_{ce}. The input side of the powertrain has the damping d_{ce} and is driven by the engine torque M_{CE}. Energy exchange between sides takes the form of the clutch torque M_C. The output side consists of the added inertias of the clutch and the gearbox J_{CG} and the inertia J_V as a result of the mass of the wheels and the vehicle. They rotate with the entry speed ω_{ge} and output speed ω_{go} respectively. The two inertias are linked with the stiffness c_s and damping d_s of the side shaft, as the component with the lowest stiffness in the drive train. Finally, the powertrain can be burdened by the load torque M_L. The values of the parameters can be taken from Appendix B.

The dynamic system is now presented in its state-space representation.[219] The state vector x_{slip} gives the full dynamic description of the system with a slipping clutch. The kinetic energy of the system is accounted for with the speeds of the engine, clutch and vehicle. The potential energy is determined by the angles of the clutch and the vehicle masses and their values at start. However, only the difference between them, the torsion angle of the drive train θ, is of importance. The state vector can be noted as follows:

$$x_{slip} = \left(\omega_{ce}, \omega_{ge}, \omega_{go}, \theta\right)^T \qquad \textit{Eq. 7.1}$$

The input vector is given by the torques of the combustion engine, the clutch and the load:

[219] Gwosch et al. 2013, Seitenbrecher 1997 and Hangos et al. 2001

$$u_{slip} = (M_{CE}, M_C, M_L)^T \qquad \textit{Eq. 7.2}$$

The balance of torques for the three mass inertias yields the system of differential equations that dynamically describes the system:

$$\begin{aligned} 0 &= J_{CE}\dot{\omega}_{ce} + d_{ce}\omega_{ce} + M_C - M_{CE} \\ 0 &= J_{CG}\dot{\omega}_{ge} + d_s\dot{\theta} + c_s\theta - M_C \\ 0 &= J_V\dot{\omega}_{go} - d_s\dot{\theta} - c_s + M_L \end{aligned} \qquad \textit{Eq. 7.3}$$

The system of three differential equations has four unknowns. The state-space representation[220] can be formulated with the following relationship:

$$\dot{\theta} = \omega_{ge} - \omega_{go} \qquad \textit{Eq. 7.4}$$

With *Eq. 7.4* the state space representation is given by the state vector x_{slip} and the output vector y_{slip}:

$$\begin{aligned} \dot{x}_{slip} &= A_{slip}x_{slip} + B_{slip}u_{slip} \\ y_{slip} &= C_{slip}x_{slip} + D_{slip}u_{slip} \end{aligned} \qquad \textit{Eq. 7.5}$$

The state matrix A and the input matrix B are defined as follows:

$$A_{slip} = \begin{pmatrix} -d_{ce}J_{CE}^{-1} & 0 & 0 & 0 \\ 0 & -d_sJ_{CG}^{-1} & d_sJ_{CG}^{-1} & -c_sJ_{CG}^{-1} \\ 0 & -d_sJ_V^{-1} & -d_sJ_V^{-1} & c_sJ_V^{-1} \\ 0 & 1 & -1 & 0 \end{pmatrix} \qquad \textit{Eq. 7.6}$$

$$B_{slip} = \begin{pmatrix} J_{CE}^{-1} & -J_{CE}^{-1} & 0 \\ 0 & J_{CG}^{-1} & 0 \\ 0 & 0 & -J_V^{-1} \\ 0 & 0 & 0 \end{pmatrix} \qquad \textit{Eq. 7.7}$$

Finally, the output vector has to equate to the state vector $x_{slip} = y_{slip}$ so that the output matrix is the identity matrix $C_{slip} = I_4$ and the feedthrough[221] D_{slip} is zero.

$$x_{slip} = y_{slip} \qquad \textit{Eq. 7.8}$$

$$D_{slip} = 0 \qquad \textit{Eq. 7.9}$$

[220] This is not to be confused with the Markov state-space representation of the environment in a RL framework. This is the state-space representation in the classic control sense.

[221] The feedthrough is often called feed forward matrix, e.g. in Hangos et al. 2001

7.1.2.2 Simulation Model in closed Clutch Mode

The simulation model in closed clutch mode is intended to reproduce as accurately as possible the behavior of the physical test bench, once the rotational speeds of the clutch and the engine are synchronized. Usually, the speed of the engine is decreased during the synchronization process and it has to be raised to the target speed in order for the maneuver to be regarded as concluded. A graphic representation of the closed clutch test bench model is depicted in Fig. 7.4.

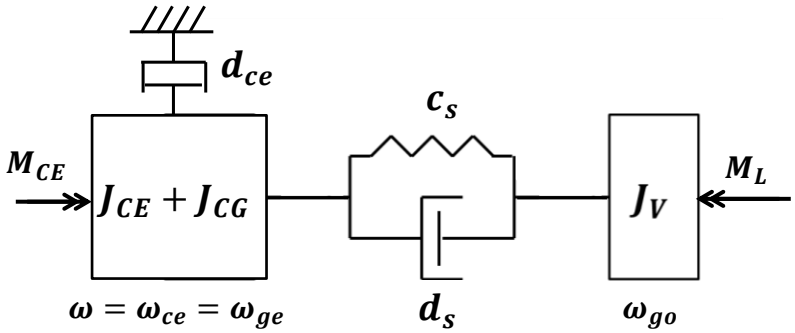

Figure 7.4: Simplified model of the reduced drive train in closed-clutch mode

In closed-clutch mode the inertia of the engine is added to the inertias of the clutch and the gearbox. They rotate at the same speed $\omega = \omega_{ce} = \omega_{ge}$. Micro-slip between the discs in closed-clutch mode is neglected. Following the same procedure as for the model in slipping-clutch mode, the dynamic system can be described in its state-space representation.

The new state vector contains only three variables, as opposed to four in slipping-clutch mode, since the engine and entry speeds are the same in closed-clutch mode:

$$x_{closed} = \left(\omega, \omega_{go}, \theta\right)^T \qquad \textit{Eq. 7.10}$$

Consequently, the clutch torque is eliminated from the input vector:

$$u_{closed} = (M_{CE}, M_L)^T \qquad \textit{Eq. 7.11}$$

In analogy to the slipping-clutch mode, the balance of torques for the inertias in the system considering *Eq. 7.4* delivers the system of differential equations of motion:

$$\begin{aligned} 0 &= (J_{CE} + J_{CG})\dot{\omega} + d_s\omega + c_s\theta - M_C \\ 0 &= J_V\dot{\omega}_{go} - d_s\dot{\theta} - c_s + M_L \\ \dot{\theta} &= \omega - \omega_{go} \end{aligned} \qquad \textit{Eq. 7.12}$$

Analogous to *Eq. 7.5*, the system is brought to its state-space representation:

$$\dot{x}_{closed} = A_{closed}x_{closed} + B_{closed}u_{closed}$$
$$y_{closed} = C_{closed}x_{closed} + D_{closed}u_{closed} \qquad \textit{Eq. 7.13}$$

With the corresponding input and output matrices A_{closed} and B_{closed}, respectively:

$$A_{closed} = \begin{pmatrix} -(d_{ce}+d_s)(J_{CE}+J_{CG})^{-1} & d_s(J_{CE}+J_{CG})^{-1} & -c_s(J_{CE}+J_{CG})^{-1} \\ -d_s J_V^{-1} & -d_s J_V^{-1} & c_s J_V^{-1} \\ 1 & -1 & 0 \end{pmatrix} \qquad \textit{Eq. 7.14}$$

$$B_{closed} = \begin{pmatrix} (J_{CE}+J_{CG})^{-1} & 0 \\ 0 & -J_V^{-1} \\ 0 & 0 \end{pmatrix} \qquad \textit{Eq. 7.15}$$

Once again, the output vector has to equate to the state vector and the feedthrough has to be zero:

$$x_{closed} = y_{closed} \qquad \textit{Eq. 7.16}$$

$$D_{closed} = 0 \qquad \textit{Eq. 7.17}$$

Finally, a criterion needs to be formulated in order to determine whether the clutch in the system is “slipping” or “closed”. In order for the clutch to be regarded as closed, two conditions need to be met. The first is the intuitive condition that the rotational speeds of the engine and the clutch are the same. Furthermore, a consideration regarding the clutch torque is made. In accordance to SEITENBRECHER[222], the clutch torque M_C has to be greater than the inner torque in the clutch M_I which will be introduced shortly. This condition is necessary to ensure that a closed clutch can break loose and slip, e.g. as a result of a decrease in the clamping force. The two conditions can be formulated as follows:

$$\omega_{ce} \neq \omega_{ge} \; or \; |M_I| > |M_C| \; \rightarrow \; slipping \qquad \textit{Eq. 7.18}$$

$$\omega_{ce} = \omega_{ge} \; or \; |M_I| \leq |M_C| \rightarrow closed \qquad \textit{Eq. 7.19}$$

The inner torque can be calculated with relative ease from the balance of torques in either one of the first two equations in *Eq. 7.3* by substituting M_C with M_I. However, a low computing effort alternative that does not require time derivatives is provided by SEITENBRECHER[222] considering that M_I is only relevant in closed-clutch mode:

$$M_I = \frac{1}{J_{CE}+J_{CG}}\left(J_{CE}\left(d_s\left(\omega_{ge}-\omega_v\right)+c_s\theta\right)+J_{CG}\left(M_{CE}+d_s\omega_{ce}\right)\right) \qquad \textit{Eq. 7.20}$$

[222] Seitenbrecher 1997

7.1.2.3 Consideration of the physical Test Bench

Thus far, there is no big difference between the simulation model presented in the previous chapter and those presented in this one other than the distinction between slipping- and closed clutch modes. However, in this section the availability of signals and values in the physical test bench is taken into account. In addition, numerous additions and modifications to the model are undertaken, in order to better reproduce the behavior of the physical drive train. These modifications result mainly from hardware and technical limitations and constraints. Furthermore, the values of the parameters in the model as well as the dead times during the operation of the test bench are a result of an extensive system identification performed by GWOSCH ET AL.[223]. Any values not specified in this chapter can be found in Appendix B.

Input Variables

The simulation model requires three torque values as input to compute the dynamic behavior of the clutch. However, only the load torque M_L can be directly specified, whereas the other two need to be calculated using the input possibilities of the physical test bench. These are the specifications of the rotational speed of the input DC motor and the axial travel path of the linear actuator responsible for closing the clutch. These two input values are used to compute the other two values of the input vector of the model in state-space form: the torques of the combustion engine and the clutch, M_{CE} and M_C, respectively.

The input value of the rotational speed of the engine is set to be constant ($900\ 1/\mathrm{min}$). The set current for the DC motor is computed using a PI-controller and the feedback value of the engine speed from the model. The corresponding torque M_{CE} results from a characteristic curve determined in the mentioned system identification. One specification made at this point is the limitation of the engine torque to a maximum $1.7\ \mathrm{Nm}$ in order to prevent an overstressing of the side shaft of the physical test bench.

The clutch torque M_C is computed as specified in chapter 2.1.1, concretely, *Eq. 2.5*. The friction coefficient is modeled according to *Eq.6.3*. The values of the static friction and the friction gradient, μ_{st} and μ' respectively, as well as the mean friction radius R_M have a constant value measured on the physical test bench. The clamping force in the clutch is the result of a correlation of the axial travel of the linear actuator in the drive train and the axial force in the clutch and a model of the actuator's dynamic behavior. The translation of axial travel to force is approximated with a third degree polynomial, whereas the actuator dynamics are assumed to possess a PT_2 behavior. Again, the parameterization resulted from measurements on the test bench.

[223] Gwosch et al. 2013

Finally, the value of load torque can be directly specified as an input for the model. For example, a load as a result of the combined effects of an uphill track and wind resistance can be applied to the drive train. However, in the context of this work additional loads are neglected.

Sampling Rate and Dead Times

A general sampling rate of $10\,\mathrm{kHz}$ was defined for the simulation model. This is more than accurate enough in order to reproduce the judder vibrations that lie in the focus of this work. This sample rate has no effect on the RL framework whatsoever; it is merely the sample rate with which the physical test bench is simulated. The criteria formulated in *Eq.7.18* and *Eq.7.19* are evaluated at this rate. The criteria need to be reformulated, since the comparison of rotational speeds involved cannot be properly processed in practice. In simulation, the floating-point number values of the two speeds are unlikely to ever be exactly the same, whereas in a physical implementation, the measured values will also unlikely be the same due to noise and/or their digital postprocessing. The empirically determined tolerance factor $\vartheta = 0.5\ 1/min$ is introduced and the criteria for slipping- and closed-clutch mode reformulated as follows:

$$\left|\omega_{ce} - \omega_{ge}\right| \geq \vartheta \quad or \quad |M_I| > |M_C| \rightarrow slipping \qquad \textit{Eq. 7.21}$$

$$\left|\omega_{ce} - \omega_{ge}\right| < \vartheta \quad or \quad |M_I| \leq |M_C| \rightarrow closed \qquad \textit{Eq. 7.22}$$

The RL framework itself samples the environment, thus the simulation model of the test bench, at a lower rate. The state of the environment is updated with a sample rate of $1\,\mathrm{kHz}$, which both suffices for the agent to sense the environment accurately and without causing unnecessary computing effort. This is also the sample rate at which the physical test bench will be sampled later on. The actual agent environment interaction takes place at a rate of ten interactions per second, thus $10\,\mathrm{Hz}$, due to the dynamics of the clutch actuator (see 7.2.1.3). The environment is updated at a higher rate, since some of the state variables require multiple samples in order to be computed. Such is the case of variables that result from derivatives or integrations. The significantly higher rate, at which these variables are sampled, ensures that an accurate value is available when the agent probes the state of the environment. An example for such a state variable is the torsion angle θ. For its computation, an integration of the entry and output speeds of the drive train is required. Also, the reward-signal can be calculated more accurately.[224]Also, the higher sampling rate

[224] cf. chapter 5.4

of $10\,\mathrm{kHz}$ allows more accurate control of the workflow in the synchronization maneuver, as will be described afterwards in this section.

Finally, there is a set of different dead times in the components of the physical drive train, that are accounted for in the form of delay blocks. A delay of one sample is caused by the real time environment in use during the transmission of measurement values of parameters from the test bench, i.e. output values of parameters, and again during the transmission of input values to the test bench. These dead times amount to delays of each $0.001\,s$. The biggest dead time is found in transmission of the input axial travel value of the clutch actuator. It amounts to a value that fluctuates between $0.005\,s$ and $0.01\,s$.

Maneuver

The maneuver that comprises the learning episodes in this work consists of synchronization of the input and output sides of the powertrain until the desired speed and torque are achieved.[225] At the beginning of the maneuver, at the input side, the engine and the input clutch disc are rotating at the velocities' setpoint value. Subsequently, the first of four phases sets in. These phases are the *standard force ramp synchronization phase*, the *RL-controlled synchronization phase*, the *acceleration to setpoint value phase* and the *output slowing down phase*. After the last phase the starting conditions are met again and the procedure can start anew. In this section, the phases as well as the conditions that need to be met in order to transition into the next phases are introduced.

At the beginning of each episode, thus of every repetition of the maneuver, the input side of the drive train is rotating at setpoint value speed, whereas the output side is standing still. The clutch is actuated following a standard force ramp in order to establish contact between the initially separated input and output discs of the clutch. Simulations in which the RL-agent was responsible for the clutch actuation from the onset of an episode showed that due to its inherent explorative behavior it is counterproductive in early phases of an episode. This is mainly due to the fact that the agent needs to learn that most actions taken at the beginning of an episode, when judder vibrations are not present yet, either cause rather than reduce vibrations or delay the process of establishing contact between the clutch discs. For this reason, the RL-agent and the corresponding RL framework remain inactive until shortly before the output speed reaches the drive train's eigenfrequency, where judder vibrations reach their highest amplitude. The gradient of the standard force ramp was determined

[225] cf. chapter 5.1.2

empirically, favoring a solution that achieves a relatively fast closing of the clutch.[226] The criterion[227] that determines the point in time at which the RL-agent assumes the control of the clamping force control is formulated as follows:

$$\left|\omega_{ge} - \omega_{go}\right| = \omega_{rel} \leq S \cdot \omega_{eigen} = S \cdot 2\pi \cdot f_{eigen} \qquad \textit{Eq. 7.23}$$

The safety factor S was determined empirically and set to $S = 1.2$. It ensures that the agent is only active when suppression of judder vibrations is necessary without having a negative effect during the beginning of the maneuver. Another function of the standard force ramp at the beginning of an episode is that of establishing contact between the initially separated input and output discs of the clutch.

Once the criterion in *Eq. 7.23* is met, the RL-agent and the rest of the RL framework are activated. The agent then sets the clamping force of the clutch until either the entry and output speeds are synchronized or three seconds have passed. If synchronization is not achieved in these three seconds, the episode is regarded as failed and the slowing down phase is initiated.[228] The input and output sides are considered synchronized when their difference in speed is deemed neglectible:

$$\left|\omega_{ge} - \omega_{go}\right| < 2\,\frac{rad}{s}\ for\ at\ least\ 0.1\ seconds \qquad \textit{Eq. 7.24}$$

If the synchronization is successful, the clutch is not slipping anymore so no consideration of judder or its suppression is necessary. For this reason, the RL agent is disengaged and the clutch actuator is set to apply the highest axial force possible, without exceeding the torque maximum specified previously. The phase is terminated when the speed of the synchronized drive train reaches the setpoint value of the input speed again. The condition is formulated as follows:

$$\left|\omega_{go} - \frac{\pi}{30} \cdot 900\,\frac{1}{min}\right| < 2.5\,\frac{rad}{s}\ \ for\ at\ least\ 0.3\ seconds \qquad \textit{Eq. 7.25}$$

The slowdown phase is initiated as soon as the condition above is met. It consists of the opening of the clutch and the subsequent application of a braking torque on the output DC motor until the output speed reaches $30\ 1/\mathrm{min}$. Afterwards, the drivetrain is allowed to roll out to a full stop.[228]

---

[226] cf. 7.2.1.3

[227] In simulation, the criterion expressed in *7.23* coincides with the actuator reaching 400 incr. following the standard ramp.

[228] Slowing down the drive train is only really necessary on the physical test bench. In simulation the episode is simply interrupted and the initial conditions are reset.

7.2 RL Framework in the simulated Test Bench Environment

Thus far in this chapter, the new RL environments, the physical and the simulated test bench, have been introduced. In this subchapter, the changes to the remaining elements of the RL framework as a consequence of its implementation on the simulated test bench environment and the corresponding results are presented. These results are evaluated in order to identify the one that yields the highest potential to be successfully implemented in the physical test bench environment.

7.2.1 Influence on the Elements of the RL Framework

Having introduced the new simulated environment, the focus now lies in the remaining elements of the RL framework introduced in 2.2.2. These are the agent, the reward-signal, and the state and action-signals and the corresponding state and action spaces.

7.2.1.1 State Space and Reward-signal

No significant adjustments regarding the definition of the state-signal and the design of the state space are undertaken in this new environment of the RL framework. The state-signal proposed in 5.2 can be implemented without constraints and is formally described by *Eq. 5.2*. However, the resolution of the discrete meshes for the rotational speeds in the state-signal is adjusted to reduce computing effort and operating time of the physical test bench. The resolutions and boundaries of the three state variables can be taken from Table 7.1.

State variable	Unit	Lower boundary	Upper boundary	Resolution
ω_{ge}	$[rad/s]$	-60	130	5
ω_{go}	$[rad/s]$	0	120	10
θ	$[rad]$	-1	2	0.1

Table 7.1: Resolution of the new state space

The adjustments are mainly necessary due to the limited amount of data that can be transferred in the real-time environment on the physical test bench. In comparison to the previous simulations of chapter 5, the discretization of the entry speed is twice as rough and that of the output speed even four times as rough. However, according to the preliminary studies presented in 6.3.1 and depicted in Fig. 6.12, the loss in effectiveness of the RL-control should not suffer substantially.

No changes are necessary in regards to the reward-signal for the new environment. Its formal definition is given by *Eq. 5.5.*

7.2.1.2 Agent

The role of the agent and its interaction with the remaining elements remains unchanged in comparison to previous implementations of this work and the definition provided in 5.5. However, two major adjustments are necessary to accomplish the new task at hand.

The first adjustment concerns the rule with which the Q-Table is updated after every interaction with the environment. Thus far, the on-policy TD-method SARSA had been preferred, due to its faster convergence and often better results. However, in environments where the Markov property is more compromised and there are no transition probabilities available for all states or state-action pairs, it is not necessarily the case. This is evident in the fact that the value of a state-action pair is determined in part by the state-action pair that follows after an interaction. If the following state-action pair is of low value, the entire current policy is updated, hence on-policy. A similar negative effect is experienced when explorative actions result in a transition to a low value state-action pair. Foreseeing that the physical test bench is most likely to violate the Markov property considerably, the Q-Table is now to be evaluated following the off-policy TD method Q-Learning[229].

The second modification is undertaken in order to adjust the explorative behavior of the agent to the task at hand. Explorative and exploitative behaviors are still balanced with the help of the ϵ-greedy policy, however, its parameters are adjusted in order to consider the number of completed episodes $N_{episodes}$:

$$\varepsilon = 0.2 \cdot e^{-0.003 \cdot N_{episodes}} + 0.005 \qquad \textit{Eq. 7.26}$$

The values were determined to ensure that approx. 20% of the actions at the beginning of the learning task are random. This value decreases exponentially with the number of completed episodes until it is neglectible. The value summed at the end ensures a persistent small amount of exploration throughout the task.

7.2.1.3 Action Space and Action-Signal

The most significant adjustment that arises from the application of the RL framework to the new environment is found in the implementations of the action-signal and the corresponding action space. As stated in the previous chapter, one of the necessary input signals for the simulation model to reproduce the dynamic behavior of the physical test bench is the clutch torque, which is dependent on the clamping force applied by the clutch actuator. In the previous chapters, this clutch actuation force was defined as the action-signal in the RL framework and the action space was designed accordingly. The response of the actuator to a force input value by the agent was

[229] cf. Fig. 2.15 in 2.2.4.3

modeled to have a PT_2-behavior and to fully apply the demanded force within $0.1\,\mathrm{s}$. However, experiments on the physical test bench determined that this value is overly optimistic. On its force (closed-loop) control mode, the actuator is unable to cover the relevant force range[230] within the defined time frame. Furthermore, the response of the actuator lacks the required reproducibility for the application of a RL algorithm. The results of an experiment with a square-wave force setpoint signal can be taken from Fig. 7.5.

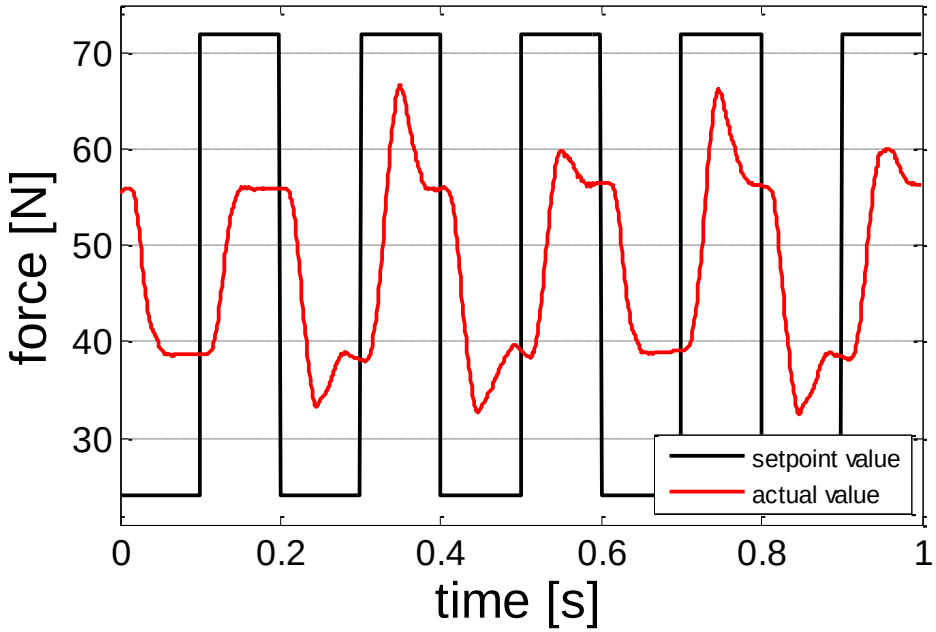

Figure 7.5: Force control of the physical clutch actuator (closed-loop)

The different responses of the actuator to the setpoint signal are evident. Also, the demanded $75\,\mathrm{N}$ cannot be set in the required time frame. Whereas the value of the set force is not as important for the RL-algorithm as the reproducibility with which it is set, an insufficient force due to the actual force not following the setpoint value could lead to the required torque not being produced in the clutch.

The use of the force control actuator is not viable mainly due to the lack of reproducibility. In the context of this work, a position (open-loop) control of the actuator is proposed. The main advantages are the gained reproducibility of the actuator's response and the increased force range accessible within the required time frame. However, the initial gain in reproducibility is only guaranteed as long as the position-force correlation introduced in the previous section is valid. This might not always be

[230] The relevant force range is limited by the force that leads to the maximum torque allowed defined previously. The lowest force considered for the range results from the transition in the maneuver from following the standard force ramp and the activation of the RL-agent.

the case on a physical test bench due to temperature and operation related effects. The response of the actuator in position control mode can be taken from Fig. 7.6.

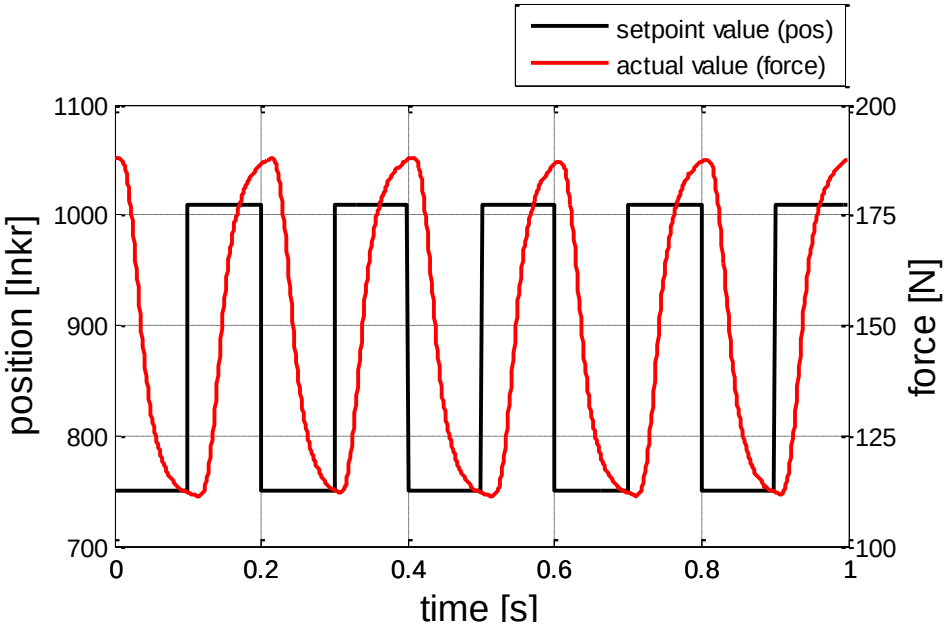

Figure 7.6: Position control of the physical clutch actuator (open-loop)

The force range accessible within the proposed time frame of 0.1 seconds in open-loop control mode is vastly greater and the response of the actuator is highly reproducible. For this reason, this form of actuation is preferred in this work, bearing in mind the potential difficulties faced during its subsequent implementation on a physical test bench.

As a consequence, the action-signal becomes a position signal instead of a force signal. Instead of selecting a force to be applied, the agent selects an incremental position the clutch actuator has to reach. In order to maintain the solution space as reduced as possible, a sensible range for incremental position of the actuator is defined as the action space for the RL task. This range is then homogenously[231] discretized to no more than five actions, thus no more than five incremental positions. In order to determine the best distribution for the actions around this orientation value, a standard engagement ramp was defined as a means to provide a benchmark for different action space proposals. After ramps with different gradients were tested and evaluated, a gradient of $670\ \mathrm{incr/s}$ was found to provide a quick closing of the clutch at a moderate rate of clutch judder.[232] Measured in terms of the reward-signal introduced in 5.4, the return for the synchronization maneuver following this standard ramp is -15. Furthermore, as an orientation about the position of the increments, the value at which

[231] The values are rounded to the nearest ten when necessary.

[232] Exemplary results of a simulated synchronization maneuver following the introduced standard ramp are presented in 7.2.2.

the criterion expressed in *7.23* is fulfilled while following the aforementioned standard ramp is used. In simulation, the transition from the standard force ramp to the RL-controlled synchronization coincides with the actuator reaching $400\,\mathrm{incr}$. Having defined every element of the RL framework except the distribution of actions in the action space, five different proposals were made and the results of a standard Q-learning process until convergence were compared. The results can be taken from Table 7.2.

Action space $\mathcal{A}$ [incr]	Reward after convergence	Reduction of judder [%]
Standard ramp	-15	0
$\{100,200,300,400,500\}$	-6.25	58.3
$\{300,350,400,450,500\}$	-1.68	88.8
$\{400,420,450,470,500\}$	-0.99	93.4
$\{400,450,500\}$	-7.77	87.0
$\{400,500\}$	-1.96	48.2

Table 7.2: Comparison of different state space configurations

From these results it becomes evident that having more actions to choose from allows the agent to find better solutions, therefore the solutions involving less than five elements in the action space are discarded.[233] From the remaining proposals, the action space that provides the agent with actions in the highlighted interval between 400 and 500 increments yields the best result and is therefore selected for all further simulations and experiments. Therefore, the action space is defined as follows:

$$\mathcal{A} = \{400\ incr, 420\ incr, 450 incr, 470\ incr, 500\ incr\} \qquad \textit{Eq. 7.27}$$

Taking the previously introduced state space into consideration, the solution space of the task consists of 15 500 state-action pairs (elements in the Q-Table).

7.2.2 RL Algorithms in the simulated Environment

Having defined all the elements of the RL framework for the new environment, a comparative study of the different approaches can be performed. Concretely, the performance of a traditional Q-learning algorithm is compared to that of the novel approach involving the Gaussian update of the value function introduced in 6.3.4. Furthermore, three further approaches for updating the value function are suggested in order to determine which one is the most likely to produce the best results on the

[233] The learning tasks involving action spaces with less than five elements did converge faster, however the effectiveness of the judder reduction decreased proportionally.

physical test bench. The first of these additional approaches involves a dynamic adjustment of the learning rate in a Q-Learning algorithm. Analogously, the second of the approaches adjusts the exploration rate in accordance to the progress of a learning task and the current episode. Finally, a numeric comparison of the presented approaches is performed. One characteristic of the learning task in the context of this chapter is that it is limited in its length to a fixed number of episodes. This is due to operating time constraints related to an implementation on a physical test bench.[234] However, in order to establish a basis for the comparison of each approach's performance, a benchmark maneuver is established first. This maneuver is the synchronization maneuver following the standard force ramp introduced previously. The maneuver is evaluated as if it had been performed by a RL agent, thus acquiring a reward after each discrete time step and a return after the maneuver is completed. Even though the reproducibility of the results in simulations is very high, the benchmark value for the comparison was averaged over 250 repetitions of the maneuver. The average return after an engagement following the standard ramp is -15, as contained in the first row of Table 7.2. The graphs of the relevant rotational speeds for an exemplary maneuver can be taken from Fig. 7.7.

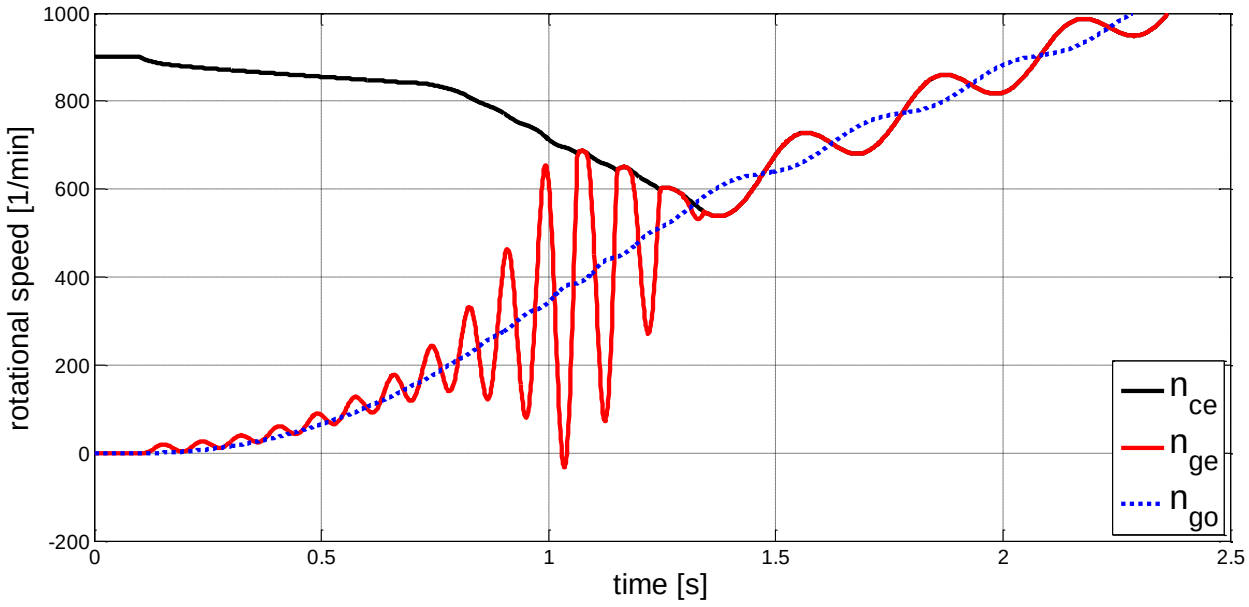

Figure 7.7: Clutch synchronization maneuver following a standard clamping force ramp ($670\ incr/s$)

7.2.2.1 Classical Q-Learning Algorithm

The first approach implemented on the new environment is a classical Q-Learning algorithm. The parameters of the algorithm and the learning task, which has not yet been specified, can be taken from Table 7.3.

[234] cf. 7.1.2

Parameter	Value	Description
α	0.2	Learning rate
γ	0.1	Discount rate
$n_{episodes}$	1000	Length of learning task

Table 7.3: Parameterization of the classic Q-Learning approach

The results of the implementation are presented graphically in the form of its learning curve, i.e. the progression of the return over the course of learning episodes, and a greedy episode after convergence. The corresponding figures are depicted in Fig 7.8 and 7.9. As evidenced by these figures, judder vibrations are considerably reduced. The best episodes effectively reduced judder by 93.4% in comparison to the standard clamping force ramp. However, outlier episodes, in which judder is occasionally even worse than the standard, are present throughout the learning task.

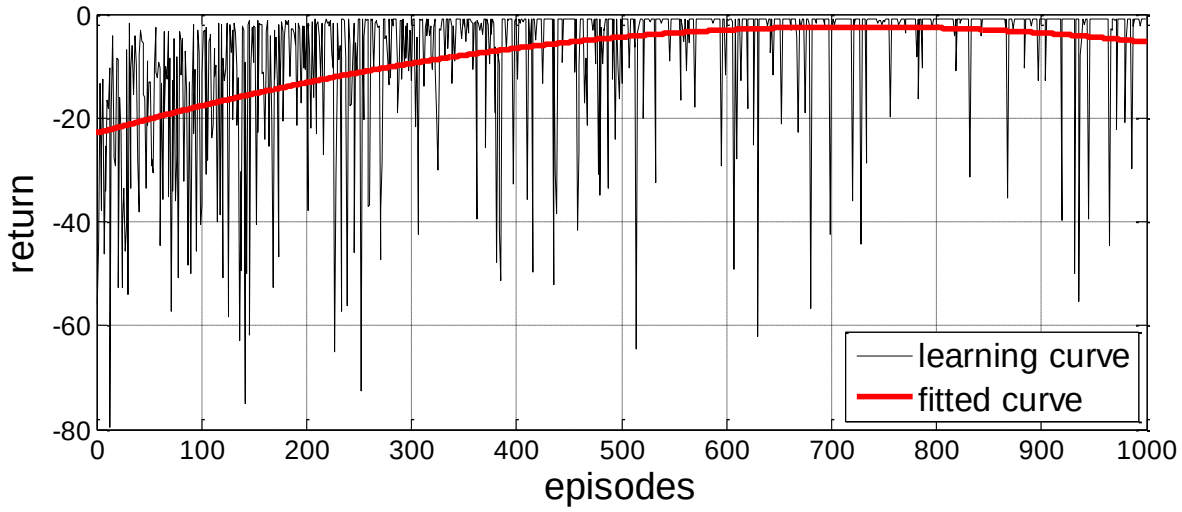

Figure 7.8: Learning curve for the implementation of the classic Q-Learning algorithm on the simulated test bench environment.

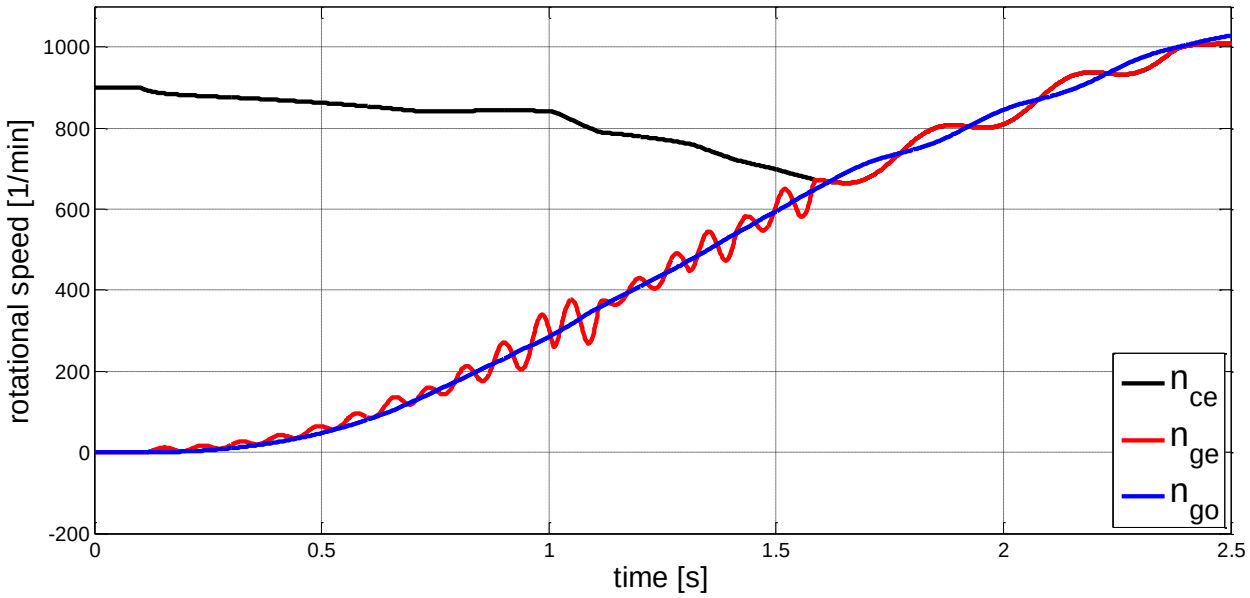

Figure 7.9: Greedy episode of the classic Q-Learning agent on the simulated test bench environment.

7.2.2.2RL algorithm with the Gaussian-update Method

The second approach implemented in the simulated test bench environment consists of a Q-Learning agent with the Gaussian-update method for the approximation of the value function. It was selected as it was the most promising of the approaches studied in the preliminary studies presented in chapter 6. This new implementation is conceived with a slightly different parameterization as a consequence of the adjustments to the state space discretization introduced in 7.2.1.1. The standard deviations of the rotational speeds are adjusted according to their resolutions. The new set of parameters for the new implementation of the Gaussian-update of the value function is contained in Table 7.4.

Parameter	Value	Description
$\sigma_{n_{ge}}$	1	Standard deviation of n_{ge} for factor $belief$
$\sigma_{n_{go}}$	2	Standard deviation of n_{go} for factor $belief$
σ_{θ}	0.4	Standard deviation of θ for factor $belief$
σ_{a}	0.4	Standard deviation of a for factor $belief$
k	10	k value for factor $need$
γ	0.1	Discount rate
$n_{episodes}$	1000	Length of learning task

Table 7.4: Parameterization of the Gaussian-update method

The results of this implementation are presented graphically in the form of its learning curve and a greedy episode after convergence depicted in Fig 7.10 and 7.11, respectively.

In contrast to the results on the abstract drive train simulation model, the Gaussian update does not seem to bring forth an improved learning behavior, at least not within the pre-established number of episodes. Nevertheless, the results are fairly satisfying, as 90.5% of judder vibrations could be suppressed and synchronization was achieved slightly faster. Also, the rotational speed of the engine is not reduced as dramatically as it was in previous approaches. However, the prevalence of outlier episodes with exacerbated vibrations seems higher.

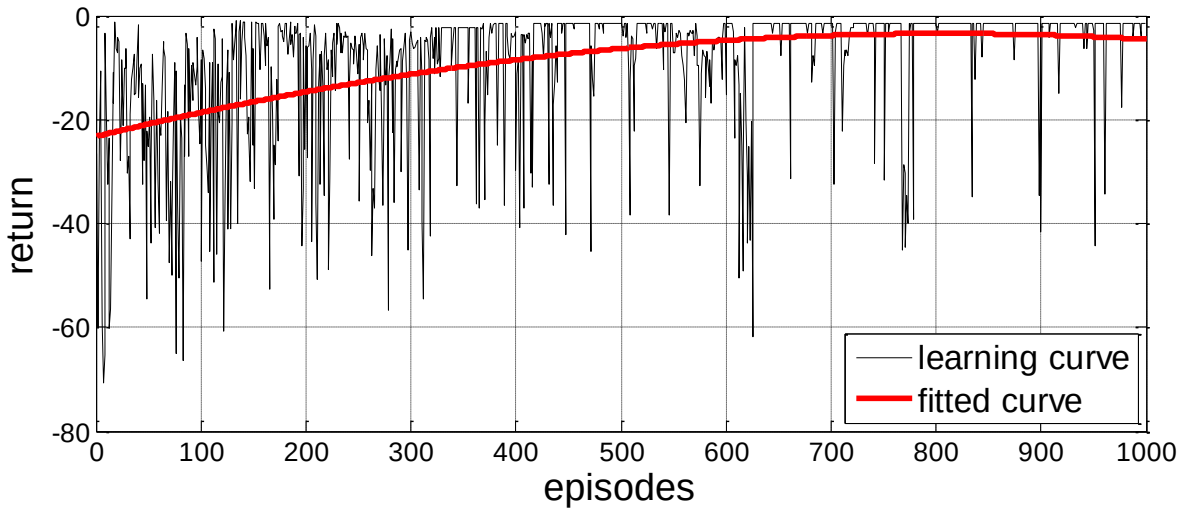

Figure 7.10: Learning curve for the implementation of RL-algorithm with the Gaussian-update method on the simulated test bench environment.

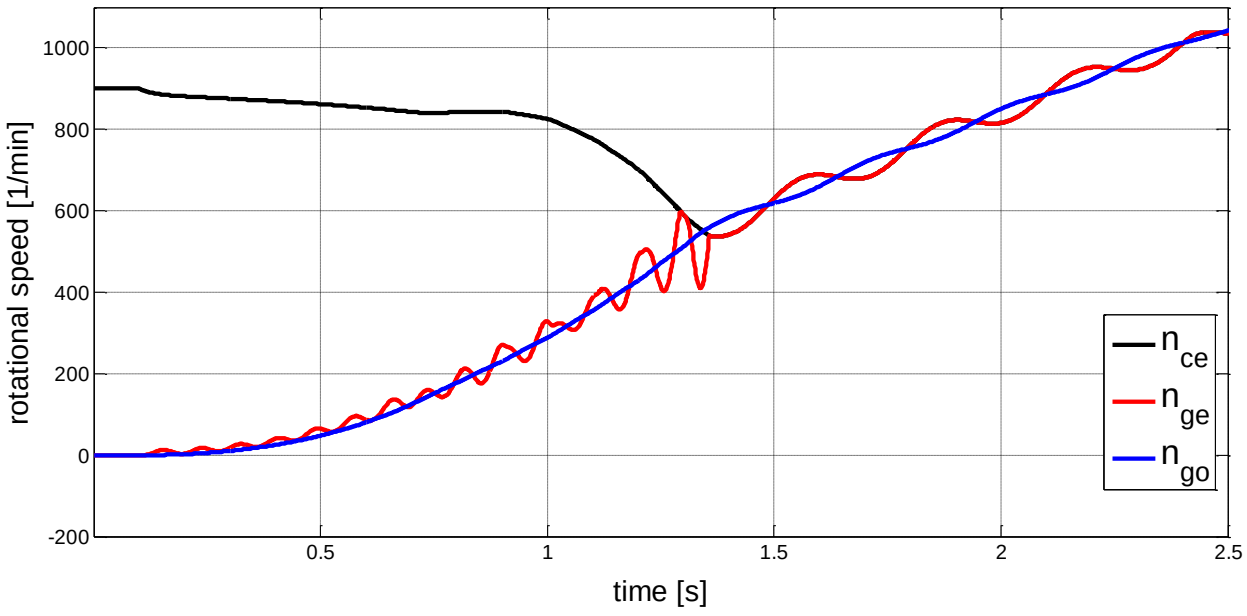

Figure 7.11: Greedy episode of RL-agent with the Gaussian-update method on the simulated test bench environment.

7.2.2.3 Dynamic Learning Rate to the RL-Agent

Given the fact, that the application of the Gaussian-update method did not yield the expected positive results, three new approaches are introduced for the new environment.

The first of these new approaches consists of a Q-Learning algorithm with a dynamic learning rate. All algorithms applied so far relied on a constant learning rate throughout the learning task. According to SUTTON AND BARTO[235], this causes recent estimations of the value function to have a higher weight than old estimation, under the assumption that the newest information is the most reliable. This is also a desirable behavior when tracking non-stationary problems. However, in a stationary environment subject to noisy signals, it might cause a bad sample to have an overly strong effect on the estimation of the value function. Under the assumption that the simulated and physical test bench environments are stationary but subject to a stochastic noise in their state-signals, a new learning rate that evaluates an average of all previous estimations should yield better results. Such a *dynamic learning rate* is introduced by EVEN-DAR AND MANSOUR[236] and defined as follows:

$$\alpha(s,a) = \frac{1}{N(s,a)^p} \qquad \textit{Eq. 7.28}$$

where $N(s,a)$ is the number of visits to the state-action pair (s,a). The learning rate is denominated *linear* for $p = 1$ and polynomial for $p < 1$. EVEN-DAR AND MANSOUR[236] achieved the best results for $p \approx 0{,}85$. In this work, dynamic learning rates with $p = 1$ and $p = 0{,}8$ are tested.

The learning curve for the implementation of the RL-algorithm with a linear learning rate is depicted in Fig. 7.12.

[235] Sutton / Barto 1998
[236] Even-Dar / Mansour 2004

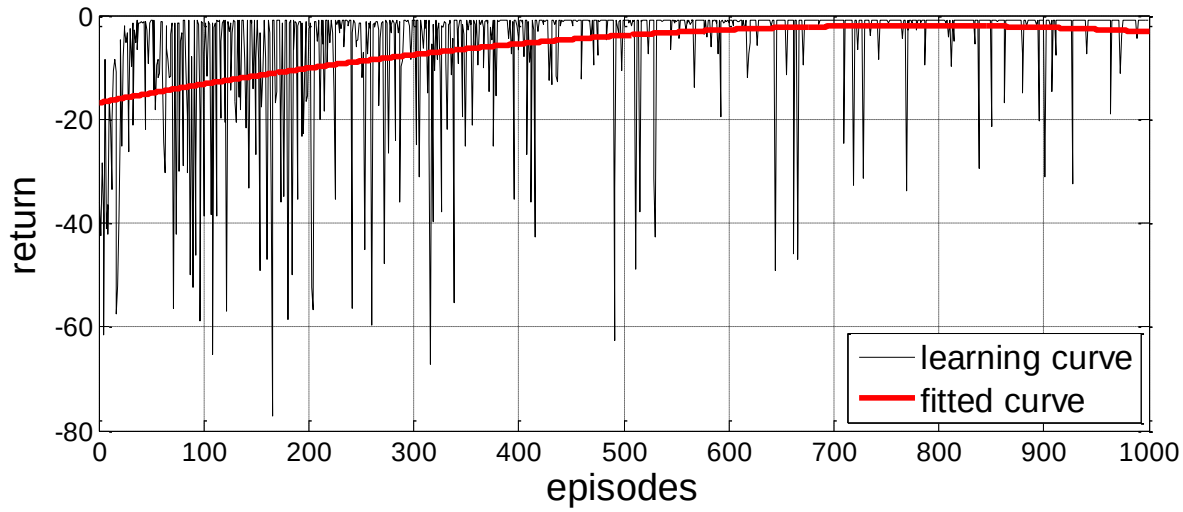

Figure 7.12: Learning curve for the RL-agent with a linear learning rate on the simulated test bench environment

The converged action sequence results in a greedy episode that is exactly the same as that of the classic Q-Learning implementation depicted in Fig 7.9. Thus, the effectiveness of the new approach is the same as that of the classic Q-Learning algorithm. However, the convergence is faster and the prevalence of outliers with high-amplitude vibrations throughout the task is lower.

Likewise, the learning curve for the implementation of the RL-algorithm with a polynomial learning rate can be taken from Fig. 7.13.

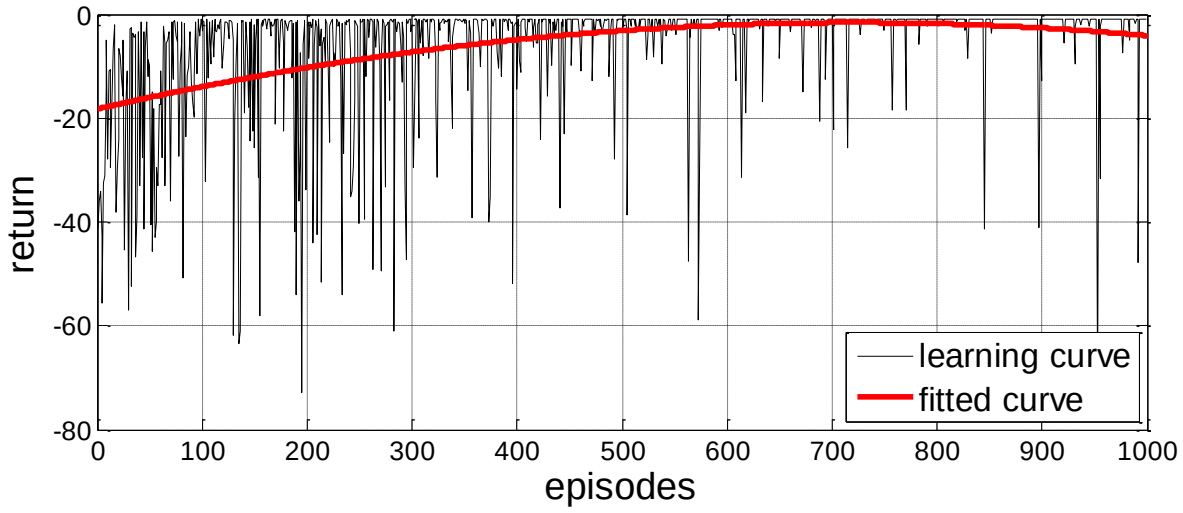

Figure 7.13: Learning curve for the RL-agent with a polynomial learning rate on the simulated test bench environment

Similarly to the agent with a linear learning rate, the one with a polynomial rate reaches convergence sooner than both the classic Q-Learning and the Gaussian-update method implementations. Nonetheless, a few strong outliers are visible near the end of the learning task. Again, the converged action sequence is the same learnt with Q-

Learning and with the linear learning rate. It delivers the same greedy episode depicted in Fig. 7.9 and thus is as effective as these two approaches.

7.2.2.4 Application of a Dynamic Exploration Rate to the RL-Agent

The second approach introduced in this chapter involves the adjustment of the exploration rate during a learning task dependent on the number of visits to a state. In general, the probability that the best in a given state has already been learnt by the agent in a stationary environment is higher if the state has been visited more often. Furthermore, most actions in early episodes of problems with optimistic initial values[237] already are mainly explorative. For this reason, an approach that adjusts the value of the exploration rate according to the number of visits to the current state $N(s,a)$ as proposed by VAN HASSELT[238] is applied to the RL-agent. The adjusted exploration rate, called *dynamic exploration rate*, yields:

$$\varepsilon(s) = \frac{1}{N(s)^p} \qquad \textit{Eq. 7.29}$$

Selecting the parameter $p < 1$ hinders the exploration rate to decrease too quickly. VAN HASSELT[238] proposes a value for the exponent of $p = 0.5$. In this work, dynamic exploration rates with $p = 0.5$ and $p = 0.8$ are proposed and the results compared. The corresponding learning curves are contained in Fig. 7.14 and 7.15, respectively.

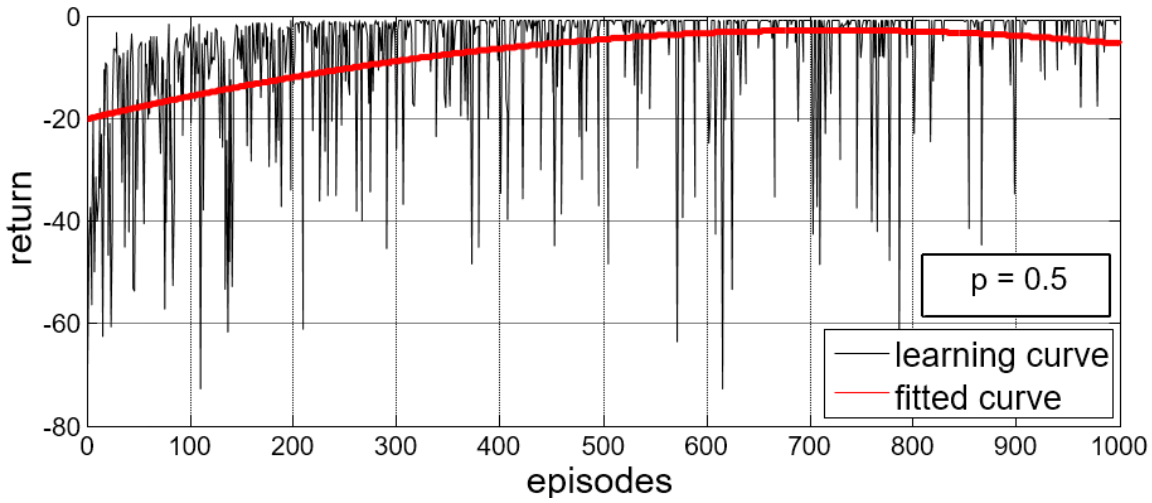

Figure 7.14: Learning curve for the RL-agent with a dynamic exploration rate (p=0.5) on the simulated test bench environment

[237] cf. 5.4

[238] van Hasselt 2010. A different approach at a dynamic exploration rate is proposed e.g. by Tokic 2010.

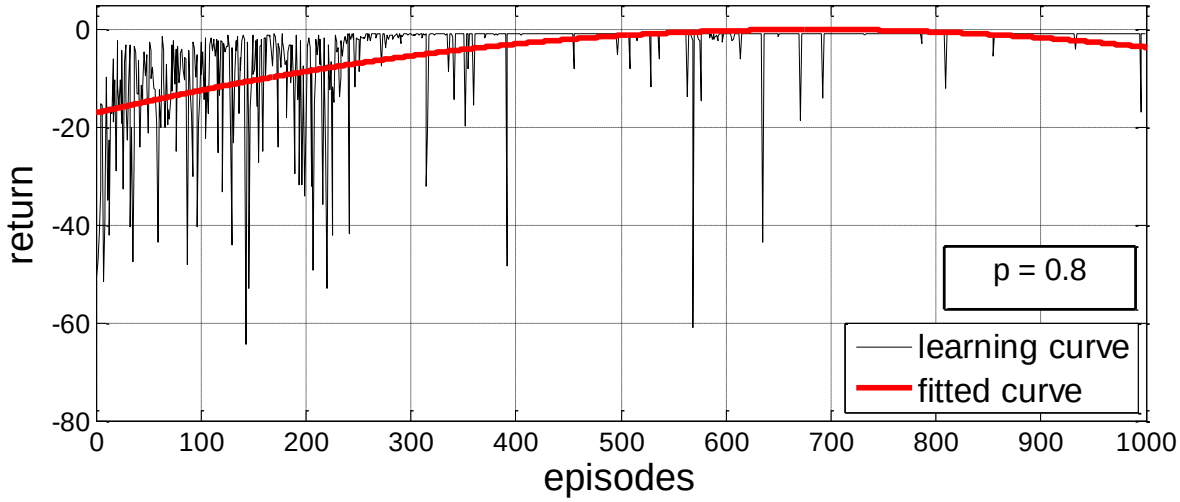

Figure 7.15: Learning curve for the RL-agent with a dynamic exploration rate (p=0.8) on the simulated test bench environment

In comparison to the previous results, the effectiveness of the dynamic exploration rate is nearly the same, suppressing vibrations by 93.4% for $p = 0.5$ and 93.3% for $p = 0.8$. However, it converges slower than the previous approaches. Nevertheless, the reduced number of explorative actions in late episodes, where most of the relevant states have been visited often enough, leads to an evident reduction of bad episodes with exacerbated vibrations. The agent with $p = 0.5$ converges to the exact same action sequence that resulted from the Q-Learning and polynomial learning rate. This action sequence leads the greedy episode depicted in Fig. 7.9. The best action sequence found by the agent with $p = 0.8$ is slightly different and leads to the greedy episode depicted in Fig. 7.16. However, no noteworthy differences can be regarded at plain sight.

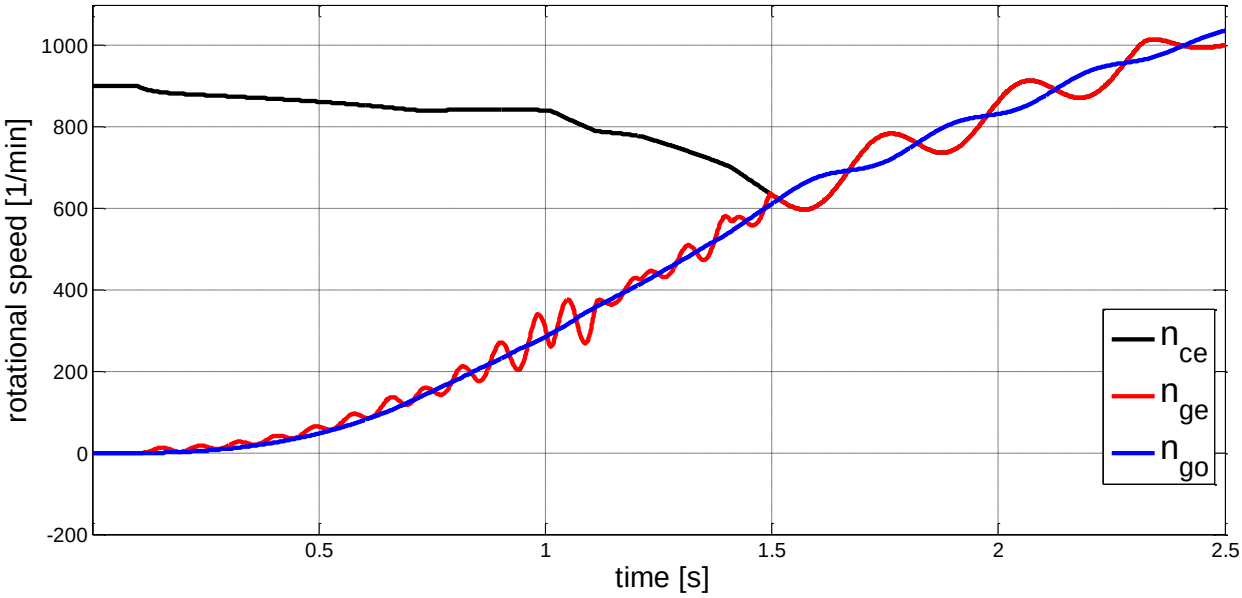

Figure 7.16: Greedy episode with RL-agent with a dynamic exploration rate (p=0.8)

7.2.3 Evaluation of the implemented Approaches

In this section, an overview of the results of the algorithms presented thus far is provided. As mentioned earlier, the selection of an algorithm for the implementation on the physical test bench environment seeks to find a favorable solution to the trade-off between the quality of a solution and the effort necessary for learning it, i.e. the effectiveness of judder suppression of an algorithm and the number of episodes it requires to learn the corresponding policy. The measure for the effectiveness of clutch judder suppression in the context of this work is the value of the reward-signal defined in 5.4, whereas the obvious measure for the learning speed of an algorithm is the number of episodes it requires to converge towards the best solution[239].

The comparison of the results presented thus far in this chapter is made on the basis of the numeric values contained in Table 7.5.

Q-Learning algorithm	Return		First episode with best greedy return	Episodes within tolerance[240] after convergence [%]
	Maximum	Greedy		
Classic	-0.99	-0.99	208	82.7
With Gaussian-update	-1.01	-1.42	137	74
With dyn. α $(\mathrm{p} = 1)$	-0.86	-0.99	51	81,2
With dyn. α $(p = 0.8)$	-0.86	-0.99	105	85,8
With dyn. ε $(p = 0.5)$	-0.99	-0.99	260	83,5
With dyn. ε $(p = 0.8)$	-0.99	-1.00	240	96,8

Table 7.5: Results of the implementation of the RL-algorithms on the simulated test bench environment

There are two return columns in Table 7.5. The first indicates the highest return achieved during the learning task, whereas the second value indicates the value of the converged return. Discrepancies between the maximum return and the converged value may occur due to the violation of the Markov property as a result of e.g. state discretization and the simulated dead times. The next column contains the number of the first greedy episode that yielded the highest return. In the last column of Table 7.5, the percentage of episodes after the first greedy episode whose value stayed within a predefined tolerance range[240] around the greedy return is specified. This last value

[239] Algorithms that do not satisfy the Markov property entirely may converge towards a suboptimal policy (cf. 2.2.3).

[240] An episode is considered to be within the tolerance range if the value of its return is within ± 5 of the greedy return after convergence.

serves as a means to assess the stability of the solutions, i.e. to avoid the occurrence of outlier episodes with high amplitude vibrations.

The fastest learning approach is the one involving a linear dynamic learning rate. At the same time, it is the approach that yields the best return after convergence. In comparison to the classic Q-Learning algorithm, it achieves the best greedy run in only 14.9% of episodes. However, it is not as stable as the agent with a dynamic exploration rate where $p = 0.8$. The latter is able to avoid bad episodes with exacerbated vibrations due to its more targeted selection of explorative actions. Nevertheless, due to the considerably faster learning speed, convergence is reached in less than a fourth the number of episodes, and the effectiveness of judder suppression, the algorithm with an agent with a linear learning rate is selected for the implementation on the physical test bench.

7.3 RL Framework on the physical Test Bench

Having selected the elements of the framework to be implemented on the physical test bench, establishing a benchmark synchronization maneuver for comparison represents the first task in the new environment.[241] This represents a slightly more difficult task than it did for the simulated environment, in which only very small fluctuations due to the dead time in the actuation of the clutch were present. Due to its stochastic nature, in the physical environment, the same input action sequence performed twice generally leads to two synchronization maneuvers that, while similar, usually end up being significantly different. In order to serve as a base for comparison, a standard actuation force ramp is used to actuate the clutch and the resulting synchronization maneuver is evaluated in terms of the return a RL agent would have acquired.[242] However, the gradient determined in simulation might prove to be too steep on the physical test bench, leading to a hasty closing of the clutch and a short synchronization time, at the time causing high stress to the side shaft and high amplitude vibrations. Therefore, two more gradual progressions of the standard force ramp are proposed and the results of each 15 synchronization maneuvers are compared. These results are graphically represented in the form of their boxplots in Fig. 7.17.

[241] cf. 7.2.2
[242] cf. 7.2.1.3

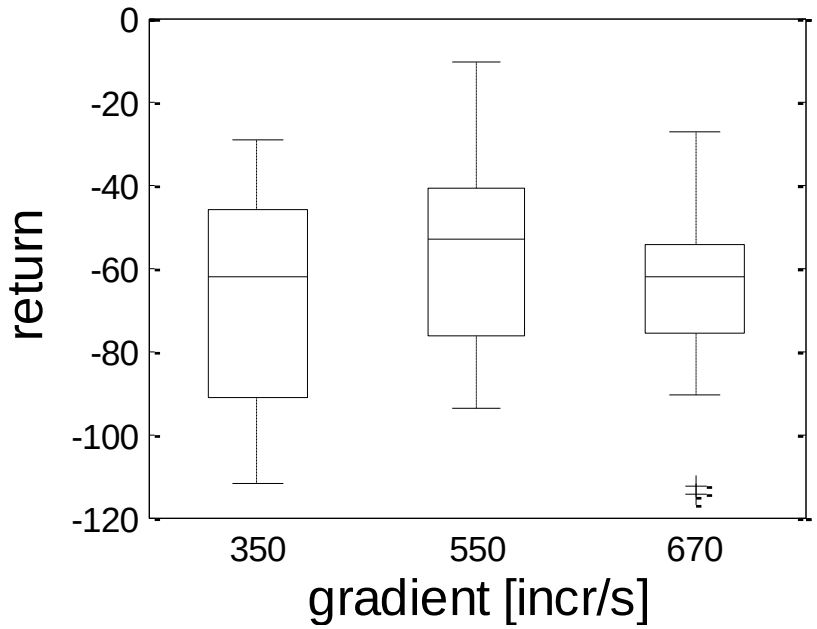

Figure 7.17: Distribution of the return acquired throughout 20 synchronization maneuvers for three different gradients of the standard force ramp

There are basically two major pieces of information that can be observed in these boxplots. The first is the degree to which the resulting returns for each of the proposed gradients are scattered, and the second piece is the value around which the scattering takes place. The most gradual of the ramps shows both the widest range across which the results of each maneuver are scattered and the lowest median value of the return at around -62, thus the strongest judder. The latter can easily be explained by the longer duration of synchronization because of the gradual progression of the force ramp. The boxplot in the middle corresponds to the actuation ramp with a gradient of $550\ incr/s$ and shows a similar degree of dispersion of the return values, but the highest median value, at around -53 of the proposed progressions of the force ramp. Finally, the steepest of the force ramps with a gradient of $670\ incr/s$ achieves a median return of approx. -62 with a significantly limited dispersion of its values. However there are two outlier values marked by crosses at the bottom left of the figure. Once an overstressing of the side shaft[243] could be ruled out and due to the significantly lower dispersion at almost the same median return, the standard ramp with a gradient of $670\ incr/s$ is selected for the implementation on the physical test bench. A sample episode of the synchronization maneuver following the selected force ramp is depicted in Fig. 7.18.

[243] c.f. 7.1.2.3 and 7.3.1.2

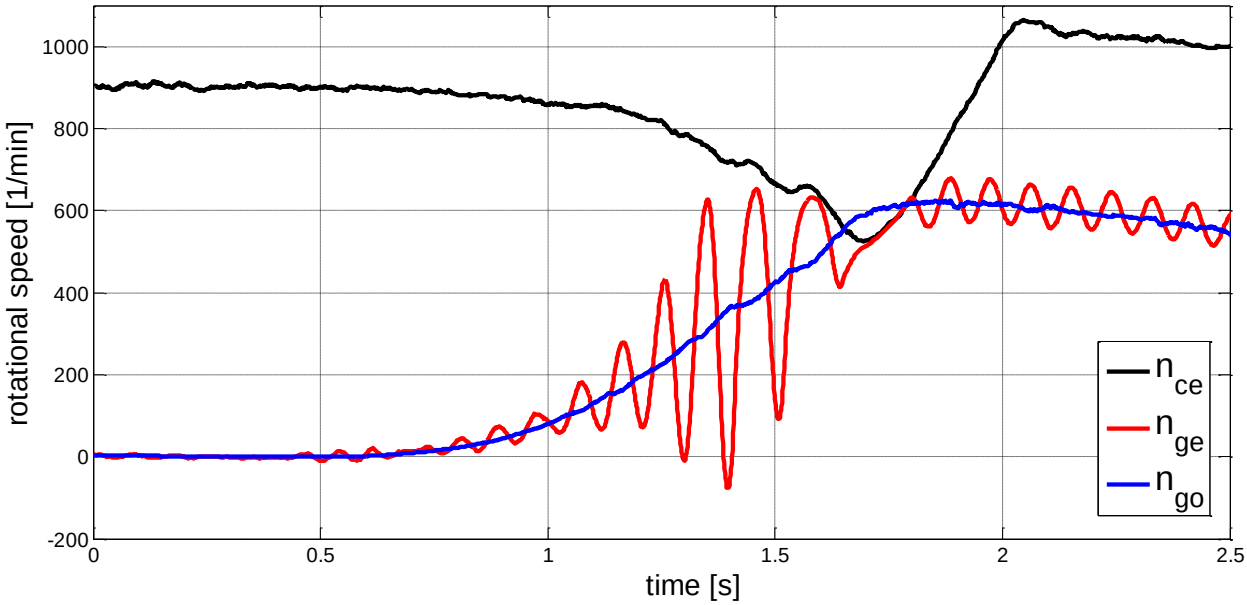

Figure 7.18: Standard synchronization maneuver following a standard force ramp on the physical Mini-HiL test bench

The synchronization maneuver on the physical test bench is noticeably shorter. Its duration is of approx. $1.2\ s$ as opposed to the approx. $1.5\ s$ in simulation. After the synchronization maneuver is completed, the clutch is opened and the speeds of the output side decelerate back to zero, whereas the engine accelerates back to its setpoint value of $900\ 1/min$.

7.3.1 Consideration of the physical Test Bench Environment

The physical test bench is subject to stochastic fluctuations in the values of the relevant parameters for the RL agent that cannot be taken into account, without at least considerable effort to measure and/or model them. An example of such factors is changes in the geometry of the test bench components due to different temperatures or heat produced in operation. Furthermore, making all information available for the agent at the required time poses another difficulty regarding a physical implementation. In this section, a way to account for three of the most important impediments is presented. The first regards the availability of the torsion angle signal for the agent at the required time, since it cannot be measured directly on the test bench. The second is a consideration of the relatively weakly dimensioned side shaft. Finally, the effect of general fluctuations on the position of the linear clutch actuator in relation to the remaining test bench elements is considered. This has a direct effect on the implementation of the action space and, therefore, on the possibilities of the agent to influence its environment.

7.3.1.1 Availability of the Torsion Angle Signal

The torsion angle θ is one of the three state variables that form the state-signal.[244] However, it is not possible to measure it directly, i.e. with a specific sensor, on the physical test bench. Instead, a discrete integration of the difference in rotational speeds between the gearbox entry shaft n_{ge} and the output speed n_{go} is computed. This poses no particular challenge in simulation where high quality signals can be sampled at very high rates and then need not be processed in real time. However, on the physical drive train the signal is noisy, cannot be sampled at an arbitrarily high rate and any calculation made with the signal values needs to be made in real time. This presents a problem in the implementation of the numeric integration method for the approximation of θ. SCHWENGER[245] states that a very high quality signal and a very high sampling rate are necessary for the approximation method to deliver satisfactory results. In this work, two further possibilities to determine the torsion angle are available. The first is the direct measurement of the clutch torque M_C.[246] with the integrated torque sensor in the test bench setup. Since M_C correlates linearly with the clutch torque, it could substitute the torsion angle as a state variable. However, since such a measurement unit is seldom available in commercial drive trains, a third possibility is proposed. Using the clutch torque as an input variable for a simulation model of the test bench, the torsion angle θ is computed. The results of the torsion angle approximation, the direct measurement of the clutch torque and the simulated torsion angle are depicted in Fig. 7.19

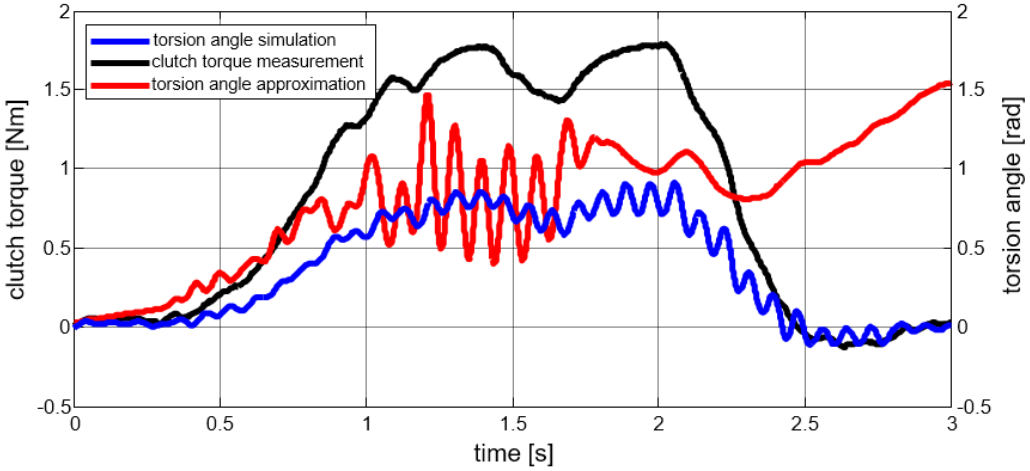

Fig. 7.19: Comparison of the measured clutch torque, the approximated and the simulated torsion angles

[244] cf. 5.2

[245] Schwenger 2005

[246] The measurement unit actually measures the torque before the clutch, however, due to the relatively small inertia of the clutch discs in comparison to the inertia of the load (approx. 1:100)

The most stable value is that of the measured torque, which undergoes barely noticeable fluctuations. The value of the approximated torsion angle using the rotational speed measurement units of the test bench only barely resembles the course of the torque curve. When the torque decreases abruptly, the approximated angle skyrockets to implausible values. However, this is due to the fast decrease of the clutch torque after the latter is opened. At this time the RL-agent is no longer active and thus the value of the torsion angle is irrelevant for the task. Finally, the simulated value of the torsion angle correlates relatively well with the torque, even though it seems to oscillate moderately despite a nearly constant torque. The torque signal represents the more reliable and reproducible signal and could be used to replace the torsion angle in the state-signal. However, in order to retain the structure of the state space proposed throughout this work and due to the previously mentioned scarcity of adequate measurement units in conventional drive trains, the state-signal of the physical environment made available to the agent is implemented with the value of the simulated torsion angle. The other two state variables remain the rotational speeds n_{ge} and n_{go}.

7.3.1.2 Torque Restriction due to the Strength of the Side Shaft

In the Mini-HiL test bench, the stiffness and damping of the drive train are physically modeled by means of a torsion shaft with a diameter of $3\,mm$. This small diameter is necessary to ensure that the eigenfrequency of the physical oscillations on the test bench coincides with those for an unscaled drive train. However, it causes the shaft to be compromised due to high stress at higher clutch torques.

Also, constructive constraints led to the shaft being fixed to the remaining components by means of four grub screws on each end. In order to provide a flat surface for the screws to press on, the ends of the shaft were additionally flattened (cold pressed), strain hardening in the area. The localized high stress areas in the shaft due to the hardening and the point load from the screws led the material to fail in one experiment after approx. 450 episodes. A view at the development of the eigenfrequency of the system throughout this experiment reveals that even before the shaft finally broke, changes in the oscillation system could already be observed as the eigenfrequency appears to slowly decrease before suddenly collapsing shortly before failure. The course of the eigenfrequency of the test bench until its failure can be taken from Fig. 7.20. A first degree polynomial fit of the first 300 episodes, thus ignoring the strong outliers at approx. 350 episodes and just before the collapse, confirms this suspicion as it presents a negative gradient.

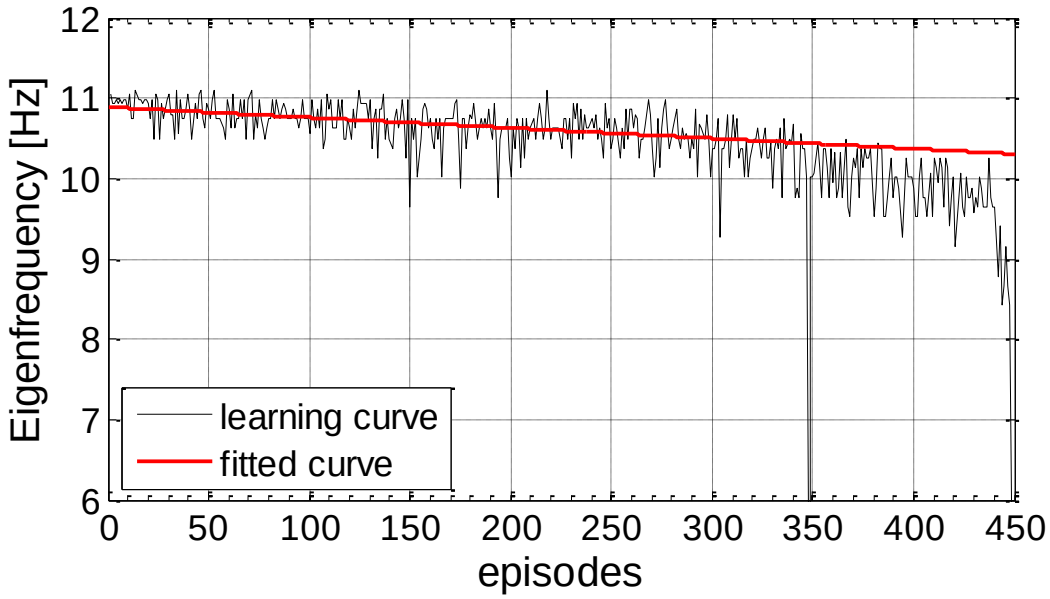

Figure 7.20: Development of the eigenfrequency of judder vibrations until failure of the torsion shaft

While the conception of the test bench proved to be adequate for most other applications, the high number of test cycles required for the RL implementation made it necessary to adopt measures to avoid a repeated failure. However, due to time, cost and available space constraints, no major constructive changes were undertaken. The first measure consisted of milling material from the ends of the torsion shaft so the grub screws could press against a flat surface. This prevents the shaft from failing due to the strain hardening as a result of cold pressing. Additionally, a high-strength adhesive was applied to the grub screws to secure them.

Since determining the fatigue strength of the shaft exactly proves a difficult task, the maximum torque was reduced in order to operate in a more secure range.[247] Although a conservative estimation of the strength suggests that the shaft is only compromised when a torque of at least $2.5\ Nm$ is applied, the current of the input DC motor is limited so the maximum input torque yields $1.7\ Nm$. Furthermore, the clamping force applied by the clutch actuator is limited so it provides a maximum of $2\ Nm$ clutch torque. Whereas the torque in the shaft can exceed this value to the vibrations in the system, simulation results showed that exceeding torque does not compromise the function of the shaft.

7.3.1.3 Position of the Action Space

The last consideration made for the implementation of the RL framework in the physical test bench environment regards the application of the agent's actions to the environment. In simulation, the actuator position at the time the clutch disks establish

[247] c.f. 7.1.2.3 and Gwosch 2011

contact, often called the kiss-point, is always known and perfectly reproducible. On the drive bench, fluctuations to the position of the kiss-point are observed. However, the position of the kiss-point determines the start of a maneuver as introduced in 7.1.2.3. Nevertheless, on the physical test bench the kiss-point constitutes more of a range than a point due to diverse factors: e.g. tilt of the clutch discs or fluctuation in distances due to thermal expansion. This would not represent a problem, if the agent would specify forces as actions, since the axial travel necessary to produce the force would be irrelevant. However, a force control of the actuator has been ruled out, as specified in 7.2.1.3. Although the actual clamping force value as a result of an action determined by the agent is not really important (provided the action space covers the relevant range), the reproducibility is. This means that the effect of an action the agent selects is relatively trivial as long as it is always the same. Therefore, the actions by the agent in the form of incremental positions should induce the same, or at least a similar, clamping force.

In order to establish a reproducible point in the maneuver and given the fact that establishing the position of the kiss-point is relatively difficult, a different solution is proposed. Instead of using the kiss-point as a reference, the incremental position at the time the agent is activated in the maneuver, when the criterion expressed in *Eq. 7.23* is fulfilled, is used to determine the lowest selectable action, which is the action space's lower boundary. The boxplot of the distribution of the position of the lower boundary of the action space over the course of a hundred maneuvers is depicted in Fig. 7.21.

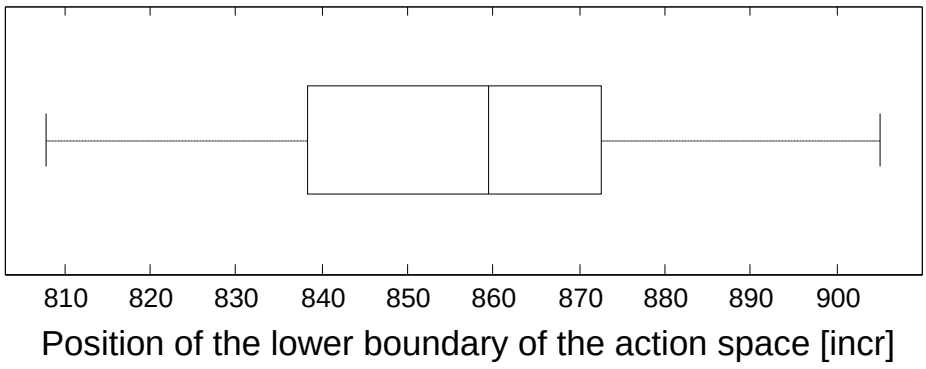

Figure 7.21: Distribution of the position of the lower boundary of the action space on the physical test bench

The median position of the lower boundary yields approx. 860 increments. However, the fluctuation of the determined boundary is scattered in a range of almost 100 increments ($0.1\,mm$), with the upper and lower quartiles covering a range of 40 increments ($0.04\,mm$). Bearing these results in mind, the position of the lower boundary of the action space was defined to be at 850 increments. Retaining the dimension of 100 increments of the action space in the simulated environment, the

upper boundary was defined to be at 950 increments. The five actions are equidistantly[248] distributed in the action space. Thus the new action space yields:

$$\mathcal{A} = \{850\ incr, 870\ incr, 900 incr, 920\ incr, 950\ incr\} \qquad \textit{Eq. 7.30}$$

7.3.2 Results and Evaluation of the RL Framework

Having described all necessary adjustments of the elements of the RL framework to the physical environment, the results of its implementation are presented and subsequently discussed.

7.3.2.1 Results of the Implementation

The selected RL-approach of 7.2 with the modifications to the RL framework described in the recent sections was implemented on the physical Mini-HiL test bench. The Q-Learning agent with a linear learning rate and the ε-greedy action selection method as described in *Eq. 7.26* performs 1000 learning episodes on the test bench. The gradient of the force ramp in the phase before the agent is activated is $670\ incr/s$ until the lower boundary of the action space (850 increments) is reached. The value of the simulated torsion angle is used in the state-signal as described in 7.3.1.1. Also, the torque input and clutch torque restrictions described in the previous section are applied. The results of the implementation of the presented RL framework in the physical test bench environment are illustrated in Fig. 7.22 and Fig. 7.23 in the form of its learning curve and a sample episode.

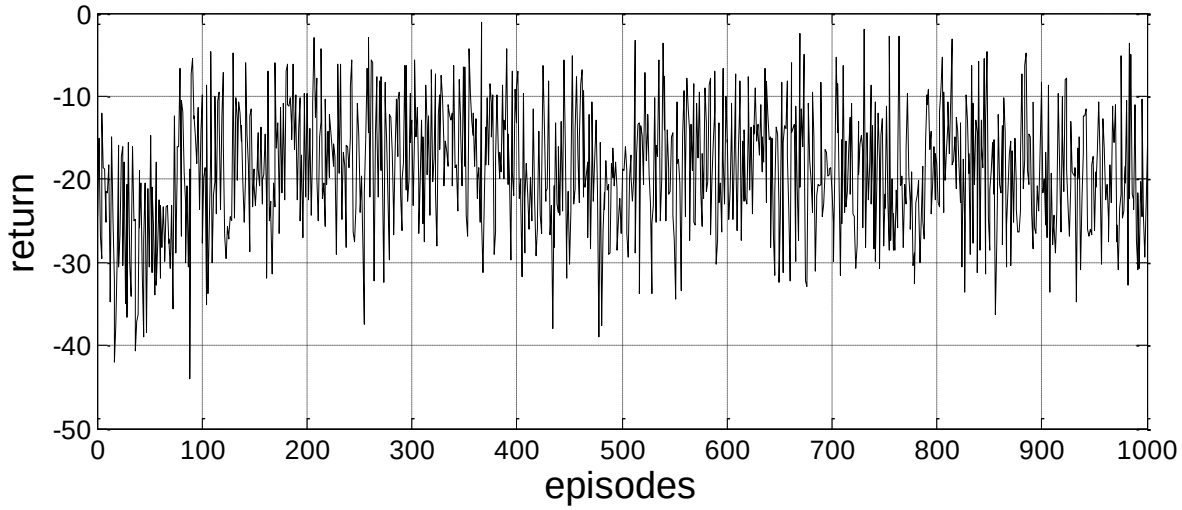

Figure 7.22: Learning curve of the implementation of RL framework in the physical test bench

[248] The actions are rounded to the nearest ten.

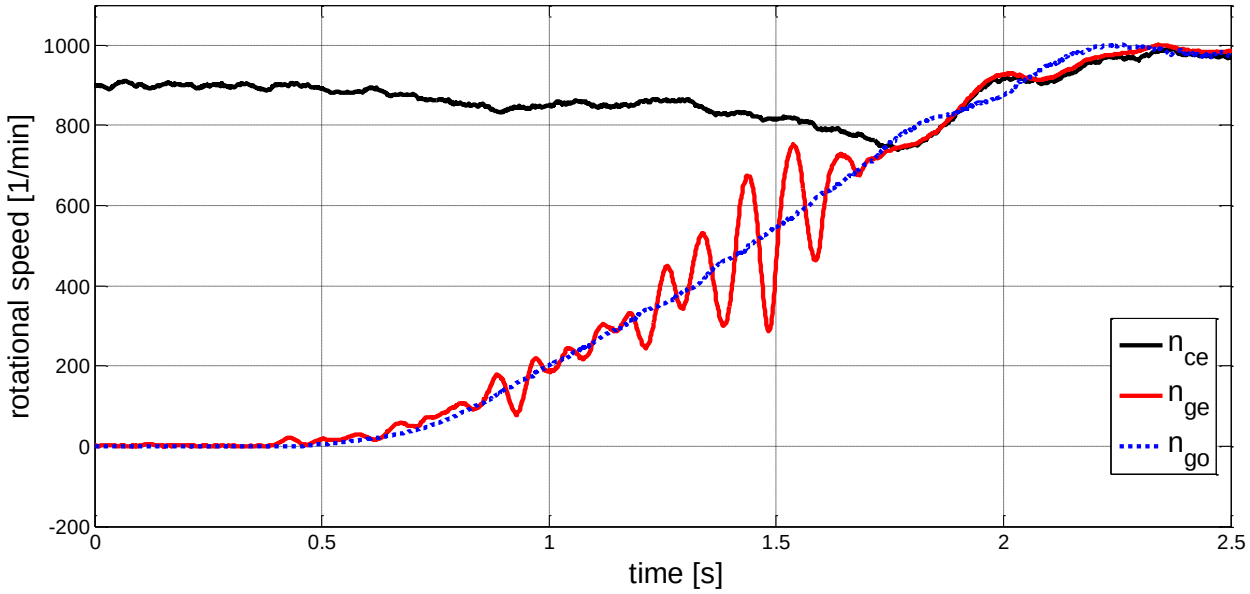

Figure 7.23: Sample episode of the RL framework implementation on the physical test bench with approx. median return of last 100 episodes (return = 19.34)

The completion of the 1000 episodes took 3 hours and 10 minutes. In contrast to the results in the simulated test bench, the learning behavior only appears significant throughout the first hundred episodes. Afterwards no real improvement of the return over the course of the episodes is noticeable at first sight. However, the average return seems to already be better than that of the standard force ramp from the beginning. In the following section the results are evaluated in detail and an analysis of probable impediments of the learning ability are presented.

7.3.2.2 Evaluation and Discussion of the Results of the Implementation

As stated previously, the results throughout the 1000 episodes on the physical drive train are not as good as the results on the simulated test bench. However, even though the results are relatively unsatisfying, a more differentiated look is necessary to put the performance of the agent in the RL framework into perspective. Particularly, it becomes necessary to compare in a statistical sense the policy the agent derived from its 1000 episodes of experience with the results of the standard engagements presented earlier in this subchapter and depicted in Fig. 7.17. Apparently, the agent does improve its behavior in the first few hundred episodes (see Fig.7.24) and this could prove to have a lasting effect in the overall performance afterwards.

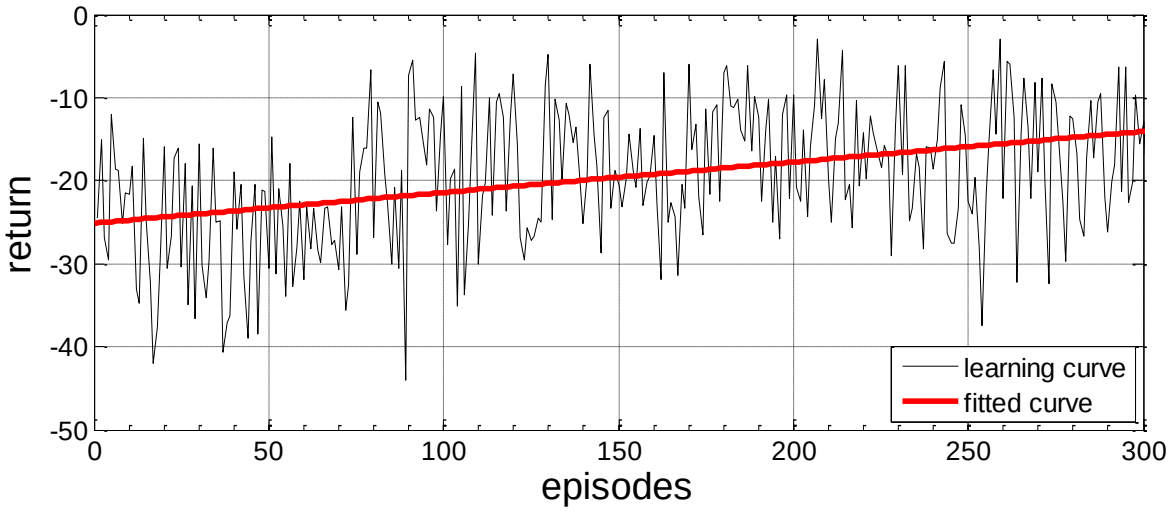

Figure 7.24: Cut-out of the first 300 episodes of the learning curve and first order polynomial fit

For this reason, the distribution of the results of the last 100 episodes of the task is compared to the results of the standard ramp synchronization maneuver depicted in Fig 7.17 in order to determine if an improvement could be achieved. The corresponding boxplots can be taken from Fig. 7.25.

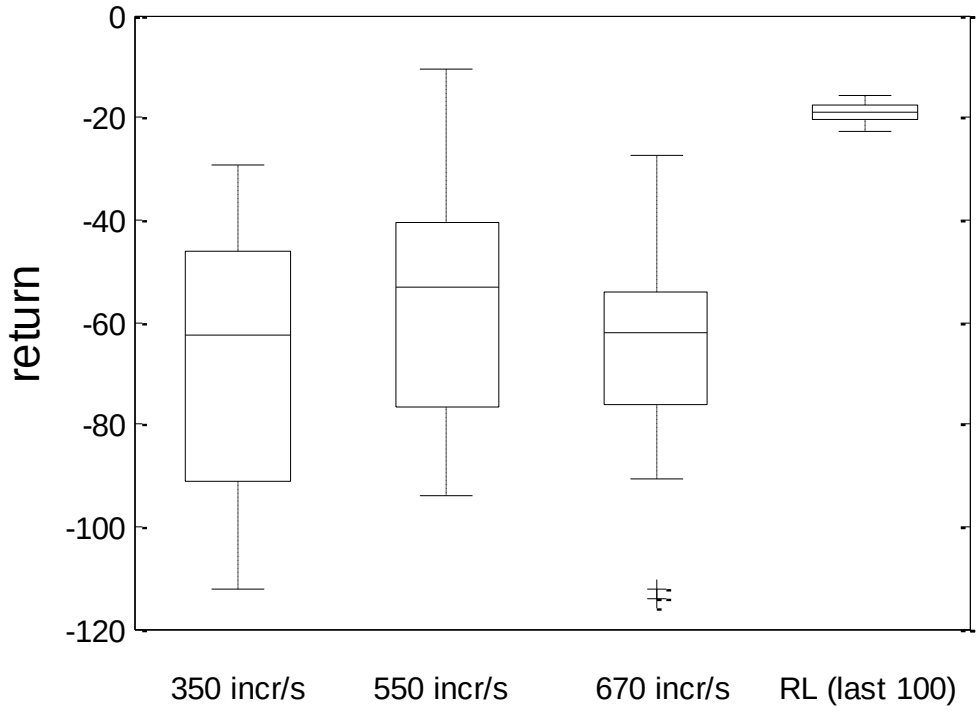

Figure 7.25: Comparison of the distribution of the return after synchronization maneuvers with standard force ramps and the RL framework

The reproducibility of the results of the RL controlled clutch synchronization maneuver is vastly superior to the results achieved with standard force ramps. Furthermore, the median value of the return is substantially higher. Measured in terms of the acquired

reward and compared to the highest median value of the standard ramps ($550\ incr/s$), the RL-agent was able to reduce vibrations by approx. 61.3%. Nevertheless, the effectiveness of judder suppression is lower than the effectiveness achieved in simulation as can be seen in Fig. 7.23, where judder vibrations are still clearly present. A tabular overview of the information expressed by the boxplots in Fig. 7.25 is contained in Table 7.6.

Clutch actuation	Median return	Mean return	Best episode	Worst episode	Standard deviation
Standard (350 incr/s)	-62.4	-66.3	-29.1	-112.1	27.7
Standard (550 incr/s)	-53.2	-58.2	-10.7	-94.2	24.5
Standard (670 incr/s)	-62.0	-66.5	-27.2	-114.2	25.3
RL-agent	**-20.6**	**-21.4**	**-7.0**	**-42.3**	**6.9**

Table 7.6: Statistical comparison of experimental results

The RL-agent controlled clutch synchronization outperforms the standard force ramps clearly in every category. This is evident in either the mean or median return, given the data is fairly symmetric[249], and the much lower standard deviation of the RL results. In this context, the implementation of the proposed RL framework successfully learnt to suppress clutch judder significantly.

Regarding the learning time, the slightly more than three hours it takes to complete the 1000 learning episodes pale in comparison to the minimum operation time mentioned in chapter 3. Furthermore, the learning procedure could also be shortened to 500 or fewer episodes, as the progression of the learning curve indicates.

[249] The statement that the data is symmetric is justified by the similitude in the mean and median values of the data sets.

8 Summary and Outlook

In this concluding chapter, a synthesis of the results of this work, as well as an outlook on them are presented. The latter comprises a discussion of the limitations of the results of this work and suggestions on future research in order to address them.

8.1 Summary

After introducing the fundamentals on both clutch judder and reinforcement learning, the state of the art research was presented. From the latter, a need for a cost effective countermeasure for clutch judder was noticed. Among the mechatronic countermeasures that satisfy this requirement, a reliance of most of them on accurate and often complex vehicle or drive train models was determined. Additionally, even if such dependable models exist for certain components or subsystems of the vehicle in early phases of the development process, the behavior of the whole system often deviates from that predicted by said models. The same uncertainty applies to the transfer of models that accurately describe the behavior of an individual system, even a whole system like a prototype, to further exemplars. Therefore, the great potential of a mechatronic countermeasure for judder capable of self-adapting to any given system was determined.

In this work, such a countermeasure is proposed in the form of a reinforcement learning (RL) framework for the task of judder suppression. In order to realize this proposition, the clutch judder task and the clutch synchronization maneuver were formulated as a RL task. Afterwards, the components of the elementary RL framework were applied to this newly formulated task, after the corresponding counterparts had been properly identified. In this fashion, a flexible and highly abstract RL framework for the judder suppression task was provided.

Subsequently, the abstract RL framework was adapted in order to be applied to three different representations of the environment throughout two experimental phases. In the first stage, the first representation consisted of an abstract simulation model of the drive train in the form of a three-mass rotational oscillator. In this stage, the general suitability of the RL framework to suppress judder vibrations was assessed and different proposals to enhance the learning process are proposed.

In the second experimental stage, the RL framework was applied to the Mini-HiL test bench at IPEK. In this stage, the application of the framework to representations of the environment closer to a physical vehicle is pursued. First, the framework is applied to a simulation model of the test bench. At this point, different realizations of the elements of the framework are undertaken and their performance is assessed on the basis of

their suitability for an implementation on the physical test bench. Finally, the necessary adjustments are made in order to apply the most promising conception of the framework elements to the physical test bench itself.

The different implementations of the RL framework on the different environments were learnt to suppress judder in an acceptable time frame, albeit not always at the same rate.

8.2 Outlook

This last section is intended for the analysis of the most likely challenges that need to be faced, in order to improve the results of the developed RL framework and for presenting suggestions on how they could be confronted in future research.

8.2.1 Challenges for the RL Framework

Despite the increased reproducibility and the reduction in judder achieved through the implementation of a RL framework, a negative discrepancy between the simulation results and the experimental results on the physical test bench are evident. In this work, the causes for the decreased performance are categorized to either be found in hardware or software. However, they often are interrelated.

The changes in performance during the application of the RL framework to the physical test bench can be mainly attributed to the loss of the Markov property of the environment. However, this can happen due to different reasons. On the one hand, the state-signal could be distorted, e.g. through noise or an inadequate discretization or definition of the state space, which leads to a faulty perception of its environment by the agent. On the other hand, the same could be the case for the actions the agent selects.

Experiments on the simulated test bench environment with additive noise on the state-signal and afterwards on the action-signal determined the latter to have a much stronger influence on the learning behavior. Unfortunately, due to the unavailability of a force controller of the clamping force and the implemented position control, substantial fluctuations in the values of the clamping force for the same actuator position occur as a result. This behavior can be observed in the distribution of the return values presented in chapter 7.3 (e.g. Fig. 1.17) or even more clearly in Fig. 8.1, where two progressions of the relevant speeds during the synchronization maneuver for the exact same course of the clamping force are depicted.

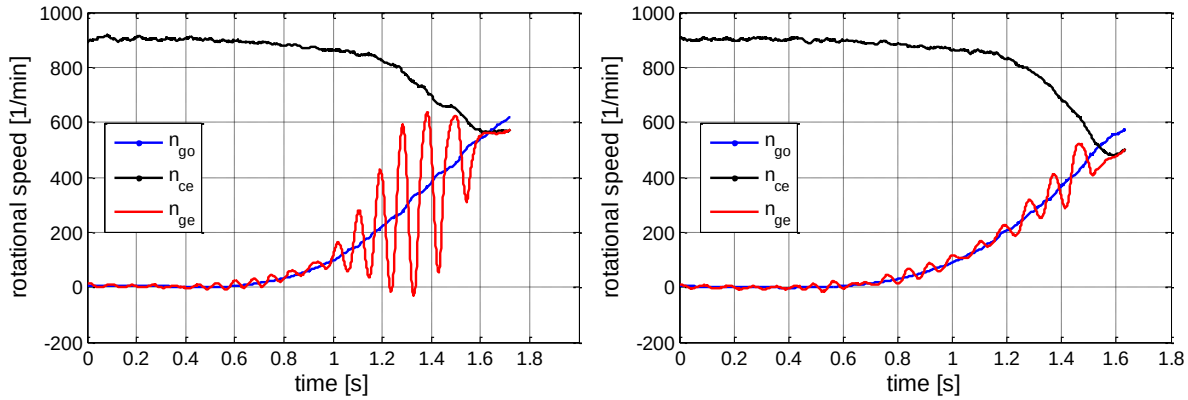

Figure 8.1: Two sample episodes for the exact same progression of the position of the clutch actuator until synchronization ($550\ incr/s$)

The limited reproducibility of the effect of the actions selected by the agent on its environment is considered personally the main challenge on the effectiveness of the RL framework on a physical test bench environment, and therefore, addressing this issue bears the most significant improvement potential.

There are other effects in the physical test bench environment that hinder the RL-agents' performance. However, since simulations show these to have comparatively little effect, only the three most important are listed below:

- Fluctuations of component measurements due to thermal expansion
- Noise and fluctuation of sensor signals within their tolerance
- Fluctuations in the friction coefficient/gradient due to run-in behavior of ceramic friction pairing

The most important software related impediments during the physical implementation are the limitations due to the real time environment in use on the test bench. The need to process information and allocate memory in real time does not always support the calculation methods that were successfully implemented in simulation. This is particularly the case during the implementation of approximation and other methods that rely on updating several values at the same time. Examples of such methods in this work are the Gaussian-Update method, the dynamically discretized state space and the approximation of the value function with RBFs of chapter 6. In order to implement such methods effectively, parallel computing is advised.[250,251]

[250] Sutton / Barto 1998
[251] c.f. 2.2.4.4

8.2.2 Future Research

In regards to the way the limitations presented previously could be addressed in future work, a distinction between hardware- and software-related measures is again preferred.

However, the first and most important suggested measure to undertake is the implementation of a high speed force controlled actuation of the clutch, which is a measure that could be regarded to involve both hardware and software. This would greatly improve the reproducibility of the clutch actuation and therefore the Markov property of the test bench as an environment. At the same time, it would help reduce the effect of thermal expansion, since the force controller would lead to a respective adjustment of the travel path of the actuator.

In this work, the use of ceramic friction pairings was preferred due to their strong tendency to cause judder vibrations. However, their run-in behavior is known to be of stochastic nature and it is not clear if it had already been concluded at the time of the experiments on the physical test bench. The use of pairings with a more reproducible tribological behavior might lead to an improved learning by the agent. Possible friction pairings with such behavior could be of metallic or organic sinter materials.

Regarding further research on the software or the algorithm, the implementation of the RL framework with an agent with a dynamic exploration rate might prove more effective than the implemented linear learning rate. In simulation, this configuration proved to be the most stable, thus the more reproducible, even if at the same time it converged to a slightly lower return value than the implemented linear learning rate. This is probably due to the more conservative approach towards selecting random actions, which quickly lead to bad episodes of strong judder.

Furthermore, the transition from a deterministic to a stochastic environment modelling could prove to deliver better results. For this purpose, the performance of Monte Carlo methods could be studied. However, the amount of necessary learning time is expected to be much higher as stated in chapter 2.2.4.2. Instead, the implementation of eligibility traces might prove a more efficient solution, even if their implementation requires additional measures. A variation of these measures could be applied to the Gaussian-update, dynamic state list and RBN value function approximation methods, which share the same complications during their implementation.

One of the most enticing perspectives is the possibility of combining the use of a robust controller and an appropriate RL framework. As presented earlier in this work, the main drawback of current robust controllers is that they only operate effectively in the individual system (component, prototype or model of them) they were designed for. A RL framework could be used to reconfigure the parameterization of the controller in-use or at a very advanced development stage. This way, the effect of inaccurate

predictions of the behavior of the system during these advanced stages, where the discriminatory power of simulations is usually lower than required[252], can be reduced.

Finally, the application of the developed RL framework on a non-scaled test bench and subsequently a whole vehicle should be pursued.

[252] c.f. Fig. 1.4

9 References

Agrawal / Srikant 1994
Agrawal, R.; Srikant, R.: Fast Algorithms for Mining Association Rules in Large Databases. In: Proceedings of the 20th International Conference on Very Large Data Bases. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc (VLDB ’94), S. 487–499, 1994.

Albers et al. 2001
Albers, A.; Arslan, A.; Herbst, D.: Keramik für den Einsatz in Bremsen und Kupplungen. In: ATZ Automobiltechnische Zeitschrift 103 (2001) H. 5, 2001.

Albers et al. 2005a
Albers, A.; Arslan, A.; Mitariu, M.: Clutches using engineering ceramics as friction materials. In: Materialwissenschaften und Werkstofftechnik, S. 102–107, 2005.

Albers et al. 2008
Albers, A.; Düser, T.; Ott, S.: X-in-the-loop als integrierte Entwicklungsumgebung von komplexen Antriebsystemen. In: 8. Tagung Hardware-in-the-loop-Simulation. Kassel: Haus der Technik, 2008.

Albers et al. 2010b
Albers, A.; Meid, M.; Ott, S.: Avoiding clutch excited judder by using an active clamping force control. In:. Leuven, 2010.

Albers 2010
Albers, A.: Five Hypotheses about Engineering Processes and their Consequences. In: Proceedings of the TMCE 2010, 2010.

Albers et al. 2010
Albers, A.; Behrendt, M.; Ott, S.: Validation - Central Activity to Ensure Individual Mobility. In: FISITA 2010, 2010.

Albers et al. 2011c
Albers, A.; Sommer Obando, H.; Yu, X.: Effects of initial knowledge on reinforcement learning based control. In: Katalinić, B. (Hg.): Annals of DAAAM for 2011 & proceedings of the 22nd International DAAAM symposium "Intelligent manufacturing & Automation: "Power of knowledge and creativity", 23-26th November 2011, Vienna, Austria. Vienna: DAAAM International (Annals of DAAAM & Proceedings), S. 99–100 2011.

Albers et al. 2011d
Albers, A.; Schroeter, J.; Frietsch, M.; Sommer, H.: New Approach for Computation

of predictive, fuel efficient Vehicle Operation Strategies based on a self-learning Algorithm. In:. 11. Internationales Stuttgarter Symposium: ATZlive, 2011.

Albers et al. 2012a
Albers, A.; Behrendt, M.; Ott, S.; IPEK - Institut für Produktentwicklung am Karlsruher Institut für Technologie: Systematisch zu Mobilitätslösungen. In: *Automobil KONSTRUKTION,* 2012.

Albers et al. 2012b
Albers, A.; Lohmeyer, Q.; Ebel, B.: Systems of objectives in complex product development. In: 9th International Symposium on Tools and Methods of Competitive Engineering TMCE 2012, Karlsruhe, Germany, 2012.

Albers et al. 2013e
Albers, A.; Behrendt, M.; Schroeter, J.; Ott, S.; Klingler, S.: X-in-the-Loop: A Framework for Supporting Central Engineering Activities and Contracting Complexity in Product Engineering Processes. In: Proceedings of the International Conference on Engineering Design, ICED 2013. Seoul, Korea, 2013.

Albers et al. 2014
Albers, A.; Fischer, J.; Behrendt, M.; Lieske, D.: Measurement and Interpretation of the Transfer Path of an Acoustic Phenomenon in the Drivetrain of an Electric Vehicle. In: *ATZworldwide,* 2014.

Albers / Albrecht 2002
Albers, A.; Albrecht, M.: Vorhersage subjektiver Komforturteile mittels künstlicher neuronaler Netze. In: VDI Tagung, Berechnung und Simulation im Fahrzeugbau, 11. Internationaler Kongress, Würzburg, 01.-02. Oktober 2002, Tagungsband: VDI-Berichte 1701, VDI-Verlag Düsseldorf 2002, S. 401-420, 2002.

Albers / Albrecht 2004
Albers, A.; Albrecht, M.: Einsatz Künstlicher Neuronaler Netze zur objektiven Beurteilung des Schwingungskomforts am Beispiel des automatisierten Anfahrens. In: VDI Tagung \textquoteleft, Darmstadt, 17.-18. März 2004, Tagungsband: VDI Berichte Nr. 1821, S. 159-182, VDI-Verlag, Düsseldorf, 2004, 2004.

Albers / Düser 2011
Albers, A.; Düser, T.: Validierung im Produktentstehungsprozess. In: Henning, F. und Moeller, E. (Hg.): Handbuch Leichtbau: Hanser Verlag, S. 133–142 2011.

Albers / Herbst 1998
Albers, A.; Herbst, D.: Kupplungsrupfen - Ursachen, Modellbildung und Gegenmaßnahmen. In: VDI-Berichte 1416. Düsseldorf: VDI Verlag (VDI Berichte), 1998.

Albers / Herbst 2000
Albers, A.; Herbst, D.: Schwingungen in Reibkupplungen - Ursachen, Auswirkungen, Abhilfen. In: VDI-Berichte 1568, Düsseldorf (2000) S. 273-295, 2000.

Albers / Krüger 2002a
Albers, A.; Krüger, A.: Reibungserregte Schwingungen in Kraftfahrzeugkupplungen und ihre aktive Dämpfung. In: VDI-Berichte, Bd. 1736, S. 135–152, 2002.

Albers / Krüger 2003b
Albers, A.; Krüger, A.: Aktive Dämpfung friktionskontaktinduzierter Schwingungen in Antriebssträngen. In: VDI-Berichte Nr. 1786, Kupplungen und Kupplungssysteme, S.193-209, ISBN 3-18-091786-5, VDI-Verlag, Düsseldorf, 2003, 2003.

Albers / Krüger 2004c
Albers, A.; Krüger, A.: Aktive Dämpfung selbsterregter Reibschwingungen am Beispiel des Kupplungsrupfens. In: Birkhofer, Feldhusen, Lindemann (Hg.): Konstruktion, Zeitschrift für Produktentwicklung und Ingenieur - Werkstoffe, 05-2004. Dusseldorf: Springer VDI Verlag, S. 71–74, 2004.

Albers / Matthiesen 1998
Albers, A.; Matthiesen, S.: Was bringt die Zukunft? Trends in der Automatisierung von KFZ-Antriebssträngen. In: Veröffentlichung zur VDI-Tagung Getriebe in Fahrzeugen Friedrichshafen 16. und 17. Juni 1998, Verein Deutscher Ingenieure, Düsseldorf, 1998, Deutschland, Bd. 1393. Düsseldorf: VDI Verlag (VDI Berichte), S. 133–158, 1998.

Albers / Meid 2010
Albers, A.; Meid, M.: Advanced ceramic-steel pairings under permanent slip for dry running clutch systems. In: Proceedings of the TAE ICT 17th International Colloquium Tribology Esslingen, 2010.

Albers / Schyr 2003
Albers, A.; Schyr, C.: Simulationsunterstützte Auslegung von Antriebsstrang-Prüfständen. In: *ATZ Automobiltech Z* 105 (6), S. 604–611, 2003.

Albers / Stier 2010
Albers, A.; Stier, C.: Analysis of Geometrical Deviations in Clutch Systems and their Interdependencies in Relation to the Excitation of Judder Vibrations. In: Proceedings of the FISITA World Automotive Congress 2010, 30 May - 4 June 2010, S. 1736–1745, 2010.

Albrecht 2005
Albrecht, M.: Modellierung der Komfortbeurteilung aus Kundensicht am Beispiel des automatisierten Anfahrens. Karlsruhe: Institut für Produktentwicklung Universität Karlsruhe (TH) (Forschungsberichte / IPEK, 18) 2005.

Andersen et al. 2009
Andersen, K. T.; Zeng, Y.; Christensen, D. D.; Tran, D.: Experiments with online reinforcement leaarning in real-time strategy games. In: *Applied Artificial Intelligence* 23 (9), S. 855–871, 2009.

Åström / Wittenmark 2008
Åström, K. J.; Wittenmark, B.: Adaptive control. 2nd ed., Dover ed. Mineola, N.Y.: Dover Publications (Dover books on engineering) 2008.

Bakker et al. 2005
Bakker, B.; Zivkovic, Z.; Krose, B.: Hierarchical dynamic programming for robot path planning. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. Edmonton, Alta., Canada, S. 2756–2761, 2005.

Balakrishna et al. 2010
Balakrishna, P.; Ganesan, R.; Sherry, L.: Accuracy of reinforcement learning algorithms for predicting aircraft taxi-out times: A case-study of Tampa Bay departures. In: *Transportation Research Part C: Emerging Technologies* 18 (6), S. 950–962, 2010.

Barto / Mahadevan 2003
Barto, A. G.; Mahadevan, S.: Recent Advances in Hierarchical Reinforcement Learning. In: *Discrete Event Dynamic Systems* 13 (4), S. 341–379, 2003.

Bauer 1993
Bauer, H.-K.: Optimierungsmethode für naßlaufende Anfahrkupplungen in modernen Kennungswandlern. In: 4. Aachener Kolloquium Fahrzeug- und Motorentechnik. Aachen, S. 563–590, 1993.

Baum 1994
Baum, W. M.: Understanding behaviorism. Science, behavior, and culture. New York, NY: HarperCollins College Publishers (Behavior analysis and society series) 1994.

Behsaz / Safabakhsh 2007
Behsaz, B.; Safabakhsh, R.: NEFRL: A New Neuro-Fuzzy System for Episodic Reinforcement Learning Tasks. In: 2007 Frontiers in the Convergence of Bioscience and Information Technologies. Jeju City, South Korea, S. 819–826, 2007.

Bellman 1957
Bellman, R.: A Markovian Decision Process. In: Indiana University Mathematics Journal, Bd. 4. 6. Aufl., S. 679–684 1957.

Bertsekas 1995
Bertsekas, D. P.: Dynamic programming and optimal control. 3rd ed. Belmont, Mass: Athena Scientific (Athena Scientific optimization and computation series) 1995.

Bishop 1996
Bishop, C. M.: Neural networks for pattern recognition. Reprinted 1996 twice. Oxford: Clarendon 1996.

Bridle 1990
Bridle, J. S.: Training Stochastic Model Recognition Algorithms as Networks can Lead to Maximum Mutual Information Estimation of Parameters. In: D.S. Touretzky (Hg.): Advances in Neural Information Processing Systems 2: Morgan-Kaufmann, S. 211–217 1990.

Bush / Pineau 2010
Bush, K.; Pineau, J.: Treating Epilepsy by Reinforcement Learning Via Manifold-Based Simulation. In: Koyejo, O. und Souvenir, R. M. (Hg.): Manifold learning and its applications. Papers from the AAAI Fall Symposium. Menlo Park, Calif.: AAAI Press (Technical report, FS-10-06), S. 18–19 2010.

Busoniu et al. 2010
Busoniu, L.; Babuska, R.; Schutter, B. D.; Ernst, D.: Reinforcement Learning and Dynamic Programming Using Function Approximators. 1st. Boca Raton, FL, USA: CRC Press, Inc 2010.

Carlock / Fenton 2001
Carlock, P. G.; Fenton, R. E.: System of Systems (SoS) enterprise systems engineering for information-intensive organizations. In: *Syst. Engin.* 4 (4), S. 242–261, 2001.

Colombetti et al. 1996
Colombetti, M.; Dorigo, M.; Borghi, G.: Behavior analysis and training-a methodology for behavior engineering. In: *IEEE Trans Syst Man Cybern B Cybern* 26 (3), S. 365–380, 1996.

Creighton / Nahavandi 2005
Creighton, D.; Nahavandi, S.: Operating policy generation using a reinforcement learning agent in a melt facility. In: Pham, D. T., Eldukhri, E. E. und Soroka, A. J. (Hg.): Intelligent production machines and systems. First I*PROMS Virtual Conference : 4-15 July 2005. 1st ed. Amsterdam, Boston: Elsevier, S. 369–374 2005.

Crook / Hayes 2003
Crook, P.; Hayes, G.: Learning in a state of confusion: Perceptual aliasing in grid world navigation. In: *Towards Intelligent Mobile Robots,* 2003.

Dalamagkidis et al. 2007
Dalamagkidis, K.; Kolokotsa, D.; Kalaitzakis, K.; Stavrakakis, G. S.: Reinforcement learning for energy conservation and comfort in buildings. In: *Building and Environment* 42 (7), S. 2686–2698, 2007.

Dempster / Leemans 2006
Dempster, M.; Leemans, V.: An automated FX trading system using adaptive reinforcement learning. In: *Expert Systems with Applications* 30 (3), S. 543–552, 2006.

Depraetere et al. 2011
Depraetere, B.; Pinte, G.; Swevers, J.: A Reference Free Iterative Learning Strategy for Wet Clutch Control. In: Proceedings of the 2011 American Control Conference. American Control Conference. O'Farrell Street, San Francisco, CA, USA, June 29 - July 01, S. 2442–2447, 2011.

Desjardins / Chaib-draa 2011
Desjardins, C.; Chaib-draa, B.: Cooperative Adaptive Cruise Control: A Reinforcement Learning Approach. In: *IEEE Trans. Intell. Transport. Syst.* 12 (4), S. 1248–1260, 2011.

DIN 1311 Part 1 2002
DIN 1311 Part 1: Schwingungen und schwingungsfähige Systeme Teil 1: Grundbegriffe, Einteilung. In: *DIN 1311*, S. 1–18, 2002.

Dolcini et al. 2010
Dolcini, P. J.; Canudas de Wit, C.; Béchart, H.: Dry Clutch Control for Automotive Applications: Springer-Verlag London 2010.

Drexl 1988
Drexl, H.-J.: Der Torsionsdämpfer in der Kupplungsscheibe. In: *VDI Berichte* 697, S. 133–158, 1988.

Düser 2010
Düser, T.: x -in-the Loop, ein durchgängiges Validierungsframework für die Fahrzeugentwicklung am Beispiel von Antriebsstrangfunktionen und Fahrerassistenzsystemen (Forschungsberichte / IPEK, 47) 2010.

Even-Dar / Mansour 2004
Even-Dar, E.; Mansour, Y.: Learning Rates for Q-learning. In: Journal of Machine Learning Research, Bd. 5, S. 1–25 2004.

Fakih 2004
Fakih, S.: A learning approach to obtain efficient testing strategies in medical diagnosis. Master ThesisUniversity of South Florida, Tampa, FL 33620, USA. Department of Industrial and Management Systems Engineering 2004.

Frietsch 2011
Frietsch, M.: Ein Beitrag zum effizienten Einsatz von Reinforcement Learning zur Steuerung nichtlinearer Systeme am Beispiel eines Zweiachsroboters. Karlsruhe (Forschungsberichte IPEK, 52) 2011.

Görlich 1968
Görlich, D.: Theoretische Untersuchung der Schwingvorgänge beim Schalten von Reibkupplungen, Karlsruhe 1968.

Griffin 1990
Griffin, M. J.: Handbook of human vibration. London, San Diego: Academic Press 1990.

Griffin 2007
Griffin, M. J.: Discomfort from feeling vehicle vibration. In: *Vehicle System Dynamics* 45 (7-8), S. 679–698, 2007.

Guez et al. 2008
Guez, A.; Vincent, R. D.; Avoli, M.; Pineau, J.: Adaptive Treatment of Epilepsy via Batch-mode Reinforcement Learning. In: Proceedings of the Twenty-third AAAI Conference on Artificial Intelligence and the Twentieth Innovative Applications of Artificial Intelligence Conference. 13-17 July 2008, Chicago, Illinois. Menlo Park, Calif.: AAAI Press, S. 1671–1678 2008.

Gwosch 2011
Gwosch, T.: Physische Modellbildung zur Abbildung von Reibschwingungen auf dem Mini-HiL Prüfstand 2011.

Gwosch et al. 2013
Gwosch, T.; Babik, A.; o. Prof. Dr.-Ing. Dr. h.c. A. Albers: Echtzeitimplementierung einer mathematischen Optimierungsmethode zur Minimierung von Rupfschwingungen in Antriebssystemen, Karlsruhe 2013.

Haberfellner 2012
Haberfellner, R.: Systems Engineering. Grundlagen und Anwendung. 12., völlig neu bearb. und erweiterte Aufl. Zürich: Orell Füssli 2012.

Hangos et al. 2001
Hangos, K. M.; Lakner, R.; Gerzson, M.: Intelligent control systems. An introduction with examples. Boston: Kluwer Academic Publishers (Applied optimization, v. 60) 2001.

Haskins 2011
Haskins, C.: Systems engineering handbook. A guide for system life cycle processes and activities. Version 3.2.2. San Diego, Calif.: International Council of Systems Engineering 2011.

Hastie et al. 2009
Hastie, T.; Tibshirani, R.; Friedman, J. H.: The elements of statistical learning. Data mining, inference, and prediction. 2nd ed. New York: Springer (Springer series in statistics) 2009.

Heilig et al. 2002
Heilig, J.; 12, Verkehrstechnik,Fahrzeugtechnik; 515 Zugl: Karlsruhe, Univ. Diss.2002: Instabilitäten in Scheibenbremsen durch dynamischen Reibkontakt. Düsseldorf: VDI Verlag 2002.

Holland 1992
Holland, J. H.: Adaptation in Natural and Artificial Systems. Cambridge, MA, USA: MIT Press 1992.

Hong / Prabhu 2004
Hong, J.; Prabhu, V. V.: Distributed Reinforcement Learning Control for Batch Sequencing and Sizing in Just-In-Time Manufacturing Systems. In: *Applied Intelligence* 20 (1), S. 71–87, 2004.

Hornung 2012
Hornung, C.: Mathematical optimization methods and their application on the NVH phenomenon clutch judder. Bachelor thesisKarlsruhe Institute of Technology, Karlsruhe. IPEK 2012.

Howell et al. 1997
Howell, M. N.; Frost, G. P.; Gordon, T. J.; Wu, Q. H.: Continuous action reinforcement learning applied to vehicle suspension control. In: *Mechatronics* 7 (3), S. 263–276, 1997.

Istituto Italiano di Tecnologia (IIT) **2014**
Istituto Italiano di Tecnologia (IIT): COmpliant HuMANoid Platform (COMAN) 2014. Online verfügbar unter http://www.iit.it/en/advr-labs/humanoids-a-human-centred-mechatronics/advr-humanoids-projects/compliant-humanoid-platform-coman.html, zuletzt geprüft am 22.10.2014.

Jäger Messtechnik GmbH 2014
Jäger Messtechnik GmbH: ADwin - Produktübersicht - ADwin-Pro II - Das modulare System schnellste Echzeit-Lösungen 2014. Online verfügbar unter http://www.adwin.de/de/produkte/proII.html, zuletzt aktualisiert am 06.02.2014, zuletzt geprüft am 04.03.2015.

Jonsson 2008
Jonsson, A.: Hierarchical decomposition in reinforcement learning. Saarbrücken: VDM Verlag Dr. Müller 2008.

Jonsson / Barto 2005
Jonsson, A.; Barto, A.: A causal approach to hierarchical decomposition of factored MDPs. In: Dzeroski, S. (Hg.): the 22nd international conference. Bonn, Germany, S. 401–408, 2005.

Jörg 1988
Jörg, B.: Abbau von Resonanzschwingungen im Antriebsstrang durch Drehschwingungstilger. In: *VDI Berichte* 697, S. 159–172, 1988.

Jürgens / Fischer 1988
Jürgens, G.; Fischer, R.: Vergleich verschiedener Systeme zur Verringerung von Triebstrangschwingungen, S. 233–256, 1988.

Karrar 2009
Karrar, C.: Ein Beitrag zur Entwicklung, Dimensionierung und Prüfung trockenlaufender Kupplungen und Bremsen in Antriebsstrangsystemen im Hinblick auf die Vermeidung von Reibschwingungen (Forschungsberichte / IPEK, 35) 2009.

Kohl / Stone 2004
Kohl, N.; Stone, P.: Policy gradient reinforcement learning for fast quadrupedal locomotion. In: IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004. New Orleans, LA, USA, S. 2619–2624 Vol.3, 2004.

Kohonen 1982
Kohonen, T.: Self-organized formation of topologically correct feature maps. In: *Biol. Cybern.* 43 (1), S. 59–69, 1982.

Kooy 2014
Kooy, A.: Auf die Isolation kommt es an: Die Evolution des Fliehkraftpendels nicht nur für ZMS: Schaeffler Kolloquium. In: Schaeffler GmbH (Hg.): Schaeffler Kolloquiumsbuch 2014, S. 78–92 2014.

Kormushev et al. 2011
Kormushev, P.; Ugurlu, B.; Calinon, S.; Tsagarakis, N. G.; Caldwell, D. G.: Bipedal walking energy minimization by reinforcement learning with evolving policy parameterization. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011). San Francisco, CA, S. 318–324, 2011.

Krause 1965
Krause, R.: Selbsterregte Reibungsschwingungen bei Kupplungslamellen eine experimentelle Untersuchung, Karlsruhe 1965.

Kroll 2013
Kroll, A.: Computational Intelligence. Eine Einführung in Probleme, Methoden und technische Anwendungen. München: Oldenbourg 2013.

Krüger 2003
Krüger, A.: Kupplungsrupfen - Ursachen, Einflüsse und Gegenmaßnahmen-. Karlsruhe: Universität Karlsruhe, Institut für Maschinenkonstruktionslehre und Kraftfahrzeugbau (Forschungsberichte / IPEK) 2003.

Lauer / Riedmiller 2007
Lauer, M.; Riedmiller, M.: Generalization in Reinforcement Learning and the Use of Observations-Based Learning 2007. Online verfügbar unter http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.617&rep=rep1&type=pdf.

Lenz et al. 1998
Lenz, M.; Bartsch-Spörl, B.; Burkhard, H.-D.; Wess, S.: Case-based reasoning technology. From foundations to applications. Berlin, New York: Springer (Lecture notes in computer science Lecture notes in artificial intelligence, 1400) 1998.

Lerspalungsanti 2010
Lerspalungsanti, S.: Ein Beitrag zur Modellierung des menschlichen Komfortempfindens und Beurteilung der NVH-Eigenschaften in der Antriebsstrangentwicklung auf Basis von Künstlichen Neuronalen Netzen. Karlsruhe: IPEK (Forschungsberichte / IPEK, 60) 2010.

Luce 1959
Luce, R. D.: Individual choice behavior. A theoretical analysis 1959.

LuK GmbH & Co. KG
LuK GmbH & Co. KG: Arbeitsvermögen und Lebensdauer von Kfz-Kupplungen: LuK Kolloquium.

Magalhaes Barros Netto, S. et al. 2008
Magalhaes Barros Netto, S.; Rodrigues coelho Leite, Vanessa; Correa Silva, A.; Cardoso de Paiva, Anselmo; de Almeida Neto, Areolino: Application on Reinforcement Learning for Diagnosis Based on Medical Image. In: Reinforcement Learning. I-Tech Education and Publishing, S. 379–398 2008.

Markov / Nagorny 1988
Markov, A. A.; Nagorny, N. M.: The Theory of Algorithms: Springer (Mathematics and its Applications (Kluwer Academic).: Soviet Series) 1988.

MathWorks
MathWorks: MATLAB and Simulink for Technical Computing. Online verfügbar unter http://www.mathworks.com/, zuletzt geprüft am 02.01.2015.

Maucher 1990
Maucher, P.: Kuplungsrupfen: Möglichkeiten zur Vermeidung. In: LuK (Hg.): LuK-Kolloquium 1990. Bühl, S. 103–117 1990.

Mitchell 1997
Mitchell, T. M.: Machine Learning. New York: McGraw-Hill (McGraw-Hill series in computer science) 1997.

Mohri et al. 2012
Mohri, M.; Rostamizadeh, A.; Talwalkar, A.: Foundations of machine learning. Cambridge, MA: MIT Press (Adaptive computation and machine learning) 2012.

Moore 1991
Moore, A. W.: Knowledge of knowledge and intelligent experimentation for learning control. In: IJCNN-91-Seattle International Joint Conference on Neural Networks. Seattle, WA, USA, 8-14 July 1991, S. 683–688, 1991.

Morimoto et al. 2005
Morimoto, J.; Nakanishi, J.; Endo, G.; Cheng, G.; Atkeson, C. G.; Zeglin, G.: Poincaré-Map-Based Reinforcement Learning For Biped Walking. In: 2005 IEEE International Conference on Robotics and Automation. Barcelona, Spain, 18-22 April 2005, S. 2381–2386, 2005.

Morimoto / Doya 1998a
Morimoto, J.; Doya, K.: Reinforcement learning of dynamic motor sequence: learning to stand up. In: Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications. Victoria, BC, Canada, 13-17 Oct. 1998, S. 1721–1726, 1998.

Morimoto / Doya 2001b
Morimoto, J.; Doya, K.: Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. In: *Robotics and Autonomous Systems* 36 (1), S. 37–51, 2001.

Mosbach 2002
Mosbach, C.: Das Reibungs und Reibschwingverfahren nasslaufender Lamellenkupplungen, München 2002.

Müller 2002
Müller, T.: Experimentelle Ermittlung der Einflußparameter auf das Kupplungsrupfen am Kfz-Antriebsstrang. Karlsruhe: Universität Karlsruhe, Institut für Maschinenkonstruktionslehre und Kraftfahrzeugbau (IPEK-thesis, 1027) 2002.

Naus et al. 2008
Naus, G.; Beenakkers, M.; Huisman, R.; van de Molengraft, R.; Steinbuch M.: Robust control to suppress clutch judder. In: Proceedings of AVEC'08. 9th International Symposium on Advanced Vehicle Control ; 2008, October 6 - 9th, Kobe International Conference Center, Kobe, Japan. International Symposium on Advanced Vehicle Control; AVEC. Kobe, 2008.

Naus et al. 2010
Naus, G. J.; Beenakkers, M. A.; Huisman, R. G.; van de Molengraft, M. J.G.;

Steinbuch, M.: Robust control of a clutch system to prevent judder-induced driveline oscillations. In: *Vehicle System Dynamics* 48 (11), S. 1379–1394, 2010.

Newcombe / Spurr 1972
Newcombe, T. P.; Spurr, R. T.: Clutch Judder. In: The Institution of Mechanical Engineers (Hg.): 14th International Automobile Technical Congress of Fisita. London, 1972.

Perkins / Hayes 1996
Perkins, S.; Hayes, G.: Robot shaping-principles, methods and architectures. In: University of Edinburgh (Hg.): DAI Research Paper 1996.

Peters et al. 2003a
Peters, J.; Vijayakumar, S.; Schaal, S.: Reinforcement learning for humanoid robotics. In: Proceedings of the third IEEE-RAS international conference on humanoid robots. Karlsruhe, Germany, S. 1–20, 2003.

Peters et al. 2003b
Peters, J.; Vijayakumar, S.; Schaal, S.: Scaling Reinforcement Learning Paradigms for Motor Control 2003. Online verfügbar unter http://www.kyb.mpg.de/fileadmin/user_upload/files/publications/petersVijayakumarSchaal_JSNC2003_5058%5B0%5D.pdf.

Peters 2008
Peters, J.: Machine learning for robotics. Learning methods for robot motor skills. Saarbrücken, Germany: VDM Verlag 2008.

Peters / Schaal 2006
Peters, J.; Schaal, S.: Policy Gradient Methods for Robotics. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. Beijing, China, S. 2219–2225, 2006.

Pfeiffer 1992
Pfeiffer, F.: Das Phänomen der selbsterregten Schwingungen. In:, Bd. 957. Düsseldorf: VDI Verlag (VDI Berichte), S. 1–24, 1992.

Pinte et al. 2010
Pinte, G.; Depraetere, B.; Symens, W.; Swevers, J.; Sas, P.: Iterative Learning Control of the filling phase of wet clutches. In: Proceedings of ISMA2010 - International Conference on Noise and Vibration Engineering including USD2010 - International Conference on Uncertainty in structural dynamics. Leuven. KU Leuven Department of Mechanical Engineering. Leuven, S. 391–408, 2010.

Rabeih / Crolla 1996
Rabeih, E.; Crolla, D. A.: Intelligent control of clutch judder and shunt phenomena in

vehicle drivelines. In: *International Journal of Vehicle Design* 17 (3), S. 318–332, 1996.

Raia 2014
Raia, J.: Manual Versus Automatic Transmissions 2014. Online verfügbar unter http://www.fix.com/blog/manual-vs-automatic-transmissions/, zuletzt aktualisiert am 19.02.2015, zuletzt geprüft am 20.02.2015.

Rokach / Maimon 2008
Rokach, L.; Maimon, O.: Data mining with decision trees. Theory and applications. Singapore: World Scientific (Series in machine perception and artificial intelligence, v. 69) 2008.

Ross 1983
Ross, S. M.: Introduction to stochastic dynamic programming. New York: Academic Press (Probability and mathematical statistics) 1983.

Röttger 2009
Röttger, M. C.: Reinforcement-Learning für kontinuierliche Zustands- und Aktionsräume unter Berücksichtigung der wissenschaftlichen Informationsverarbeitung. DissertationsschriftUniversity of Freiburg, Freiburg 2009.

Rummery / Niranjan 1994
Rummery, G. A.; Niranjan, M.: On-line q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166. Hg. v. Cambridge University Engineering Department 1994.

Schwenger 2005
Schwenger, A.: Aktive Dämpfung von Triebstrangschwingungen. DissertationUniversität Hannover, Hannover 2005.

Seitenbrecher 1997
Seitenbrecher, A.: Berechnung optimaler Verläufe des Kupplungsmomentes bei Ein- und Auskuppelvorgängen. DiplomarbeitTU Chemnitz, Chemnitz 1997.

SEW-EURODRIVE
SEW-EURODRIVE: Getriebemotoren, Frequenzumrichter und dezentrale Antriebstechnik. Online verfügbar unter http://www.sew-eurodrive.de/, zuletzt geprüft am 04.03.2015.

Singh et al. 2000
Singh, S.; Jaakkola, T.; Littman, M.; Szepesvári, C.: Convergence Results for Single-Step On-Policy Reinforcement-Learning Algorithms. In: *Machine Learning* (39), S. 287–308, 2000.

Sommer Obando et al. 2010
Sommer Obando, H.; Albers, A.; Frietsch, M.: A new approach for solving positioning tasks of robotic systems based on Reinforcement Learning. In: Annals of DAAAM 2010 - DAAAM International World Symposium, 2010.

Sommer Obando / Albers 2012
Sommer Obando, H.; Albers, A.: Hybrid Reinforcement Learning-based approach for agent motion control. In: 2012 IEEE International Conference on Industrial Technology (ICIT), 2012.

Sony 2014
Sony: Sony Aibo website for all models of Sony Aibo ERS-7, ERS-210, ERS-220, ERS-11x, ERS31x 2014. Online verfügbar unter http://www.sony-aibo.co.uk/, zuletzt geprüft am 22.10.2014.

Spurr 1961
Spurr, R. T.: A Theory of Brake Squeal. In: Proceedings of the Institution of Mechanical Engineers: Automobile Division, Bd. 15, S. 33–52 1961.

Sutton / Barto 1998
Sutton, R. S.; Barto, A. G.: Reinforcement learning: An introduction. Cambridge, Massachusetts: The MIT Press (9) 1998.

Szepesvári 2010
Szepesvári, C.: Algorithms for Reinforcement Learning. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 4 (1), S. 1–103, 2010.

Taylor 2009
Taylor, M. E.: Transfer in reinforcement learning domains. Berlin: Springer (Studies in computational intelligence, v. 216) 2009.

Tesauro 1992
Tesauro, G.: Practical issues in temporal difference learning. In: *Machine Learning* 8 (3-4), S. 257–277, 1992.

Tesauro 1995
Tesauro, G.: Temporal difference learning and TD-Gammon. In: *Commun. ACM* 38 (3), S. 58–68, 1995.

Tokic 2010
Tokic, M.: Adaptive \varepsilon-greedy exploration in reinforcement learning based on value differences. In: KI 2010: Advances in Artificial Intelligence: Springer Berlin Heidelberg, S. 203–210 2010.

van Hasselt 2010
van Hasselt, H.: Double Q-learning. In: Advances in Neural Information Processing Systems, S. 2613–2621, 2010.

van Hasselt / Wiering 2007
van Hasselt, H.; Wiering, M. A.: Reinforcement Learning in Continuous Action Spaces. In: IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning. Honolulu, HI, USA, S. 272–279, 2007.

Verein Deutscher Ingenieure (2009): VDI 2057 Blatt 2, Einwirkung mechanischer Schwingungen auf den Menschen Hand-Arm-Schwingungen.

Watkins, C. J. C. H. 1989
Watkins, C. J. C. H.: Learning from Delayed Rewards. PhD thesisCambridge University, Cambridge, England 1989.

White 1969
White, D. J.: Dynamic Programming. Edinburgh, London, San Francisco: Oliver & Boyd; Holden-Day (Mathematical economics texts, 1) 1969.

Whitehead / Ballard 1990
Whitehead, S. D.; Ballard, D. H.: Active Perception and Reinforcement Learning. In: *Neural Computation* 2 (4), S. 409–419, 1990.

Winkelmann / Harmuth 1985
Winkelmann, S.; Harmuth, H.: Schaltbare Reibkupplungen 7 Besonderheiten der Kraftfahrzeugkupplungen. Heidelberg: Springer-Verlag Berlin Heidelberg New York Tokyo 1985.

Witten 1976
Witten, I. H.: The apparent conflict between estimation and control—a survey of the two-armed bandit problem. In: *Journal of the Franklin Institute* 301 (1-2), S. 161–189, 1976.

Wooldridge / Jennings 1995
Wooldridge, M.; Jennings, N. R.: Intelligent agents: theory and practice. In: *Knowl. Eng. Rev.* 10 (02), S. 115, 1995.

www.researchinchina.com 2012
www.researchinchina.com: China Automotive Transmission Industry Report, 2012-2015 Report-Research In ChinaRIC 2012. Online verfügbar unter http://www.researchinchina.com/Report/ShowReport.aspx?id=6593&date=2012, zuletzt geprüft am 04.03.2015.

Yoshimoto et al. 1999
Yoshimoto, J.; Ishii, S.; Sato, M.: Application of reinforcement learning to balancing of

Acrobot. In: IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics. Tokyo, Japan, 12-15 Oct. 1999, S. 516–521, 1999.

Zell 1997
Zell, A.: Simulation neuronaler Netze. 2., unveränd. Nachdr. München [u.a.]: Oldenbourg 1997.

Zhang / Huang 2006
Zhang, Y.; Huang, Q.: A Learning-based Adaptive Routing Tree for Wireless Sensor Networks. In: *JCM* 1 (2), 2006.

Zink et al. 2002
Zink, M.; Shead, R.; Welter, R.: Kupplungsausrücksysteme. In: *7. LuK Kolloquium* 7, 2002.

Zink et al. 2010
Zink, M.; Hausner, M.; Welter, R.; Shead R.: In Depth Details on the Clutch and Release Systems of Audi 2010. Online verfügbar unter http://www.audizine.com/forum/showthread.php/350288-In-Depth-Details-on-the-Clutch-and-Release-Systems-of-Audi, zuletzt geprüft am 29.04.2015.

Student Research Work

Hirsenkorn et al. 2012
Hirsenkorn, N.; Sommer Obando, H.; Burger, W.; Albers, A.: Development of a Reinforcement Learning Framework for Clutch Judder Suppression; Bachelor's thesis at IPEK, 2012.

Xu, T. et al. 2013
Xu, T.; Sommer Obando, H.; Burger, W.; Albers, A.: Development of a learning algorithm for Clutch Judder Minimization; Master's thesis at IPEK, 2013.

Kastl, T. et al. 2013
Hirsenkorn, N.; Sommer Obando, H.; Burger, W.; Albers, A.: Real-Time algorithm for to counter Clutch Judder; Bachelor's thesis at IPEK, 2013.

Hauser, F. et al. 2014
Hirsenkorn, N.; Sommer Obando, H.; Burger, W.; Albers, A.: Machine Learning based Reduction of Clutch Judder; Bachelor's thesis at IPEK, 2014.

10 Appendix

Appendix A

Values for simulation of the reduced drive train from HORNUNG 2012

Parameter	**Symbol**	**Value**	**Unit**
Idle speed (input)	n_0	900	$1/\mathrm{min}$
Combustion engine inertia	J_{CE}	$6 \cdot 10^{-3}$	kgm^2
Added inertia of clutch and gearbox	J_{CG}	$8 \cdot 10^{-5}$	kgm^2
Vehicle inertia	J_V	$9.91 \cdot 10^{-3}$	kgm^2
Side shaft stiffness	c_s	0.7219	Nm/rad
Side shaft damping	d_s	0	Nms/rad
Combustion engine damping	d_{ce}	$3.565 \cdot 10^{-3}$	Nms/rad
Mean friction radius	R_M	0.1	m
Static friction coefficient (coefficient of adhesion)	μ_{st}	0.43	-
Friction gradient	μ'	-0.013	s/m

Appendix B

Values for the simulation model of the Mini-HiL test bench from GWOSCH ET AL. 2013.

Parameter	Symbol	Value	Unit
Sample rate (real time environment)	T_a	0.001	s
Idle speed (input)	n_0	900	1/min
Combustion engine inertia	J_{CE}	$6 \cdot 10^{-3}$	kgm^2
Added inertia of clutch and gearbox	J_{CG}	$3.85 \cdot 10^{-4}$	kgm^2
Vehicle inertia	J_V	$2.4 \cdot 10^{-2}$	kgm^2
Side shaft stiffness	c_s	2.1376	Nm/rad
Side shaft damping	d_s	$7.64 \cdot 10^{-5}$	Nms/rad
Combustion engine damping	d_{ce}	$3.565 \cdot 10^{-3}$	Nms/rad
Mean friction radius	R_M	0.05	m
Static friction coefficient (coefficient of adhesion)	μ_{st}	0.71	-
Friction gradient	μ'	$-3.7 \cdot 10^{-2}$	s/m

Curriculum Vitae

Personal data	
Name:	Hermann Sommer Obando
Date of birth:	April 22, 1984
Place of birth:	Mexico City, Mexico
Nationality:	German / Mexican
Family status:	single
Educational background	
1992 – 2003	Attended primary, secondary and high school at „Colegio Alemán Alexander von Humboldt” in Mexico City
2003 – 2010	Mechanical engineering studies until acquisition of the degree of ”Diplom-Ingenieur (Dipl.-Ing.)” at Karlsruhe Institute of Technology (KIT)
Professional life	
2006 – 2008	Student research assistant at wbk – Institute of Production Science, Karlsruhe Institute of Technology (KIT)
2008 – 2009	Intern in the frame and body construction area at Daimler AG Mercedes-Benz Plant Rastatt
2010 – 2011	Graduate research assistant at IPEK – Institute of Product Engineering; Karlsruhe Institute of Technology
2011 – 2015	CONACYT / DAAD-stipendiary and graduate research assistant at IPEK - Institute of Product Engineering; Karlsruhe Institute of Technology (KIT)
Since 09/2015	Software Engineer in Control Systems & Model Based Design at ITK Engineering AG