



Machine Learning Based Calibration of Dual Clutch Transmissions for Optimizing the Launch Behavior of Passenger Vehicles

Am Fachbereich Maschinenbau

an der Technischen Universität Darmstadt

zur

Erlangung des Grades eines Doktor-Ingenieurs (Dr.-Ing.)

genehmigte

DISSERTATION

vorgelegt von

MARIUS SCHMIEDT, M.SC.

aus Schwäbisch Hall

Berichterstatter:	Prof. Dr.-Ing. Stephan Rinderknecht
Mitberichterstatter:	Prof. Dr.-Ing. Ulrich Konigorski
Tag der Einreichung:	05.11.2023
Tag der mündlichen Prüfung:	28.02.2024

Darmstadt 2024

D17

Schmiedt, Marius : Machine Learning Based Calibration of Dual Clutch Transmissions for Optimizing the Launch Behavior of Passenger Vehicles, Technische Universität Darmstadt,
Jahr der Veröffentlichung der Dissertation auf TUpriints: 2024
URN: urn:nbn:de:tuda-tuprints-273659

Tag der mündlichen Prüfung: 28.02.2024

Veröffentlicht unter CC BY-SA 4.0 International <https://creativecommons.org/licenses/>

Abstract

The drivability of a vehicle is strongly affected by its transmission. Especially dual clutch transmissions (DCT) offer the chance of a comfortable drivability through its ability of blending the torque during gear shifts from one clutch to the other (jerkless shifting). Another advantage is the higher efficiency compared to torque converters. These advantages come with the drawback of a high control effort for the clutch engagement of the two clutches. The control effort is handled with software functions (developed using model-based programming languages) deployed on the transmission control unit (TCU) with adjustable control parameters (calibration parameters). With these control parameters e.g., the driving behavior is adjustable for different vehicle, engine combinations. Calibration engineers set these parameters at different ambient conditions to comply with customer requirements in an iterative time-consuming process on costly test trips. Therefore, costs are increasing with increasing control opportunities. An approach for decreasing these costs is to automate the optimization of the calibration parameters. Several approaches have already been introduced but some suffer from lack of stability or time efficiency. Hence, to optimize these parameters the target state optimization (TSO) algorithm is illustrated where a target state is approached with a hybrid solution of reinforcement learning (RL) and supervised learning (SL) to overcome existing drawbacks. Since particularly at low speeds the transmission behavior must meet the intention of the driver (drivers tend to be more perceptive at low speeds) the control of the launch behavior is crucial and is therefore investigated in this study. The algorithm is applied in different environments e.g., in a software in the loop (SiL) environment as well as in different test vehicles to optimize the launch behavior and to verify if a deployment in existing development processes is possible. Further the application in different environments such as in different test vehicles proves the ability of the TSO algorithm to generalize.

Kurzfassung

Das Fahrverhalten eines Fahrzeugs mit Verbrennungsmotor wird stark von dessen Getriebe beeinflusst. Vor allem Doppelkupplungsgetriebe (DCT) bieten durch die Fähigkeit bei Schaltungen das Drehmoment von der aktiven Kupplung (aktueller Gang) auf die passive Kupplung (nächst höherer/kleiner Gang) zu überblenden die Möglichkeit eines komfortablen Fahrverhaltens. Mit dieser Eigenschaft lassen sich Gangwechsel ruckfrei durchführen. Ein weiterer Vorteil ist die höhere Effizienz im Vergleich zu automatisierten Getrieben mit Drehmomentwandlern. Diese Vorteile gehen mit dem Nachteil eines hohen Kalibrationsaufwands einher. Die aufwendige Regelung wird mit Softwarefunktionen (entwickelt mit modellbasierten Programmiersprachen), welche auf dem Getriebesteuergerät (TCU) ausgeführt werden realisiert. Diese Softwarefunktionen enthalten einstellbare Kalibrationsparameter. Mit diesen Kalibrationsparametern kann das Fahrverhalten für verschiedene Fahrzeug-Motoren-Kombinationen eingestellt werden. Kalibrierungsingenieure stellen diese Parameter in verschiedenen Umgebungsbedingungen unter Berücksichtigung von Kundenanforderungen ein. Dieser iterative und zeitaufwendige Prozess findet üblicherweise auf kostspieligen Fahrzeugerprobungen statt. Ein Ansatz, die entstehenden Kosten zu senken, besteht darin, die Optimierung der Kalibrierungsparameter zu automatisieren. Dazu gibt es bereits mehrere Ansätze, wobei einige allerdings sehr zeitintensiv oder Ergebnisse teilweise nicht reproduzierbar sind. Aus diesem Grund wird in dieser Arbeit der Target State Optimization (TSO) Algorithmus veranschaulicht, welcher versucht, ein vordefiniertes Ziel anzunähern. Der Algorithmus ist hierbei eine Hybridlösung aus Reinforcement Learning (RL) und Supervised Learning (SL) und versucht damit bestehende Nachteile zu überwinden. Da besonders bei niedrigen Geschwindigkeiten das Fahrverhalten dem Fahrerwunsch entsprechen sollte (Fahrer neigen dazu, bei niedrigen Geschwindigkeiten feinfühlicher gegenüber Beschleunigungsänderungen zu sein), wird in dieser Arbeit die Optimierung des Anfahrvorgangs angestrebt. Dabei wird der Algorithmus in verschiedenen Testumgebungen (z.B. in einer Software-in-the-Loop-Umgebung), aber auch in verschiedenen Testfahrzeugen eingesetzt. Dabei soll ermittelt werden, ob ein Einsatz in bestehende Entwicklungsprozesse möglich ist. Durch Anwendung des Algorithmus in verschiedenen Testumgebungen und Testfahrzeugen wird die Anpassungsfähigkeit des Algorithmus auf unterschiedliche Optimierungsumgebungen gezeigt.

Table of Contents

Abstract.....	I
Kurzfassung	II
Table of Contents.....	III
List of Figures	V
List of Tables.....	VIII
Nomenclature	IX
1 Introduction	1
2 Fundamentals.....	3
2.1 Driving Resistance – Power Demand	3
2.2 Propulsion Unit – Power Source	7
2.2.1 Combustion Engines – Characteristic.....	7
2.2.2 Combustion Engines – The purpose of the transmission	11
2.2.3 Electric Motors	15
2.3 Hybrid Vehicles	18
2.3.1 Hybrid functions.....	19
2.3.2 Hybrid Topologies.....	20
2.4 Transmission	22
2.4.1 Clutch Engagement.....	22
2.4.2 Dual Clutch Transmission (DCT)	24
2.4.3 Hybrid Dual Clutch Transmission (HDT)	26
2.5 Transmission Control Unit (TCU)	27
2.5.1 TCU Architecture	27
2.5.2 Transmission functions	27
2.6 Calibration.....	30
3 Automizing the calibration – Known approaches (State of the Art)	35
3.1 Reinforcement Learning	36
3.1.1 Background and Motivation.....	37
3.1.2 Deployment in the scope of the study.....	39
3.2 Genetic Algorithms	40
3.2.1 Background and Motivation.....	40
3.2.2 Deployment in the scope of the study.....	41
3.3 Outcome.....	42
4 Optimization objectives.....	43

4.1	Customer Objectives	44
4.1.1	<i>Acceleration Peak and Acceleration Build-Up Objective</i>	44
4.1.2	<i>Reaction Time Objective</i>	46
4.2	Discomfort Objectives	47
4.2.1	<i>Engine Speed Objective</i>	47
4.2.2	<i>Clutch Torque/Jerk Objective</i>	48
4.3	The Reward	51
5	Test environments	53
5.1	Test Vehicle	53
5.2	Software in the Loop (SiL) Environment	54
5.3	Simplified model	56
5.3.1	<i>Simplified Engine Model</i>	57
5.3.2	<i>Simplified Vehicle Model</i>	59
5.3.3	<i>Simplified Transmission Model</i>	60
5.3.4	<i>Main Loop</i>	62
5.3.5	<i>Influence of the Calibration Parameters</i>	63
6	Target State Optimization	66
6.1	Generation of the Action	67
6.2	Brief Introduction into Neural Networks	71
6.3	Model Optimization and Hyperparameter-Tuning	73
6.4	Relevance of the Activation Function	76
6.5	Batch Size	78
6.6	Robustness	79
7	Results of the calibration of the launch behavior	80
7.1	Simplified Model	80
7.2	Software in the Loop (SiL) environment	86
7.3	Test vehicles	89
8	Discussion	97
9	Conclusion	99
	Bibliography	101

List of Figures

Figure 2.1. Illustration of the Driving Resistances based on [5]. 5

Figure 2.2. Driving Resistance of the fictive Vehicle..... 7

Figure 2.3. Effective mean Pressure: (a) Gasoline engine, (b) Diesel Engine according to [11] 9

Figure 2.4. Engine torque behavior: (a) Gasoline Engine, (b) Diesel Engine based on [11]. 10

Figure 2.5. Fictive torque map. 10

Figure 2.6. Engine Power over engine speed. 11

Figure 2.7. Optimal Traction effort and the engine behavior based on [12]. 12

Figure 2.8. Enhanced operating points by deploying a transmission..... 13

Figure 2.9. Provided Engine Force and Driving Resistance 14

Figure 2.10. Gear Selection based on the Driving Resistance 15

Figure 2.11. Torque map of an electric motor based on [14]. 16

Figure 2.12. Torque map of the Tesla Model S based on [17]. 17

Figure 2.13. Optimal tractive effort of the Tesla model S derived from [16] and [17] without considering wheel slip thresholds. 17

Figure 2.14. Types of hybrid Vehicles based on [12]. 18

Figure 2.15. Efficiency map of an ICE based on [23] illustrating the specific fuel consumption in g/kWh. 20

Figure 2.16. Hybrid topologies according to [22] and [24]. 21

Figure 2.17. Clutch configuration of manual transmissions (based on [4])..... 23

Figure 2.18. Engine torque behavior for manual transmissions (based on [5]) 23

Figure 2.19. Clutch configuration of DCT (based on [5]) 24

Figure 2.20. Clutch torque behavior during the shift with a DCT (based on [26])..... 25

Figure 2.21. Engine speed behavior during the gear shift (based on [26])..... 25

Figure 2.22. HDT / P 2.5 hybrid topology according to [25]. 26

Figure 2.23. TCU function structure based on [36], [37] 28

Figure 2.24. signal structure of a TCU based on [5] 29

Figure 2.25. Simplified Calibration process 30

Figure 2.26. Engine speed (red) and input-shaft speed (green) behavior during vehicle launch..... 32

Figure 2.27. Engine torque (red) and clutch torque (green) behavior during vehicle launch. 33

Figure 2.28. Relation of Clutch Torque and longitudinal acceleration: (a) Clutch torque, (b) Longitudinal acceleration 34

Figure 3.1. Q-learning structure according to [52]. 37

Figure 3.2. RL example task: cartpole 38

Figure 4.1. Acceleration profile of a launch [3]. 45

Figure 4.2. (a) Engine speed (red) does not drop, (b) Engine speed drops; Input-shaft speed is green.

.....	48
Figure 4.3. Influence of the filtering on the signal (a) unfiltered, (b) filtered.	50
Figure 4.4. Clutch torque with one local minimum.....	51
Figure 5.1. Clutch torque behavior of (a) a real vehicle and (b) the SiL environment.	55
Figure 5.2. Engine speed (red) and input-shaft speed behavior of (a) a real vehicle and (b) the SiL environment.....	56
Figure 5.3. Simplified vehicle model.....	57
Figure 5.4. Engine torque of the simplified engine model.....	58
Figure 5.5. Desired engine speed of the simplified model.....	61
Figure 5.6. Influence of the "SpeedEngineGradient "; (a/b) : desired engine speed (blue), engine speed (red), input-shaft speed (green).....	64
Figure 5.7. Influence of the "SpeedEngineGradient "; (a/b) : engine torque (blue), clutch torque (green).....	64
Figure 5.8. Influence of the P-gain; (a/b) : desired engine speed (blue), engine speed (red), input-shaft speed (green).	65
Figure 5.9. Influence of the P-gain (a/b) : engine torque (blue), clutch torque (green).	65
Figure 6.1. Flowchart of the TSO algorithm: main loop.....	67
Figure 6.2. Progress of ε with (a) $\varepsilon_{dec} = 0.98$ and $\varepsilon_{min} = 0.1$, (b) $\varepsilon_{dec} = 0.995$ and $\varepsilon_{min} = 0.01$	68
Figure 6.3. Flowchart of the TSO algorithm: generation of the action.....	70
Figure 6.4. Neural network architecture according to [103].....	71
Figure 6.5. Cost function with (a) small learning rate, (b) large learning rate.	73
Figure 6.6. Flowchart of the model optimization.	75
Figure 6.7. (a) linear, (b) ReLU, and (c) logistic sigmoid activation function (red) and its derivative (blue).....	77
Figure 6.8. Comparison of the influence of the activation functions of the TSO algorithm on the results. (a) sigmoidal activation function, (b) ReLU activation function.	78
Figure 7.1. Results of the fourth of five test runs of the TSO algorithm applied to the simplified model with a reaction time objective of 0.4 s. The red dots indicate successful launches.	82
Figure 7.2. Action distribution generated by the TSO algorithm in the fourth test run on the simplified model for a reaction time objective of 0.4 s; red: randomly generated actions, green: predicted actions of the neural network, blue: epsilon threshold.	84
Figure 7.3. Action distribution of the successful iterations; red: randomly generated actions, green: predicted actions of the neural network, blue: epsilon threshold.....	85
Figure 7.4. P-Gain of the different parameter sets of Table 7.3. (a) Parameter set 1 recommended to be chosen ($x_S = 790$), (b) Parameter set 4 ($x_S = 1$).....	86
Figure 7.5. Results of an NSGA-II test run in the SiL environment. The red dots indicate successful launches.....	88

Figure 7.6. Results of the TSO algorithm within the SiL environment. The red dots indicate successful launches.....	89
Figure 7.7. NSGA-II results tested in a vehicle with an HDT and a three-cylinder Otto engine : (a) Generations: 5, Populations: 4; (b) Generations: 5, Populations: 12.	90
Figure 7.8. Results of the TSO algorithm tested in a test vehicle with an HDT and a three-cylinder Otto engine. The red dots indicate successful launches.	91
Figure 7.9. (a) Successful launch (TSO: iteration 11 of Figure 7.8), (b) Failed launch (NSGA-2: iteration 0 of Figure 7.7a); Engine Speed (red); Input-shaft speed (green).....	92
Figure 7.10. Clutch torques of the different launches accordingly to Figure 7.9: (a), desired clutch torque behavior (b) clutch torque with local minima.....	92
Figure 7.11. NSGA-II results tested in a test vehicle with a conventional DCT and a three-cylinder Otto engine: (a) generations: 5, population: 4; (b) generations: 5, population: 12. The red dot indicate a successful launch.....	93
Figure 7.12. Results of the TSO algorithm tested in a test vehicle with a conventional DCT and a three-cylinder Otto engine. The red dots indicate successful launches.....	94
Figure 7.13. Results of the TSO algorithm tested in a test vehicle with an HDT and a four-cylinder diesel engine (Acceleration target: 3.25 m/s ² , reaction time target: 0.5 s).....	95
Figure 7.14. Results of the TSO algorithm tested in a test vehicle with an HDT and a four-cylinder diesel engine (Acceleration target: 3.5 m/s ² , reaction time target: 0.3 s).....	96

List of Tables

Table 2.1. Estimated coefficients	6
Table 2.2. Fictive Gear ratios (estimation based on [5]).....	13
Table 3.1. Q-Table example	38
Table 4.1. Target state for the SiL environment.....	52
Table 5.1. Example of a calibration map–Parameter 3: P-gain.	54
Table 5.2. P-Gain of the simplified model.....	63
Table 6.1. Hyperparameter propagation test run 1.....	76
Table 6.2. Hyperparameter propagation test run 2.....	76
Table 6.3. Comparison of the activation functions.....	78
Table 6.4. Successful iterations of the robustness test with the TSO algorithm.....	79
Table 7.1. TSO algorithm applied on the simplified model.....	81
Table 7.2. Parameter range of the simplified model for the optimization.	82
Table 7.3. Parameter sets of the fourth of five test runs leading to successful results in the simplified model for a reaction time objective of 0.4s.	83
Table 7.4. Results of the different successfully applied parameter sets (generated by the TSO algorithm in the fourth test run on the simplified model) for a reaction time objective of 0.4 s.....	83
Table 7.5. Comparison of TSO with existing algorithms.	87
Table 7.6. Optimization targets.	90
Table 7.7. Comparison of the TSO algorithm with different optimization targets within the vehicle with a four-cylinder diesel-engine and an HDT.....	95

Nomenclature

Abbreviations

Acronym	Description
A2C	Advantage Actor-Critic
AMT	Automatic Manual Transmission
AWD	All-Wheel Drive
CPU	Central Processing Unit
CVT	Continuously Variable Transmission
DCT	Dual Clutch Transmission
DDPG	Deep Deterministic Policy Gradient
DOE	Design of Experiments
DQL	Deep Q-Learning
DQN	Deep Q-Network
ECU	Engine Control Unit
EEPROM	Electrically Erasable Programmable Read-Only Memory
EPROM	Erasable Programmable Read-Only Memory
EV	Electric Vehicle
FWD	Front-Wheel Drive
GA	Genetic Algorithm
HDT	Hybrid Dual Clutch Transmission
ICE	Internal Combustion Engine
IMTR	Iterative Model and Trajectory Refinement
ML	Machine Learning
MT	Manual Transmission
NSGA-II	Nondominated sorting genetic algorithm 2
PHEV	Plug-In Hybrid
PPO	Proximal Policy Optimization
RAM	Random Access Memory

ReLU	Rectified Linear Unit
RL	Reinforcement Learning
ROM	Read Only Memory
SAC	Soft Actor-Critic
SiL	Software in the Loop
SL	Supervised Learning
SPEA-II	Strength Pareto Evolutionary Algorithm 2
TCU	Transmission Control Unit
TSO	Target State Optimization

Latin Symbols

Symbol	Description
A	Reference area for the aerodynamic drag calculation
a	Acceleration; Single unit of a neural network; Action
b	Batch size of a neural network
b_e	Specific fuel consumption
c	Number of cylinders
c_D	Drag coefficient
d	Number of input variables of a neural network
d_c	Cylinder bore diameter
E	Value of the error function for training the neural network
f	Rolling resistance coefficient; Fail count
F	Force
fac_T	Factor to multiply the torque request to determine the engine torque of the simplified model
g	Gravitational acceleration; Non-linear activation function of a neural network

G	Number of generations of a genetic algorithm
H_U	Calorific value of the fuel
i	Revolutions per stroke; Total gear ratio; Iteration
it	Initial iterations for the optimization with the TSO algorithm
J	Inertia
L	Cylinder stroke length
m	Mass; Maximum numbers of models created during the optimization with the TSO algorithm
n	Speed; Number of data points to train a neural network; Current number of neural network models created during the optimization with the TSO algorithm
obj	Objective
p_{me}	Mean effective pressure
P	Power; Population size of a genetic algorithm
Q	Value function of the deep Q-learning algorithm
r	Tire radius
t	Time; Size of the dataset produced by the TSO algorithm
T	Torque; Threshold for optimizing the neural network model during the optimization with the TSO algorithm
v	Velocity; Value of the customer objective
V	Volume
w	Weight of a neural network
x	Input of a neural network
\hat{y}	Output value of a neural network
y	True value for comparison with the predicted value \hat{y} of the neural network to determine the Error E
z	Number of teeth on a gear; output of a unit of a neural network a

Greek Symbols

Symbol	Description
α	Slope angle
ε	Epsilon of the ε -Greedy approach
λ_L	Volumetric efficiency
η	Efficiency
ρ_a	Density of the air
ω	Angular velocity

Subscripts

Index	Description
<i>o</i>	Initial value
<i>a</i>	Axle
<i>acc</i>	Translatory acceleration for determining the driving resistance
<i>build</i>	Acceleration build-up
<i>clu</i>	Clutch
<i>customer</i>	Customer
<i>d</i>	Differential transmission
<i>dec</i>	Decrease
<i>drag</i>	Aerodynamic drag
<i>drop</i>	Drop of the engine speed
<i>dyn</i>	Dynamic
<i>e</i>	Engine; Effective
<i>fuel</i>	Fuel
<i>g</i>	Gear box; Gradient (threshold)
<i>in</i>	Input

<i>max</i>	Maximal
<i>meas</i>	Measured value
<i>min</i>	Minimal
<i>mix</i>	Air fuel mixture
<i>n</i>	Engine speed
<i>out</i>	Output
<i>pwt</i>	Powertrain
<i>r</i>	Resulting driving resistance
<i>rea</i>	Reaction
<i>red</i>	Reduced
<i>ref</i>	Reference value
<i>req</i>	Request
<i>roll</i>	Rolling resistance
<i>slope</i>	Gradient resistance
<i>stat</i>	Static
<i>T</i>	Torque; Tuning (threshold)
<i>v</i>	Vehicle
<i>w</i>	Wheel

1 Introduction

Since the legislative requirements regarding emissions are constantly getting stricter, manufacturers of the automotive industry are urged to improve their products to meet these requirements and additionally the customer demands. The improvements must be considered by the vehicle manufacturer itself as well as by their suppliers. Therefore, beside the improvement of internal combustion engines also parts of the entire powertrain must be improved. Since changing mechanical parts is very expensive the software of the engine control unit (ECU) and the transmission control unit (TCU) are preferably improved to change the behavior of the corresponding powertrain components.

With an increasing number of legislative requirements also the amount of software functions e.g., due to the hybridization of the powertrain are increasing. With the number of functions within the software, also the number of parameters to be optimized is increasing.

The transmission is also the component which affects the drivability of a vehicle the most. Thus, leads to the issue of an optimization of the software of the TCU regarding emissions, drivability without worsening the lifetime of a gear box. The problem is that adjustments of the software of the ECU (e.g., due to legislative requirements or engine optimization) requires necessarily the adjustment of the parameters of the TCU. The adjustments on the TCU are required since usually the engine torque behavior is controlled by the ECU which is an input of the TCU. To fit e.g., the clutch engagement (during gear shifts or vehicle launches) to the new engine torque behavior, therefore the TCU parameters also need an adjustment.

Since the dual clutch transmission (DCT) delivers several advantages like decreasing emissions, a comfortable drivability through jerkless shifting and a higher efficiency, the study focuses on the investigation of DCTs and hybrid dual clutch transmissions (HDTs) of the company Magna. Nevertheless, beside these advantages there are also some disadvantages which must be considered, such as the higher control effort. The software of the TCU for example consists of about 15000 control parameters, of which 600 are only for the launching behavior of the vehicle. These parameters are necessary to influence the drivability at all possible driving conditions. This leads to high costs due to necessary test drives in different countries. The test drives are necessary to improve and validate the drivability at e.g., high altitudes, high and low temperatures and on different street surfaces.

Since there are many vehicle, engine, transmission combinations (the same engine and the same transmission can be deployed in different vehicles with different weights resulting in different driving resistances) the same parameters have to be adjusted multiple times in different vehicles. Therefore, an efficient parameter optimization method is required, which is decreasing development time of the different powertrain combinations to further save costs.

The aim is to automate the parameter optimization with a machine learning (ML) based algorithm, which is a hybrid of reinforcement learning (RL) and supervised learning (SL). The proposed algorithm is called target state optimization (TSO) and is introduced in the conference paper [1] and the article [2]. These publications are the foundation of this thesis. Further, the optimization is performed considering objective values chosen with respect to the test subject study of He et al.



[3].

To understand the motivation of the thesis some fundamentals are introduced in Chapter 2. Chapter 2 explains the driving resistance and how it is calculated. In Chapter 2.2 a brief introduction into internal combustion engines (ICEs) and electric motors is given to illustrate why transmissions are required in a vehicle. The fundamentals of hybrid powertrains and its benefits are described in Chapter 2.3. Chapter 2.4 is introduced to examine the comparison between manual and dual clutch transmissions. The heart of an automatic transmission is the TCU. An overview of the TCU architecture and its functionalities is given in Chapter 2.5. The functionalities of the TCU enable the software to be calibrated regarding a specific vehicle which is outlined in Chapter 2.6. The current state of the art of automating the calibration process with knowledge-free self-learning algorithms is illustrated in Chapter 3. The optimization objectives which are required to automatize the calibration is introduced in Chapter 4. The environments to test different self-learning approaches are outlined in Chapter 5. The new promising TSO approach to optimize the parameters of the TCU based on [1] and [2] is illustrated in Chapter 6. The results of the different self-learning approaches are given in Chapter 7. A brief discussion is introduced in Chapter 8. The thesis is concluded in Chapter 9.

2 Fundamentals

The Chapter “Fundamentals” is introduced to provide a brief summary about the driving resistance, the characteristic of combustion engines and electric motors and their joint use in hybrid vehicles. This introduction is intended to clarify the need of transmissions and illustrates the difficulties in the calibration specially for dual clutch transmissions.

2.1 Driving Resistance – Power Demand

To ensure a vehicle motion or driving at a certain velocity any vehicle has to provide torque to overcome the driving resistance required for the demanded driving maneuver. The driving resistance is calculated using the resistance equations from [4].

The rolling resistance F_{roll} is calculated with:

$$F_{roll} = f \cdot m \cdot g \cdot \cos(\alpha) \quad (2.1)$$

With f as the rolling resistance coefficient, m as the vehicle mass and g as the gravitational acceleration.

The gradient resistance F_{slope} is another driving resistance which is calculated with:

$$F_{slope} = m \cdot g \cdot \sin(\alpha) \quad (2.2)$$

With α as the slope angle. It is observable that the gradient resistance F_{slope} and rolling resistance F_{roll} are constant values and not dependent on the velocity.

In contrast the aerodynamic drag F_{drag} is calculated with:

$$F_{drag} = c_D \cdot A \cdot \rho_a \cdot \frac{v^2}{2} \quad (2.3)$$

Where c_D is the drag coefficient, A is the reference area, ρ_a is the density of the air and v is the velocity of a vehicle. It is observable that the aerodynamic drag F_{drag} does only have a minor influence on the driving resistance at low velocities. In contrast at greater speeds the aerodynamic drag F_{drag} has a major influence on the driving resistance since the aerodynamic drag is calculated with the squared velocity.

To accelerate a vehicle the force resulting from acceleration has to be overcome. The translatory acceleration F_{acc} is determined with:

$$F_{acc} = m_{red} \cdot \frac{dv}{dt} \approx m_{red} \cdot a \quad (2.4)$$

With a as the acceleration. To determine the force the reduced mass m_{red} is needed:

$$m_{red} = m + \frac{J_w + i^2 \cdot J_e}{r_{stat} \cdot r_{dyn}} \quad (2.5)$$

With J_w as the inertia of the wheel, J_e as the inertia of the engine, r_{stat} as the static rolling radius, r_{dyn} as the dynamic rolling radius and i as the ratio of the drivetrain. The total gear ratio i of the drivetrain is:

$$i = i_g \cdot i_d \quad (2.6)$$

With i_g as the gear ratio of the transmission and i_d as the gear ratio of the differential transmission.

The resulting driving resistance F_r is determined with:

$$F_r = F_{drag} + F_{roll} + F_{slope} + F_{acc} \quad (2.7)$$

To illustrate the forces which are influencing the driving behavior Figure 2.1 is introduced.

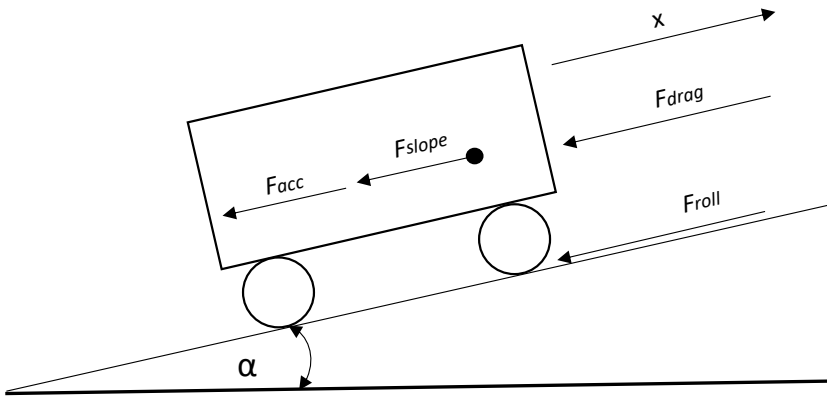


Figure 2.1. Illustration of the Driving Resistances based on [5].

With the driving resistance and the velocity, it is possible to determine the power at driven axle P_R :

$$P_r = F_r \cdot v \quad (2.8)$$

The torque at the driven axles T_a to achieve the desired motion can be determined with:

$$T_a = \frac{P_r}{\omega_r} \quad (2.9)$$

Therefore, the angular velocity at the wheel is needed:

$$\omega_r = \frac{v}{2 \cdot \pi \cdot r_{dyn}} \quad (2.10)$$

Hence, the torque summarizes to T_a :

$$T_a = \left(c_D \cdot A \cdot \rho \cdot \frac{v^2}{2} + f \cdot m \cdot g \cdot \cos(\alpha) + m \cdot g \cdot \sin(\alpha) + m \cdot a \right) \cdot 2 \cdot \pi \cdot r_{dyn} \quad (2.11)$$

And by neglecting the slope the torque is:

$$T_a = \left(c_D \cdot A \cdot \rho \cdot \frac{v^2}{2} + f \cdot m \cdot g + m \cdot a \right) \cdot 2 \cdot \pi \cdot r_{dyn} \quad (2.12)$$

To visualize the resulting torque as a function of the velocity the following estimations are considered:

Table 2.1. Estimated coefficients

Description	Abbreviation	Value
Drag coefficient [4]	c_D	0.3
Cross sectional area [4]	A	1.9 m^2
Density of the air	ρ	$1.225 \frac{\text{kg}}{\text{m}^3}$
Rolling resistance coefficient [4]	f	0.01
Static / Dynamic rolling radius (estimated)	$r_{dyn} = r_{stat}$	0.3 m
Vehicle mass (estimated)	m	1800 kg
Inertia of the wheel (estimated)	J_w	$1.2 \frac{\text{kg}}{\text{m} \cdot \text{s}^2}$
Inertia of the engine (estimated based on [6])	J_e	$0.3 \frac{\text{kg}}{\text{m} \cdot \text{s}^2}$
Gear ratio 1 st gear [5]	i_g	4
Gear ratio differential transmission (estimated)	i_d	4

These estimations are also used in the simplified powertrain model of Section 5.3 to illustrate the behavior of the TSO algorithm of Chapter 6.

Based on these equations the aerodynamic drag and the rolling resistance are shown as a function of the velocity:

— Aerodynamic Drag — Resulting Driving Resistance — Rolling Resistance

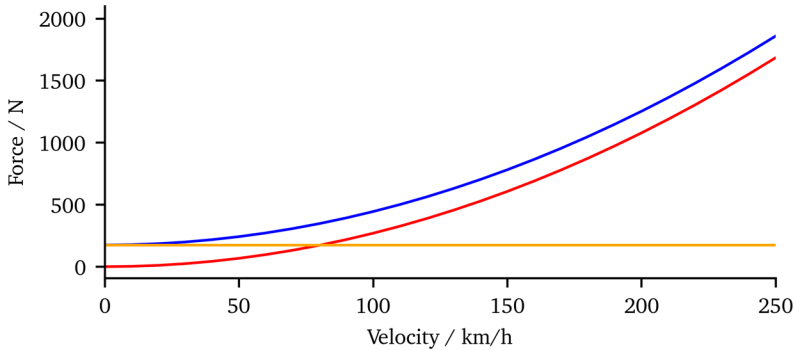


Figure 2.2. Driving Resistance of the fictive Vehicle

The aerodynamic drag is squared velocity dependent. Therefore, the driving resistance increases with the velocity. The rolling resistance has a constant level since the velocity does not influence it. Hence, it builds an offset for the resulting driving resistance.

The figure above illustrates the driving resistance of a vehicle driving with a constant velocity (no influence of acceleration).

2.2 Propulsion Unit – Power Source

The Chapter is introduced to present a brief overview over common propulsion units of a vehicle. Section 2.2.1 describes the behavior of an ICE and why a transmission is necessary in vehicles with ICEs. Electric motors are described in Section 2.2.3. Also, the benefits of having a transmission in electric drivetrains is outlined.

2.2.1 Combustion Engines – Characteristic

Although policies aim to reduce carbon emissions and hence urge manufacturers to accelerate innovations in the electric mobility sector [7] the vast majority of sold vehicles are still powered by ICEs [8]. Therefore, this Section illustrates the basic behavior of an ICE.

An ICE provides torque to accelerate a vehicle. For examining the engine torque and speed and its creation only the full load curve of the engine is observed. To calculate the torque of an ICE according to Heywood the following calculations have to be considered:

The cylinder swept volume V_s is determined by [9]:

$$V_e = \left(\frac{\pi}{4} \cdot d_c^2 \right) \cdot L \cdot c \quad (2.13)$$

With d_c as the cylinder bore diameter, L as the stroke length and c as the number of cylinders.

Further, the engine torque T is calculated by:

$$T = p_{me} \cdot V_e \cdot \frac{i}{2 \cdot \pi} \quad (2.14)$$

With p_{me} as the mean effective pressure and i as the revolutions per stroke (for four stroke engines $i = 0.5$).

The Power is:

$$P = T \cdot \omega = p_{me} \cdot V_e \cdot \frac{i \cdot \omega}{2 \cdot \pi} \quad (2.15)$$

With ω as the angular velocity of the crankshaft.

According the equation above a linear increase of the power could be expected with increasing engine speeds and a constant mean effective pressure [10].

The effective mean pressure is determined with [9]:

$$p_{me} = \eta_e \cdot H_{uMix} \cdot \lambda_L \quad (2.16)$$

With H_{uMix} as the calorific value of the air fuel mixture, λ_L as the volumetric efficiency and η_e as the effective efficiency.

The volumetric efficiency for a gasoline engine is calculated by:

$$\lambda_L = \frac{V_{mix}}{V_e} \quad (2.17)$$

With V_{mix} as the volume of the air fuel mixture. With increasing engine speed there is a higher flow resistance in the cylinder that is leading to a decreasing volume of the air fuel mixture.

The effective efficiency is calculated by:

$$\eta_e = \frac{1}{b_e \cdot H_u} \quad (2.18)$$

With b_e as the specific fuel consumption and H_u as the calorific value of the fuel. The calorific value of the air fuel mixture is calculated by:

$$H_{uMix} = \frac{H_u \cdot m_{Fuel}}{V_{mix}} \quad (2.19)$$

With m_{Fuel} as the mass of the fuel. Therefore, the effective mean pressure is:

$$p_{me} = \frac{m_{Fuel}}{b_e \cdot V_e} \quad (2.20)$$

Hence the torque at the crankshaft is:

$$T = \frac{m_{Fuel} \cdot i}{b_e \cdot 2 \cdot \pi} \quad (2.21)$$

Since the volume of the air fuel mixture is decreasing, also the fuel mass is influenced by this behavior. The torque is therefore dependent by the mass of fuel in the cylinder and the specific fuel consumption, which also varies with the engine speed. Therefore, an engine map is not comparable to a linear curve, instead especially at higher engine speeds there are additional losses. According to van Basshuysen and Schäfer [11] the full load curve of the effective mean pressure over the engine speed is different whether the ICE is a Diesel or Gasoline engine. For the corresponding engine types the following curves of Figure 2.3 are exemplary [11].

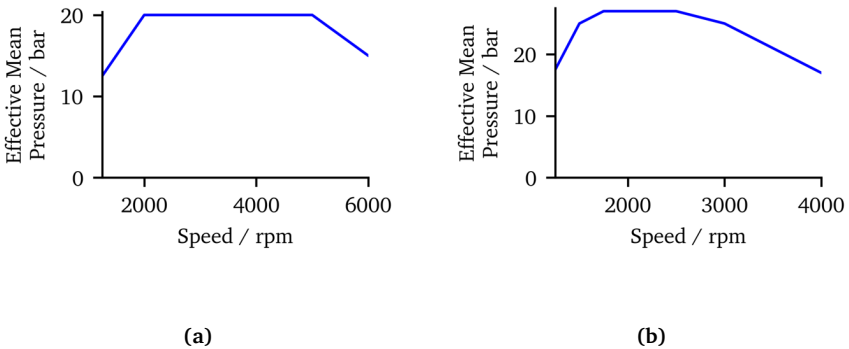


Figure 2.3. Effective mean Pressure: **(a)** Gasoline engine, **(b)** Diesel Engine according to [11]

Based on Equation (2.14) from these values of the effective mean pressure the engine torque can be derived. For the exemplary visualization of the torque a cylinder swept volume of $V_e = 1.8 \text{ l}$ is

assumed. For the four stroke engines the revolutions per stroke is $i = 0.5$. This leads to the following engine characteristic.

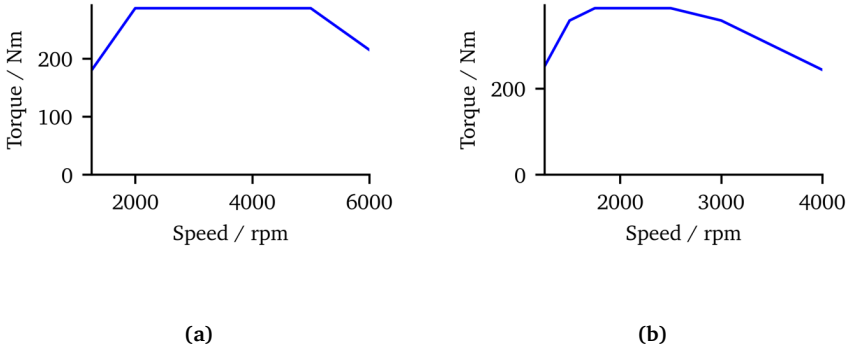


Figure 2.4. Engine torque behavior: **(a)** Gasoline Engine, **(b)** Diesel Engine based on [11].

With these curves in mind the following curve of Figure 2.5 is set up with fictive values for illustrating the engine behavior in interaction with the fictive transmission of Section 2.4.

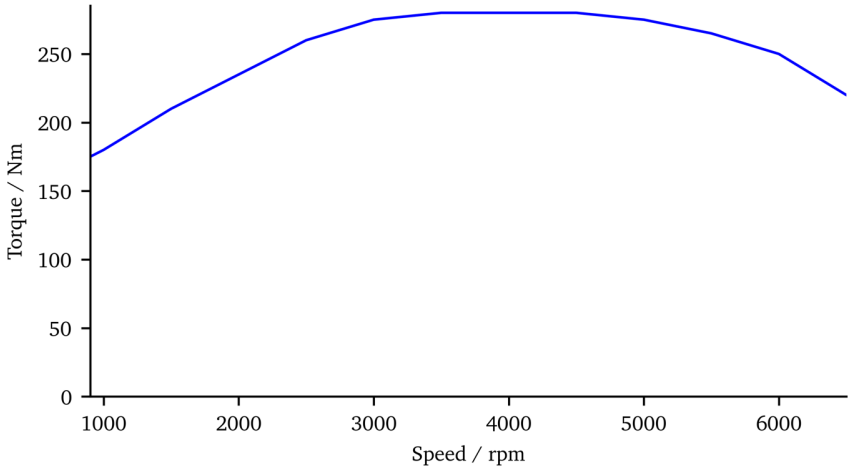


Figure 2.5. Fictive torque map.

By multiplying the torque of Figure 2.5 with the angular velocity, the power is calculated which is

illustrated with the following map (with fictive values).

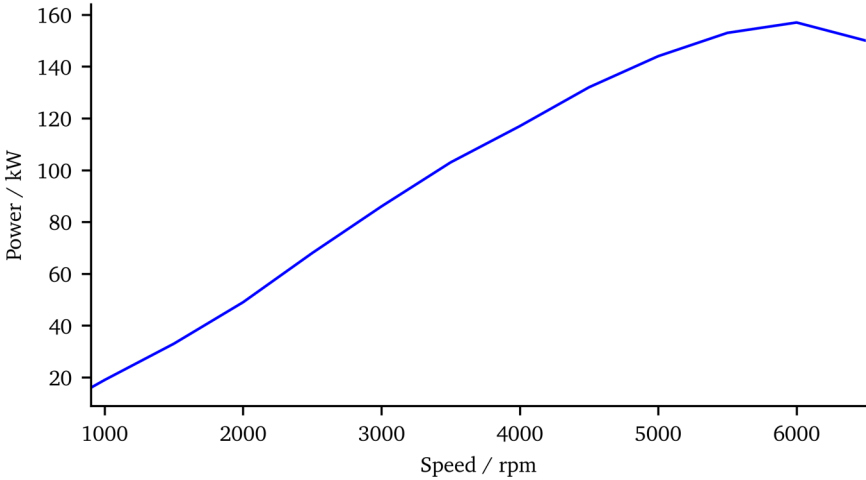


Figure 2.6. Engine Power over engine speed.

The typical behavior of an ICE is illustrated in Figure 2.5 and Figure 2.6. It demonstrates that the optimal operating mode is dependent from the engine speed.

2.2.2 Combustion Engines – The purpose of the transmission

The fictional engine behavior of Figure 2.6 illustrates a typical behavior of an ICE. It is observable that the maximum power can be determined at 5.500 rpm with 127 kW. Based on this maximum power the optimal tractive effort can be determined with the vehicle speed:

$$F = \frac{P_{max}}{v} \quad (2.22)$$

The equation above results in the red curve of Figure 2.7. The blue curve of Figure 2.7 is indicating the actual tractive effort of the ICE.

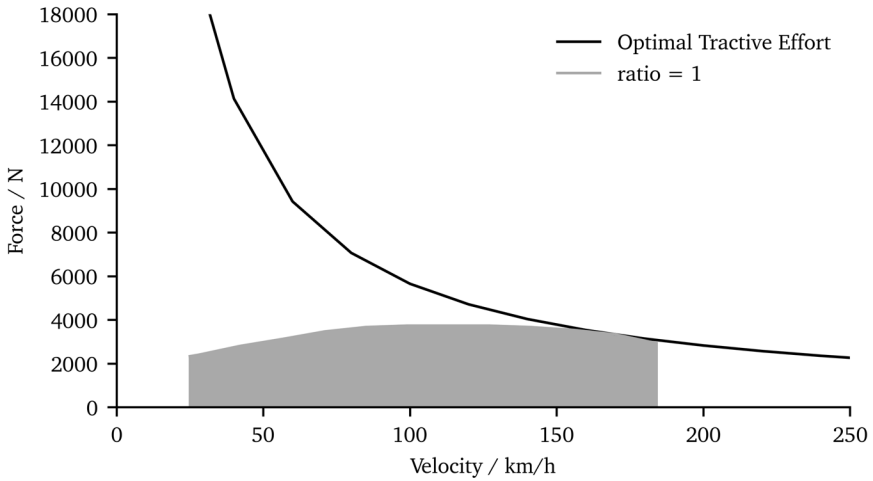


Figure 2.7. Optimal Traction effort and the engine behavior based on [12].

The grey area below the blue curve marks the possible operating points of the ICE and illustrates that not every operating point in the optimal tractive effort map is reachable without a transmission or with a transmission with only one gear and a fixed ratio of one [12].

Therefore, the main purpose of a transmission is to transform forces, torques and speeds onto a desired level since ICEs have limited operation modes [12], [13]. For this aim the gears of a transmission are designed to fit a desired gear ratio. The gear ratio of a transmission is determined by the ratio of the speed of the input-shaft and the speed of the output-shaft [12].

$$i_g = \frac{n_{in}}{n_{out}} \quad (2.23)$$

But also, by the ratio of the output torque and the input torque as well as the ratio of the number of teeth z of a gear.

$$i_g = \frac{T_{out}}{T_{in}} = \frac{z_{out}}{z_{in}} \quad (2.24)$$

Based on the example of driving resistance of Figure 2.7 the gear ratio and the number of gears should be adjusted in a way that most operating points are reachable. Therefore, the gear ratio of the first gear is estimated with 4 and the gear ratio of the differential transmission is estimated with 4 as well. The other gears are set in the following table:

Table 2.2. Fictive Gear ratios (estimation based on [5])

Description	Value				
Differential gear ratio	4				
Gear ratio	4	3.2	2.4	1.6	0.8
Total gear ratio	16	12.8	9.6	6.4	3.2

By combining these gear ratios with the map of the fictive ICE from Section 2.2.1 the curves for each gear can be determined. It is observable that a wider space of operating points can be reached by the ICE. These curves are illustrated in Figure 2.8.

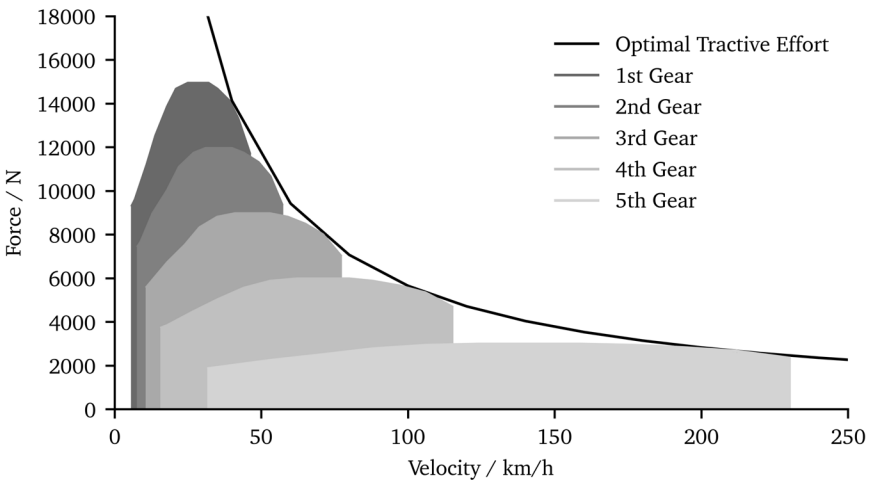


Figure 2.8. Enhanced operating points by deploying a transmission.

It is observable that compared to Figure 2.7 almost the entire area below the optimal tractive effort curve is now reachable with the combination of the ICE and the transmission (the white areas are still not reachable). The behavior could still be improved with more gears.

By comparing the curves with the driving resistance of Figure 2.2 the actual limitations of the vehicle can be investigated in Figure 2.9.

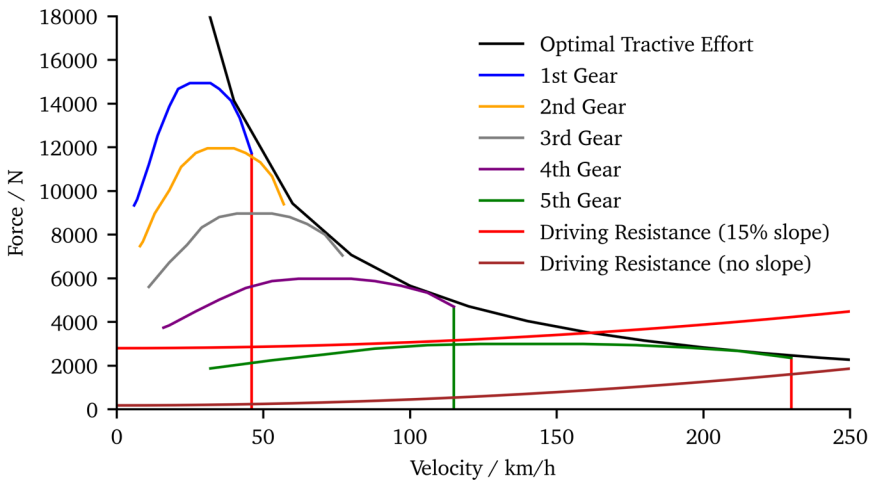


Figure 2.9. Provided Engine Force and Driving Resistance

Since there is not any intersection of the curves from the engine with the driving resistance (without slope – dark red curve) the maximum velocity of the vehicle is limited by the maximum speed of the engine.

Observable is that the maximum velocity with the first gear is at about 44 km/h and the maximum velocity with the fifth gear is at about 230 km/h (indicated with the red vertical lines). The map illustrates that different gear ratios lead to different velocity profiles (the higher the gear ratio the lesser the maximum velocity but the higher the force transmitted to the wheels). With the high forces of the first gear, it is possible to perform launches at slope but not to drive with high velocities. With the fifth gear instead, it is possible to drive with high velocities but at higher driving resistances (e.g., due to slopes) the shifting into a lower gear is required. Therefore, with increasing slope or wind the maximum velocity is limited (green vertical line). The red curve indicates a driving resistance with 15% slope and illustrates that driving in fifth gear is not possible at these slopes. The maximum velocity is limited to 115 km/h. Hence, the gear ratio and the transmission itself strongly influences the maximum velocity and the desired usage of the vehicle.

If a vehicle is accelerating, it also has an influence on the resulting driving resistance. The torque needed to accelerate a vehicle with 2.5 m/s^2 to 10 km/h is about 2.050 Nm at the axle. The simple example illustrates the need of torque conversion especially during launch. With a gear ratio of 4 [5] for the first gear and also a gear ratio of 4 for the differential transmission the engine has to provide a torque of 137 Nm for performing the desired launch. Due to the high level of torque needed to launch a vehicle it is necessary to run a passenger car with a transmission as a torque transformer. From the torque needed the force to launch the vehicle can be determined to 6.840 N. Figure 2.10 illustrates the driving resistance and the possible provided force by the engine.

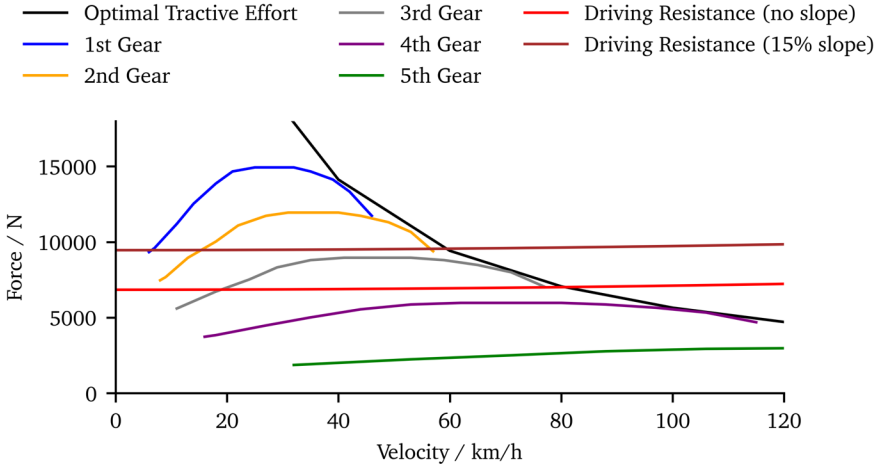


Figure 2.10. Gear Selection based on the Driving Resistance

The driving resistances with and without slope indicate the force needed to launch the fictive vehicle. Observable is a launch in first, second and third gear is possible since the force the engine provides is higher than the force needed for a launch without slope.

By increasing the slope to 15% the driving resistance is increasing further. Hence, a launch is not possible in third gear anymore. Thus, the transmission is required to compensate the limited operating modes of ICEs.

2.2.3 Electric Motors

Despite the fact that the study is not specifically focused on the calibration of transmissions of electric vehicles (EVs), this Section serves as a quick overview. The behavior of the torque of an electric motor is very different compared to the one of an ICE. A typical behavior is illustrated in Figure 2.11 based on [14].

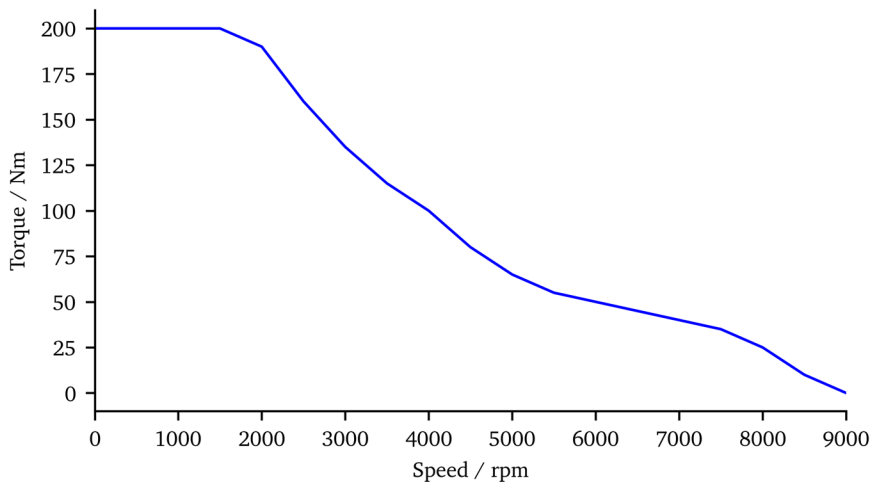


Figure 2.11. Torque map of an electric motor based on [14].

It is observable that the maximum torque is available at zero rpm. According to Naunheimer [12] this is an advantage compared to ICEs, since ICEs have the drawback of not being able to provide torque during standstill [12]. Therefore, EVs do not require a clutch for launching a vehicle. Further, most EVs not even have a multi-speed transmission like the Nissan Leaf, the Tesla Model S or the Chevrolet Spark. Instead, a single speed transmission and therefore a transmission with a fixed gear ratio is installed [15]. According to Vynakov [16] the Tesla Model S P85 has an electric motor with a gear ratio of the final drive of 9.73 at the rear axle. Figure 2.12 illustrates the torque map of the Tesla Model S based Sieklucki [17].

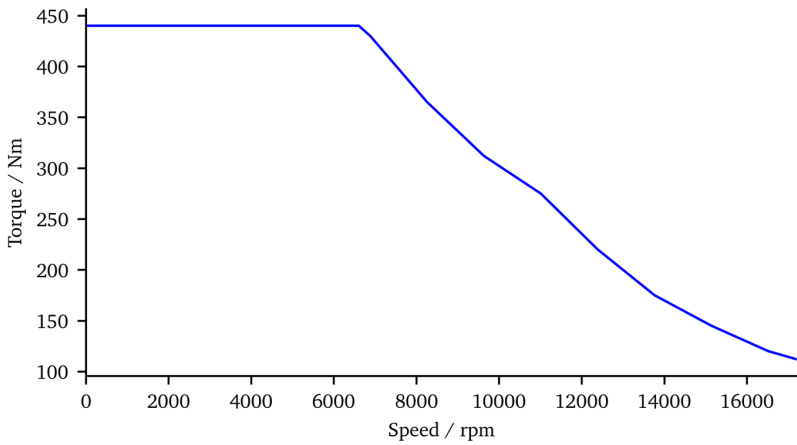


Figure 2.12. Torque map of the Tesla Model S based on [17].

From this map and the gear ratio the optimal tractive effort can be derived which is illustrated in Figure 2.13 without considering wheel slip thresholds.

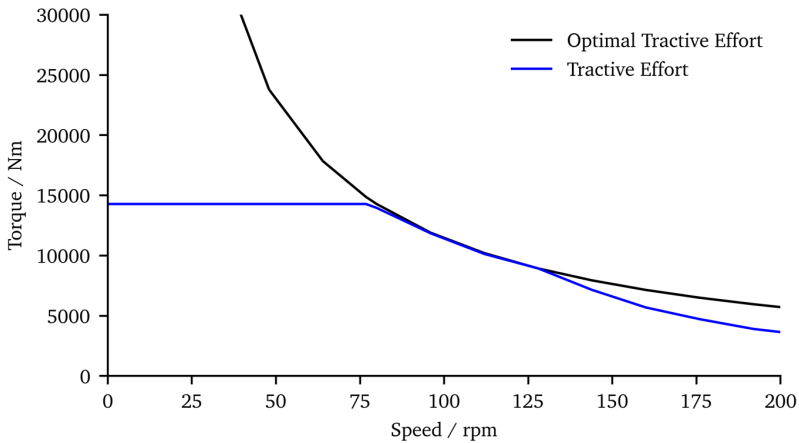


Figure 2.13. Optimal tractive effort of the Tesla model S derived from [16] and [17] without considering wheel slip thresholds.

The optimal tractive effort of Figure 2.13 is again determined with Equation (2.12). It is observable

that in Figure 2.13 the tractive effort at lower speeds does not reach the optimal tractive effort.

According to Kollmeyer [15] an approach to overcome the issue is to also apply a transmission similar to powertrains with ICEs to provide high wheel torques at low speeds but additionally maintain high top speeds. A further advantage of applying a multi speed transmission to the electric drivetrain is that it comes with the possibility of downsizing the motor of the electric vehicle (EV) and reducing the energy consumption [18]. Therefore, a two speed transmission is already applied in the Porsche Taycan and in the Rimac concept cars [19].

2.3 Hybrid Vehicles

Hybrid vehicles combine multiple engine types usually ICEs and electric motors. The hybridization has several advantages for reducing the emissions of a vehicle [20]. Therefore, for reducing emissions different levels of hybridization exist according to Chan et al. [21]. These levels are micro hybrid, mild hybrid, full hybrid and plug-in hybrid (PHEV). Figure 2.14 illustrates the different levels based on the level of electrification according to Naunheimer [12]. The longer the grey bars are the more electric energy is used for the hybrid functions. Therefore, the micro hybrid covers the least hybrid functionality compared to the other types.

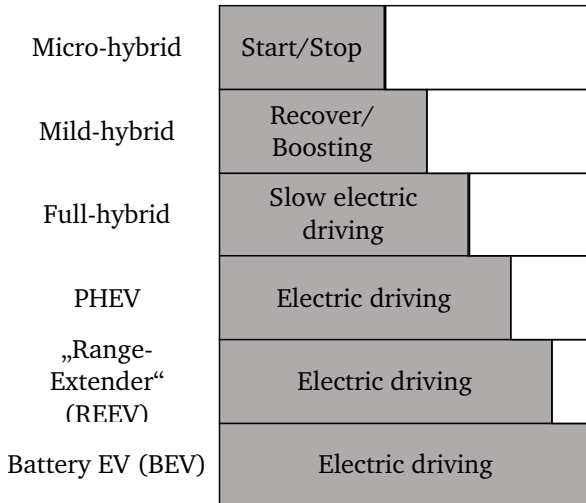


Figure 2.14. Types of hybrid Vehicles based on [12].

The next higher levels of hybridization include the functionality of the prior ones. The notation is only introduced to outline the functionality which comes with the next higher level of electrification. This means the functionality of the mild hybrid also includes the functionality of the micro hybrid. The length of the bars is chosen exemplary and not determined by the actual power ratio.

In contrast to micro-, mild- and full hybrids the battery of the PHEV (and those of the levels above) has the ability to be charged externally. If the battery can be additionally charged by the ICE the hybrid vehicle is a range extended EV [21].

2.3.1 Hybrid functions

The hybrid functions of Figure 2.14 are further illustrated in this Section based on Reif et al. [22].

Electric Driving

One advantage is that the vehicle can have the ability of driving with electrical energy which improves the fuel consumption of the vehicle and hence the overall emissions. Therefore, if the strategy enables the vehicle to drive with electric energy only the energy is provided by the battery. If the energy capacity of the battery is too low or if the driver demand requires additional power the ICE can be switched on [22].

Start Stop

In hybrid powertrains the ICE is often switched off if there is no torque demand of the ICE. One situation is e.g., when the vehicle is in standstill the ICE is usually in idle which leads to emissions without providing torque to the axle. Therefore, switching the ICE off during the standstill reduces the emissions but the vehicle still requires power e.g., to maintain a functioning air condition. This power can be provided with the electric engine. Further also the engine can be switched off during braking or coasting. The ICE is switched on again with the help of the electric motor [22].

Boosting

The electric motor also can be used in parallel with the ICE. One use case is boosting. While driving with maximum power provided by the ICE the electric motor can be used to increase the overall power and therefore the torque to assist the ICE and hence fulfilling the drivers request e.g. for a higher demand of torque during vehicle acceleration [22].

Load point shifting

As explained in Section 2.2.1 the operating points of the ICE are very limited. Further the optimal fuel consumption is dependent on the load point of the ICE. The highest amounts of torque the ICE can provide are exemplary illustrated with the load points of Figure 2.5. The dependency of the specific fuel consumption on the load points are exemplary illustrated in Figure 2.15. The values are chosen arbitrarily and are not realistic. The characteristic of the efficiency map is based on [23].

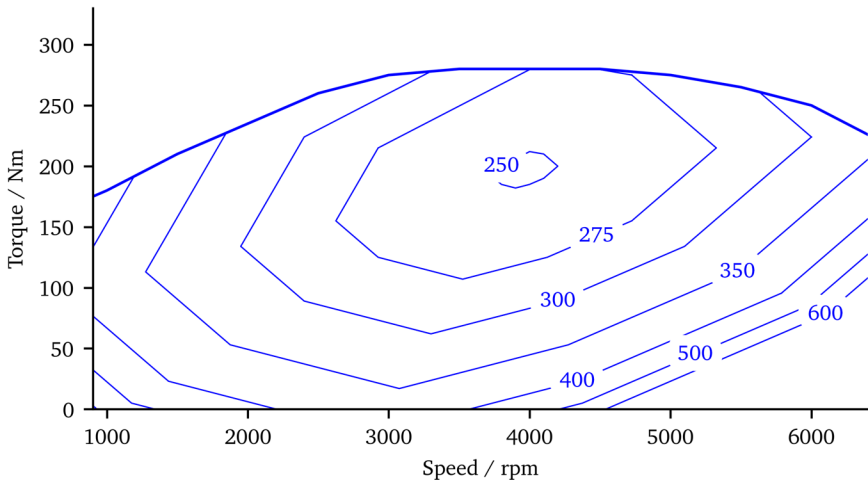


Figure 2.15. Efficiency map of an ICE based on [23] illustrating the specific fuel consumption in g/kWh.

With the load point shifting the torque of the ICE is shifted to a more beneficial point in the efficiency map to reduce the fuel consumption. Therefore, the ICE is used to fulfill the driver request and the torque exceeding the driver demand is used to charge the battery. With the increased state of charge the electric motor can further be used in other driving situations to reduce the fuel consumption [22].

Energy recovery

For the energy recovery the electric motor is operated in generator mode. For decelerating the hybrid vehicle, the mechanical brakes are only used partially. With the generator mode of the electric motor, it is possible to buffer kinetic energy within the battery. With a control unit the desired brake torque is divided into the torque from the mechanical brake and the torque required by the electric motor.

2.3.2 Hybrid Topologies

To realize the hybridization there are multiple options to mount the electric motor within the powertrain. The following topologies illustrate different options which are illustrated in Figure 2.16.

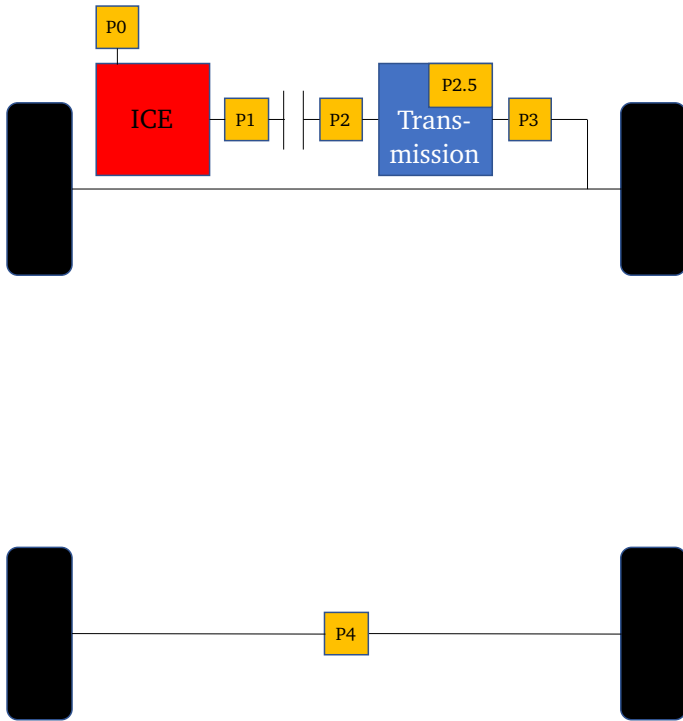


Figure 2.16. Hybrid topologies according to [22] and [24].

P0 Hybrid

The electric motor is connected to the ICE with a belt. With this configuration the electric motor has the ability to start the ICE [24].

P1 Hybrid

In a P1 hybrid the electric motor is mounted at the crankshaft before the clutch. The P1 hybrid also has the ability to start the engine but additionally enables boosting. Through boosting the torque especially at lower engine speeds can be increased [24].

P2 Hybrid

The electric motor within a P2 hybrid is mounted between the ICE and the transmission after the clutch [22] and offers in contrast to the former topologies the possibility of electric driving and load point shifting [24].

P2.5 Hybrid

Another solution to implement an electric motor in the powertrain is the P2.5 hybrid where the electric motor is mounted on one shaft of the dual clutch transmission [25]. This hybrid configuration is further explained in Section 2.4.3.

P3 Hybrid

The P3 hybrid has the ability to avoid a dropping acceleration during gear shifts since the electric motor is mounted at the output-shaft of the transmission. Therefore when the clutch is opened during gearshifts and the torque of the ICE cannot be transmitted to the wheels the electric motor provides torque to ensure a comfortable driving behavior [24].

P4 Hybrid

Within a P4 hybrid the electric motor is not coupled to the ICE and the transmission instead one axis is driven by the electric motor. This enables the possibility of an all-wheel drive where one axis is driven by the electric motor solely [24].

2.4 Transmission

As it has been outlined in Section 2.2 and 2.2.3 transmissions are needed in powertrains with ICEs and preferable to be installed in powertrains with electric motors the scope of this Section is on the transmission itself.

To shift the gears of the transmission and hence to change the gear ratio, the transmission has to be decoupled from the engine with a clutch. The clutch is also used to launch a vehicle since as mentioned previously in Section 2.2.3 an ICE is not able to provide torque during standstill. Instead, the engine speed has to be greater than zero to launch a vehicle, so the clutch reduces the slip between a standing vehicle and a rotating engine. Therefore, the engagement of the clutch has a crucial role.

2.4.1 Clutch Engagement

The clutch is the element which connects the engine to the transmission (and the driven axle) and thus transfers the torque and speed of the engine (Figure 2.17) [4].

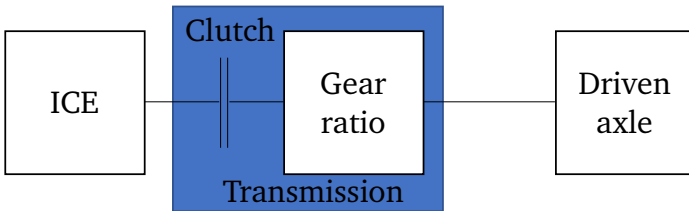


Figure 2.17. Clutch configuration of manual transmissions (based on [4])

For gear shifts the torque of the engine must be decoupled from the transmission to synchronize the desired gears. To decouple the engine from the transmission the clutch needs to be opened until the engagement of the gear is finished. After the gear engagement the clutch is closed again. During the engagement of the clutch (after the gear shift) the engine has a different speed than the transmission input-shaft speed since the gear ratio has been changed during the gear shift. The speed difference is called slip. The slip speed needs to be zero to transmit the entire power of the engine to the wheels. To reduce the slip speed the engine speed is adjusted to the new gear ratio. Since, the connection between the engine and the transmission is lost during gear shifts the longitudinal acceleration collapses due to the lack of transmitted engine torque to the wheels in manual transmissions [5]. The engine torque of a powertrain with manual transmission is illustrated in Figure 2.18.

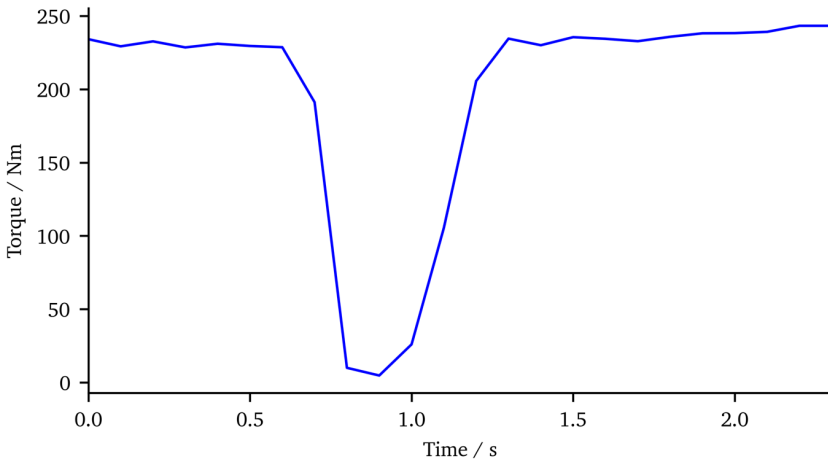


Figure 2.18. Engine torque behavior for manual transmissions (based on [5])

To avoid this behavior the DCT is invented.

2.4.2 Dual Clutch Transmission (DCT)

The advantage of the DCT is the ability to perform gear shifts without reducing the load of the engine. This is realized with the installation of two clutches. The transmission itself is divided into two parts. One with odd gears attached and one with even gears. Each clutch is connected to one of those gear sets e.g. clutch 1 is connected to the 1st, 3rd and 5th gear and clutch 2 is connected to the 2nd, 4th, 6th and the reverse gear as illustrated in Figure 2.19 [5].

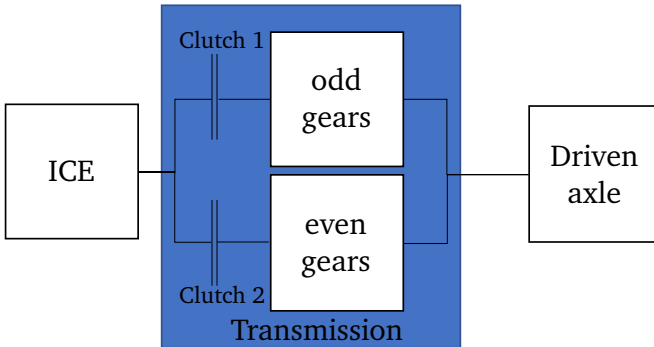


Figure 2.19. Clutch configuration of DCT (based on [5])

With this transmission configuration it is possible to switch from an odd to an even gear or vice versa without reducing the load. Therefore, the next lower or higher gear is pre-selected. If e.g., the vehicle is running in second gear either the first or the third gear (depending on the shift strategy and the engine speed) is already engaged so the torque is blended from clutch 2 to clutch 1. So, during a gear shift the clutch torque of the former active clutch is decreased and simultaneously the clutch torque of the new active clutch is increased. Therefore, the sum of both clutch torques remains steady. This leads to the advantage of avoiding torque interruptions which are typical for gear shifts with automatic manual transmissions (AMTs). Hence the longitudinal acceleration does not drop uncomfortably. The torque behavior is shown in Figure 2.20 [26].

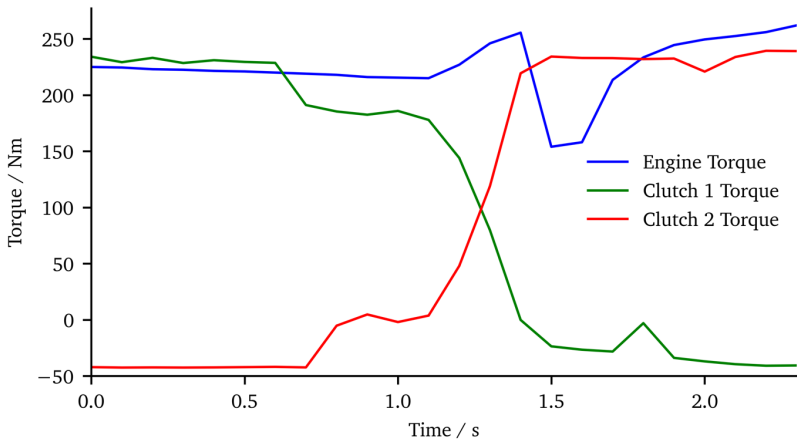


Figure 2.20. Clutch torque behavior during the shift with a DCT (based on [26]).

When the clutch torque of the new gear is in a steady state an engine intervention is performed (engine torque is decreased under the clutch torque) to decrease the engine speed to fit the new gear ratio. The behavior of the engine speed is shown in Figure 2.20.

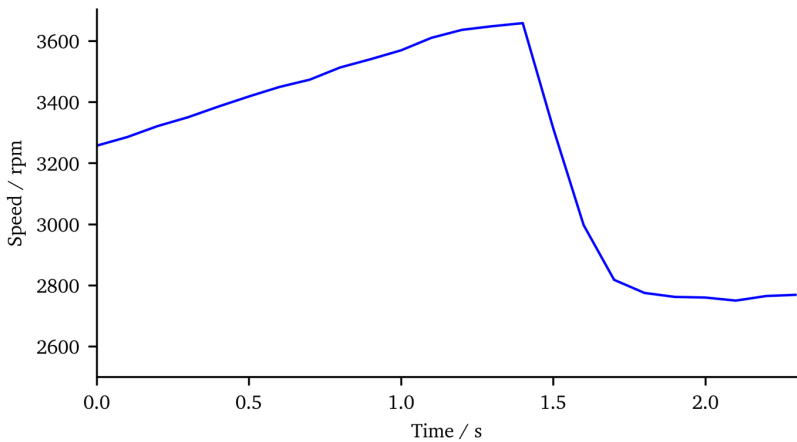


Figure 2.21. Engine speed behavior during the gear shift (based on [26]).

Shifting with the DCT is, therefore, more comfortable than with manual transmissions (MT) or

with an AMT. Another advantage is the better efficiency and hence the better fuel consumption compared to a continuously variable transmission (CVT) and transmissions using a torque converter. This results in a comfortable drivability but also enables the vehicle to be sporty, responsive and efficient like a manual transmission [27]. An advantage of the DCT is the ability to vary parameters to fit in several vehicles under different driving conditions. The flexible application is a result of the fact that the DCT needs much more calibration work compared to an automatic transmission with torque converter [12], [28].

2.4.3 Hybrid Dual Clutch Transmission (HDT)

Since the legislative requirements regarding the CO₂ emissions in Europe became stricter there is also an approach to enable the conventional DCT (Section 2.4.2) to a hybrid system. This approach led to the hybrid dual clutch transmission (HDT) [29]. The HDT is a DCT with an electric motor applied. It has the ability to transmit torque with an electric motor if the second of the two clutches is closed [25]. As mentioned in Section 2.3.2 the HDT has a P 2.5 hybrid topology. This topology is illustrated in Figure 2.22 according to [25].

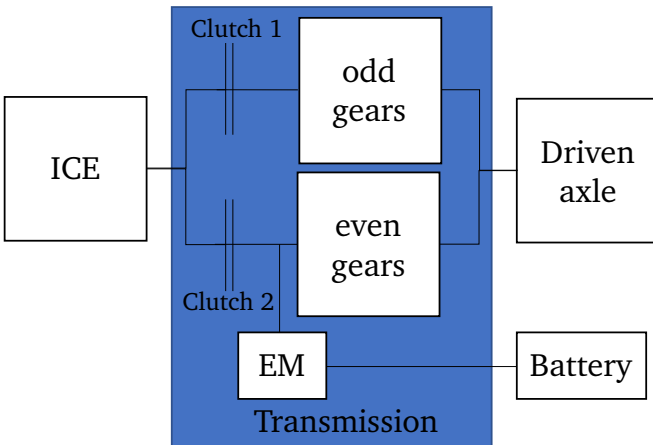


Figure 2.22. HDT / P 2.5 hybrid topology according to [25].

With this transmission it is possible to enable multiple hybrid functionalities like the load point shifting and the start of the engine during standstill but also during electrical driving (after the ICE is switched off during coasting) (see Section 2.3.1) [29]. The electric motor can be used if the ICE is operated with an engaged odd gear and by closing clutch 2 without engaging an even gear [30]. If clutch 2 is open and an even gear is engaged it is also possible to use the electric motor [29]. With the configuration of the HDT the fuel consumption can be decreased compared to a vehicle equipped with a DCT of about 14.5% [31].

2.5 Transmission Control Unit (TCU)

There are several control units installed in a common vehicle e.g., the ECU (which is calculating the engine torque, etc.), the TCU, control units for the suspension, steering and many others which are necessary to control the behavior of a vehicle. The clutch engagement in an automatic transmission is realized with functions implemented on the TCU. With these functions the clutch engagement for shift and launch events are realized and the clutch pedal can be omitted [12]. Any signal which is important for the control of the clutch and the gear engagement is transmitted to the TCU e.g., the engine speed and torque are received from the ECU or from sensors. These signals are processed by the Software implemented on the TCU. The software calculates output signals for other control units and signals for actuators [5].

2.5.1 TCU Architecture

The operations are performed on the microcontroller of the TCU. The architecture of the microcontroller itself consists of a central processing unit (CPU), non-volatile memory like Read only memory (ROM), erasable programmable read-only memory (EPROM), Flash-EPROM or electrically erasable programmable read-only memory (EEPROM) and a working memory like random access memory (RAM) [32]. According to [33] the ROM is a non-volatile memory so it can keep data if the system is powered off but is not erasable. It contains the operating program which is executed by the CPU first after the system is switched on [33]. The EPROM and the EEPROM can keep data during the system is powered off like the ROM (non-volatile) but these memories are erasable [34], [35]. The EEPROM has the advantage that the memory is byte wise replaceable compared to the EPROM [35]. The RAM is a volatile memory so it loses its data when the system is powered off [33].

The TCU uses the ROM for hardware specific software components, the Flash-EPROM is used for flashing updates of the TCU software and can be deleted and rewritten e.g., during a vehicle service. Adaption values and errors are stored in the EEPROM. The RAM memory can be read but also be written and is used to store data which is needed during the operation [12].

Since the electronic hardware is deployed under different ambient conditions it must fulfill special requirements. It must be ready to be used at temperatures from -40 °C to +140 °C, under vibrations and contamination of oil and other fluids [12].

2.5.2 Transmission functions

The microcontroller is used to run the TCU software which can be divided into subsystems. Figure 2.23 illustrates the common function structure of a TCU according to [36].

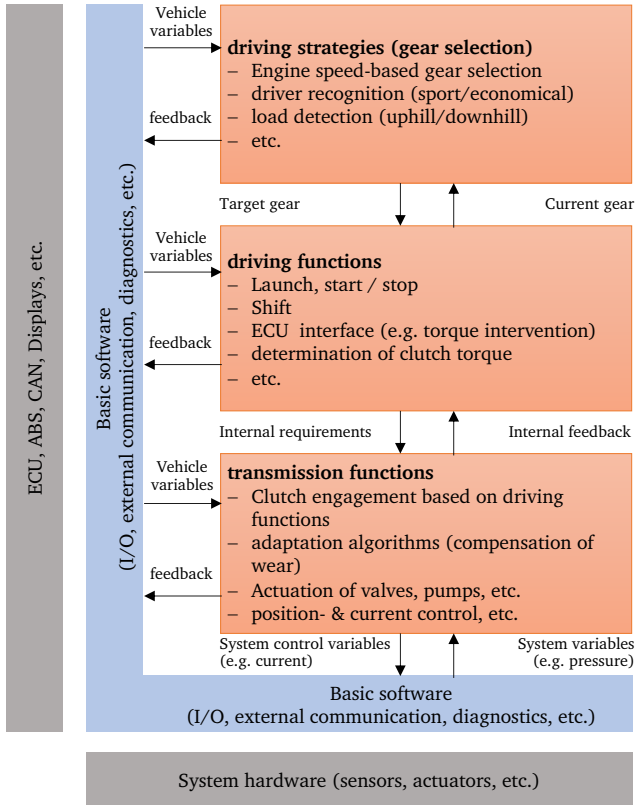


Figure 2.23. TCU function structure based on [36], [37]

The raw signals from sensors and other control units must be pre-processed before they are passed to the software of the microcontroller which is performed from the basic software [36].

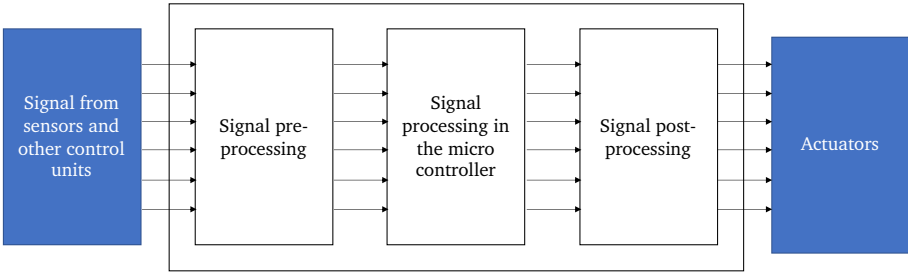


Figure 2.24. signal structure of a TCU based on [5]

The pre-processing of the raw signals can include e.g., the filtering of the signal, setting signal boundaries, etc. During the signal pre-processing the signals can be monitored to detect failures hence the pre-processing is used for diagnostics. These sensor signals are usually electrical voltage or current signals which are converted to the desired signals units for e.g. pressure or temperature [12].

The driving strategy is used to determine the gear depending on the driving situation and the driver intention. The driving situation is influenced by the vehicle velocity and the detection of the driving resistance which can influence the gear selection. Since the vehicle velocity can get slower although the acceleration pedal remains constant the driving resistance probably is increasing e.g. due to slope which could force a downshift. If the vehicle velocity is increasing an upshift is forced. The driver can also influence the driving strategy through the variation of the acceleration pedal. If the acceleration pedal is increased the driver signals the demand of a higher acceleration (e.g., for overtaking another vehicle) which leads to a downshift. The driving strategy is also influenced through the different driving modes e.g. eco, sport, normal which set the vehicle velocity thresholds for the shift strategy depending on the drivers selection [36].

The driving functions can be divided in several subfunctions for launch, creep, upshift, downshift, start/stop, slip control, etc. With these functions the calibration engineer can set values to meet customer defined targets. Therefore, it is possible to set the desired torques which shall be transmitted with the clutch for each driving situation and hence to influence the drivability through the determination of the clutch engagement [36].

The transmission functions include the clutch engagement through providing pressure to meet the desired clutch torque determined within the driving functions, engaging gears and the pump actuation to provide the clutch cooling. It also consists of adaption functions to consider the clutch wear. Since the clutch is actuated with pressure there is a correlation factor between the clutch pressure and the clutch torque. The correlation factor is altering due to the changing friction coefficient of the clutch i.e. the pressure to actuate the clutch in the same way is different at 0 km and 50000 km clutch life so the correlation factor needs to be adapted [36].

The functions are developed using typical programming languages like C, C++, Assembler [36] or model-based programming languages. To influence the characteristic of a vehicle the functions include parameters to calibrate the vehicle [12]. The calibration is used to optimize the clutch

engagement depending on the vehicle type but with the same software. With these parameters it is therefore possible to consider the different vehicle weights, engine specifications and other factors influencing the clutch engagement.

2.6 Calibration

The aim of the calibration is to fit transmissions to different combinations of vehicles, engines, gear sets and markets. For this purpose the parameters of the driving functions of the TCU are adjustments considering the customer requirements and the legislative requirements of the market [5].

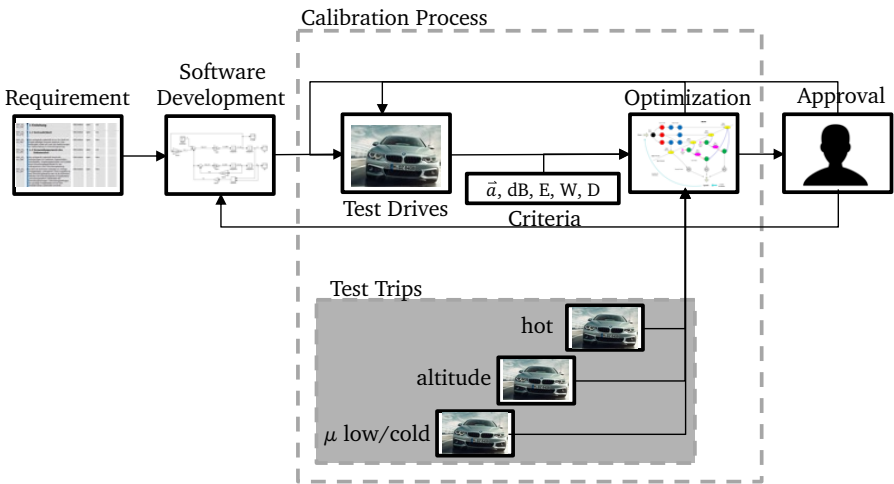



Figure 2.25. Simplified Calibration process

Figure 2.25 describes the simplified calibration process. The software of a transmission is usually based on requirements created by the customer or by the transmission manufacturer itself. During test drives under different ambient conditions the transmission is calibrated to meet the requirements in any situation. After an approval the calibration is completed, or another development loop is performed to meet the requirements. This process is usually performed by several calibration engineers in an iterative way during test trips [5]. The optimization of the parameters of the TCU software is performed to meet different criteria like reducing vibrations, fuel consumption, emissions, ensure comfort and/or sportiness, safety and avoiding damage [12]. Within these trials the calibration engineers try to find the parameters which fits the requirements. The optimization of these calibration parameters is based on subjective feelings of the calibration engineer [26].



The process is performed for every demanded transmission / engine / vehicle variant of the customer.

The calibration process of a DCT is driven by the system optimization under meeting internal, customer and legislative requirements. With regard on the optimization of the launch behavior, the goal is the optimization of the clutch engagement.

The driver is expecting a quick response without acceleration inconsistencies during the clutch engagement throughout the entire lifetime of the clutch and under each driving condition [38]. To achieve these customer requirements parameters are introduced into the TCU software to calibrate the behavior of the vehicle. Since a DCT enables jerkless shifts with its two clutches it also requires a greater calibration effort due to the complex architecture and actuation compared to torque converters. Also, legislative requirements lead to new powertrain concepts with electrified powertrains and hence to a wider range of engine, transmission and vehicle combinations, which also result in an increasing calibration effort [28]. The variety of these combinations are necessary to fulfill the customer demands for different markets but lead to increasing test trips and engineers involved. With new requirements (like legislative requirements), and therefore growing software functions and calibration parameters, the calibration process is getting even more expensive. During these test trips the calibration engineer has to optimize parameters in an iterative manner to improve the driving behavior [5]. This conventional process is very time consuming [39] and is getting even more expensive with further requirements since also the number of software functions are increasing which can lead to further calibration parameters.

Vehicle Launch

As outlined in Section 2.2 the clutch is also necessary to launch a vehicle with ICEs. During standstill the clutch is disengaged. By engaging the clutch, the connection between the engine and the transmission leads to a moving vehicle [12]. Such as during shift events, the slip between the engine speed and the input-shaft speed of the transmission must be reduced while launching a vehicle with a clutch to fully connect the engine to the driven axle. Before the vehicle is launched, it is (as mentioned earlier) in standstill or in low velocities and the engine is idling. Further the launch process is divided into three phases which are observable in Figure 2.26 [40]:

1. The engine and clutch torque are increased. The clutch torque is lower than the engine torque, so the engine speed is increasing to reach the desired launch speed. With increasing clutch torque, the vehicle starts accelerating. The launch speed and the engine torque are dependent on the driver demand (left to the red marked area – second 0 to 0.8).
2. The engine and the clutch torque are equal, so the engine speed remains steady until the input-shaft speed of the transmission is almost equal to the engine speed (red marked area – second 0.8 to 1.2).
3. The clutch is opened slightly to increase the engine speed for engaging the clutch with almost equal speed gradients of the engine speed and input-shaft speed (for a comfortable engagement) (right to the red marked area – second 1.2 to 2).

After the engine speed is equal to the input-shaft speed the launch is finished. At this point the engine torque is accelerating the vehicle with closed clutch. The process is illustrated in Figure 2.26.

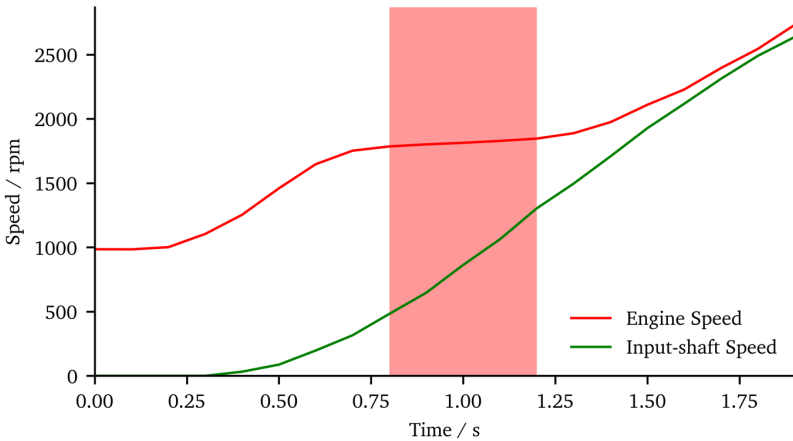


Figure 2.26. Engine speed (red) and input-shaft speed (green) behavior during vehicle launch.

The gradient of the engine speed is during slip phases dependent from the difference of the engine torque and the clutch torque. If e.g., the engine torque is increasing, and the clutch torque is zero (transmission in neutral) the engine speed is increasing faster than with clutch torque applied since there is no resistance from the road. In general, the engine speed is increasing, if the clutch torque is lower than the engine torque. The greater the difference of the engine and the clutch torque the faster the engine speed is increasing. Accordingly, the engine speed remains constant if the clutch torque is equal to the engine torque and if the clutch torque is greater than the engine torque the engine speed is decreasing.

The behavior is illustrated in Figure 2.27. In the red area, the clutch torque is equal the engine torque, so the engine speed is not increasing. In the other areas it is observable that the clutch torque is lower than engine torque and that the engine speed is increasing.

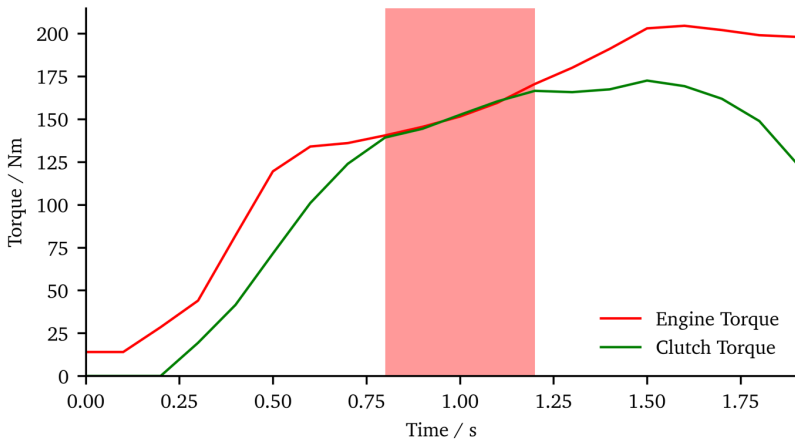
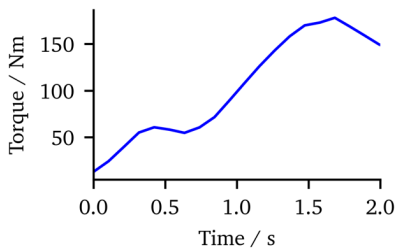


Figure 2.27. Engine torque (red) and clutch torque (green) behavior during vehicle launch.

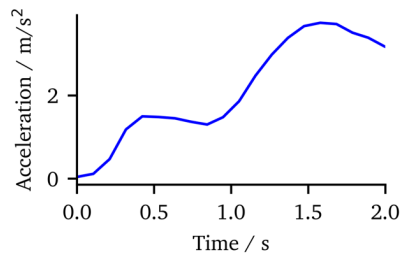
Therefore, if the clutch is closed too fast the clutch torque rises soon higher than the engine torque and the engine speed could fall below idle speed and urge the engine to stall.

Thus, the engine speed can be controlled by the clutch during clutch slip phases [5].

Further, the gradient of the clutch torque has an influence on the driving behavior during slip phases since it is transmitted directly to the wheels [37]. This fact leads to a relation of the clutch torque and the longitudinal acceleration. Figure 2.28 illustrates the similarities of the shape of the clutch torque and the longitudinal acceleration.



(a)



(b)

Figure 2.28. Relation of Clutch Torque and longitudinal acceleration: **(a)** Clutch torque, **(b)** Longitudinal acceleration

It is observable that the longitudinal acceleration is changing its gradient similarly to clutch torque gradient. After reviewing the importance of the clutch engagement, the urgency of a proper clutch control is illustrated in this Chapter. The control of the clutch is performed with the TCU.

3 Automizing the calibration – Known approaches (State of the Art)

The aim of this study is to find ways to reduce the tasks of a calibration engineer, to shorten the time on test trips and to generally shorten the development time. To fulfill this requirement the study strives to find methods to automize the calibration for reducing the effort of engineers.

According to Liao [41] the automated calibration can be classified into knowledge based methods, knowledge-free methods and model-based methods. Knowledge-based methods rely on logical rules set up by experienced engineers. These rules e.g., describe which parameters have to be changed based on specific criteria. An advantage of knowledge-based methods is that solutions are found fast but require experienced engineers. Knowledge-free methods in contrast have the advantage of not requiring experienced engineers in the system domain. Instead, stochastic algorithms are applied to find a set of optimal parameters with the disadvantage of a potentially long search process (e.g., genetic algorithms). Model-based methods rely on simulation models which can predict the system behavior. This prediction is used in optimization algorithms therefore the search process is shortened compared to entirely knowledge-free methods. The disadvantage is the need of a design of experiments (DOE) to create the required simulation models [41].

The method to automize the calibration is desired to be able to generalize for different calibration tasks without needing experienced engineers and pre-defined models. Therefore, knowledge-free methods are evaluated in this study. In contrast to the mentioned drawback of having a potentially long search process the method should also be able to find optimal solutions relatively fast to ensure the ability of applying the method in development processes in simulations as well as in test vehicles. Further the methods should be applicable for any kind of embedded software without being limited on a certain software structure. The methods applied in this study are introduced to optimize the launch behavior of a vehicle (equipped with a DCT or HDT) from standstill.

Since the conventional iterative calibration process is very time consuming, the automation of the calibration process has gained attention in research since it is expected to optimize the system in a shorter time period.

One option to automize the calibration process is the iterative model and trajectory refinement (IMTR) method used by Jennifer Hudson [42] to optimize the clutch trajectory for the launch behavior of a vehicle with a DCT. The optimization is performed considering constraints like energy input on the clutch, vehicle acceleration, engine speed bound, etc. With acceleration and jerk as the optimization criteria Zhao [43] introduced a PID controller to optimize control parameters for the clutch engagement of a DCT transmission during launch within a simulation environment. The model-based iterative learning control is deployed by Gregory Pinte [44] to optimize the quality of the clutch filling process during clutch engagement to reduce the common calibration work. Van der Heijden [45] and Horn [46] both optimize the clutch engagement for the launch behavior of automatic manual transmissions with different control strategies. Dolcini [47] optimized the clutch engagement by reducing the clutch slipping time with a sliding speed trajectory with an analytically derived controller.

There are also similar calibration approaches for optimizing the behavior of ICEs like the fuzzy

based optimization algorithms of Tino Naumann [48] to minimize the fuel consumption while complying emission thresholds. The algorithm has been verified in offline optimization loops and then applied at the test bench of a common-rail diesel engine. The fuzzy based method has also been applied to optimize the quality of the shift comfort for automatic transmissions [49].

However, these authors did not implement knowledge-free methods for optimizing the driving behavior. Beside knowledge-based approaches usually model-based approaches are proposed [41].

Since ML is gaining attention through optimizing the performance of common tasks through learning by past experience [50] this thesis investigates the potential of knowledge-free self-learning algorithms in scope of parameter optimization especially for clutch engagement.

3.1 Reinforcement Learning

Deep learning has gained attention by fulfilling tasks like speech recognition and object recognition with neural networks [51], which also led to advanced reinforcement learning (RL) algorithms. A famous example of a RL algorithm is the deep Q-learning (DQL) algorithm which is e.g. playing Atari Games [52]. In Q-learning the learning behavior of creatures is mimicked by the RL algorithm (called agent) which is a computational approach by making machines interact with an environment to obtaining information about the problem. The agent affects the environment through actions, and the environment reacts by adopting a state in response to the action which is fed back to the agent. The next action is chosen regarding the fed back state, to influence the environment towards a goal. The quality of an action is measured with the reward. A reward is high if an action leads to an environmental state toward an optimization goal and low if the state indicates a contrary behavior. The agent strives to maximize the reward and hence learns to take actions accordingly. Therefore, the aim is to increase the reward with actions of higher quality [53]. According to [52] the following structure of

Figure 3.1 represents the Q-learning scheme in order to illustrate the interaction of the actions with the environment and the state as the response of the environment.

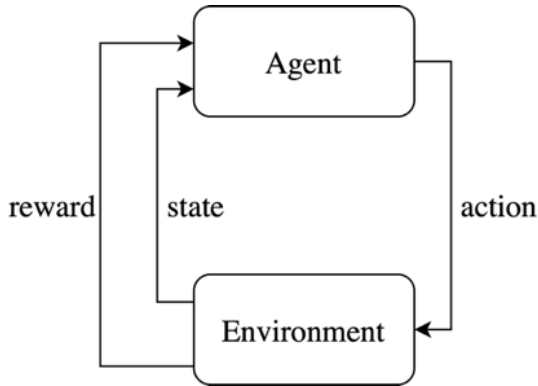


Figure 3.1. Q-learning structure according to [52].

3.1.1 Background and Motivation

Any RL algorithm consists of a policy which maps states and a possible set of actions which could be taken. A simple RL approach is based on lookup tables mapping these state action pairs [53]. A common task to test RL algorithms is the cartpole problem of Michie and Chambers [54]. The idea is to have a cart which can be moved left and right to balance a pole mounted to the cart. The task is fulfilled if the pole is kept upright by the movements of the cart and the collapsing is avoided. Thereby the cart can only be moved within a defined range. The states are therefore containing (1) x the position of the cart, (2) α the angle of the pole, (3) \dot{x} the velocity of the cart and (4) $\dot{\alpha}$ the rate of the change of the angle. The actions are the left and right movements [54]. The example is illustrated in Figure 3.2:

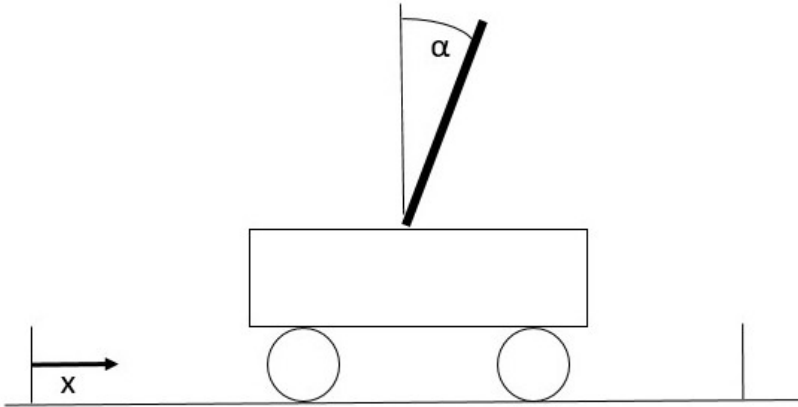


Figure 3.2. RL example task: cartpole

The policy of the task can be realized with the Q-Table in Table 3.1 which is initialized with zeros:

Table 3.1. Q-Table example

States	Left	Right
$S_1(x_1, \alpha_1, \dot{x}_1, \dot{\alpha}_1)$	0	0
...
$S_n(x_n, \alpha_n, \dot{x}_n, \dot{\alpha}_n)$	0	0

The values in Table 3.1 are the rewards of each action. Choosing the action based on the depending state can either be greedily by picking the action with the higher expected reward or exploratory by choosing the action randomly. The resulting reward is then updated in the table for the corresponding state-action pair. The more often the RL agent interacts with the environment the more values are stored in the table and the lesser actions are chosen randomly. The filled table is the learned value function [53].

The more complex the problem is the greater such a table gets which led to the implementation of other value functions. With DQL the Q-Table has been replaced by a neural network as the value function. According to Li [55] RL methods had been unstable and sometimes divergent before the deep Q-network (DQN) has been introduced. The DQL algorithm of Mnih et al. [56] brought the advantages through having an experience replay implemented where the agents experience is stored into a memory [57]. DQL updates are performed with randomly chosen samples from the memory. The advantage of having an experience replay rather than updating the DQN every consecutive step is that data can be used more efficiently since a single sample can be used for weight

updates multiple times and through choosing the samples randomly neglects the correlations between consecutive steps which reduces the variance between updates. Therefore, the probability of getting stuck in a local minimum or diverging is reduced by avoiding unwanted feedback loops. Another advantage is the implementation of the target network. Since the Q-network is a function approximator which is updated at each time step it does not only affect the next action rather than many further actions (in a Q-Table on the other hand the value of a state-action pair is assigned certainly – refer to Table 3.1). Therefore, the target network is updated (unlike the Q-network) periodically with the parameters of the Q-network. The weights of the target network are kept constant to avoid instability [56]. The DQL algorithm is illustrated with the following pseudo-code of Mnih et al. [56]:

Algorithm 3.1. DQL algorithm of Mnih et al. [56].

```


Initialize replay memory D with size N
Initialize action-value function Q with random weights w
Initialize target action-value function  $\hat{Q}$  with random weights  $w^- = w$ 
For episode = 1 to M do
    Initialize sequence  $s_1 = \{x_1\}$  and preprocessed sequence  $\phi_1 = \phi(s_1)$ 
    For t = 1 to T do
        Set an action  $a = \begin{cases} \text{random action} & \text{with probability } \epsilon \\ Q \text{ action value} & \text{otherwise} \end{cases}$ 
        Execute action and observe reward  $r_t$  and resulting state  $x_{t+1}$ 
        Set  $s_{t+1}$  and preprocess  $\phi_{t+1}$ 
        Store results in memory D
        Set discounted reward
        Perform gradient descent step on Q-network
        Every C step reset  $\hat{Q} = Q$ 
    End For
End For

```

Since the DQL example above is limited to a discrete action space Lillicrap et al. [58] enhanced the DQL algorithm to an actor-critic algorithm called deep deterministic policy gradient (DDPG). The actor function is the policy which maps states and actions and selects actions and the critic is the value function (as in DQL) and determines the Q-Values. The actor function is updated with respect to the determined Q-Value of the critic function [58]. The ability of the actor of choosing continuous actions enable actor-critic algorithms to solve a wider range of tasks e.g., the automated calibration of control units.

3.1.2 Deployment in the scope of the study

To solve combinatorial issues like the traveling salesman problem, RL algorithms are frequently employed [59], [60]. However, RL has also been applied to optimize the clutch engagement. Within simulations the parameters of a PID controller have been varied with a RL algorithm to minimize the jerk and the friction losses for the clutch engagement during launch by Xiaohui et al.



[61]. RL has also been applied by Gagliolo et al. [62] and Van Vaerenbergh et al. [63] to optimize the clutch engagement by minimizing the piston velocity and the engagement time of the clutch which led to promising results after a few hundred epochs [62], [63]. A similar optimization problem is investigated by Brys et al. [64] with focus on multi-objective aspects. Further the effects of scalarization on the performance is investigated. The study is carried out within simulations. Lampe et al. [65] applied a RL algorithm to optimize the quality of the clutch engagement while ensuring an immediate response. None of the studies focused on the optimization of the calibration parameters of the TCU. The calibration problem is not a combinatorial one which could be a reason. Also, the time spent to find a possible solution could be one fact not to use RL algorithms for solving such kind of problems since many optimization epochs are needed for the algorithm to converge, as illustrated in [62], [66].

3.2 Genetic Algorithms

Another option to achieve parameters by a self-learning system is the usage of a genetic algorithm (GA). These algorithms are inspired by the evolution species must experience in nature to populate further and avoid being extinct [67]. Further GAs are introduced to optimize problems with more than one and even conflicting optimization objectives [68].

3.2.1 Background and Motivation

A GA typically consists of the steps selection, mutation and recombination. To perform these steps a first population is usually initialized with arbitrary values [69]. Each population consists of several individuals. For each individual a fitness value is assigned based on the behavior the individual performs in the desired environment (how it fits to the optimization objectives) [70]. Hence, the fitness value is introduced to select the individuals for the further steps: mutation and recombination [71]. To understand the logic beyond a GA it is necessary to observe the process of genetic evolution in nature.

From a biological point of view, Herbert Spencer delivers a definition of genetic fitness. In his book: “A System of Synthetic Philosophy. The Principles of Biology.” Spencer describes fitness as the ability of an individual adapting to its environment. In detail, he describes that a species can vary compared to its parents of the former generation and the environment can change either. This variation leads to specific modifications of a species, which can be unexperienced. Thus, the environmental change can be fatal for some species, which does not have the ability to adapt to its new environment while other species are still able to live or even are more stable due to altering genetics. This leads to the well-known sentence “survival of the fittest” by Herbert Spencer [72]. The ability of adaption of a species to its environment called fitness is therefore the decisive argument for natural selection (the better the fitness the better the chance of a species to survive) [70]. By altering the genetic over several generations, the genetic of a species is turning to an equilibrium (stable state). When an equilibrium is reached, further alteration is not needed anymore as long as the environment does not change. This is called genetic drift [73].

To alter the genetic of a species, two different methods are common: recombination and mutation. Mutation describes a spontaneous altered genetic, which is inheritable to the next generations

[74]. Recombination is the process of sharing genetics of parents of a population to a child of the next generation [75]. Recombination comes with the advantage of fixing beneficial mutations and transferring them to further generations. Negative mutations respectively will not be transferred due to natural selection [76].

After mutation and recombination, new individuals are part of the following generation. For using a GA, the population size and the number of generations have to be defined [69].

A simplified GA is presented in [77] and [78] and could be presented with the following pseudo-code.

Algorithm 3.2. Pseudo-Code of a simple GA.

```
Set number of generations G
Set population size P
Set g = 0
Set p = 0
Initialize population of size P randomly
While g < G do
    Determine the fitness value of each chromosome
    While p < P do
        Select two parent chromosomes depending on the fitness values
        Define point where to split each chromosome
        Based on probability  $child = \begin{cases} \text{Combine} & \text{or} & \text{Copy} \\ \text{Mutate} & \text{the chromosomes} \end{cases}$  the chromosomes
        Add child to new generation
    End While
    If termination condition is met
        Break
    End If
End While
```

3.2.2 Deployment in the scope of the study

To solve complex tasks like the automated calibration of control units several GAs exist. The GA NSGA-II from Deb et al. [69] is a well-tested algorithm for solving such kinds of multi-objective optimization problems, and has been applied in a wide range of optimization problems [26], [39], [79], [80], [81], [82].

In order to achieve the best shift control for DCTs, Kahlbau [26], [81] used the NSGA-II algorithm for parameter optimization. As optimization objectives, the jerk and the derivative of the jerk are minimized to optimize the shift control. Additionally to the optimization objective of Kahlbau [26], [81] another optimization objective is introduced by Wehbi et al. [39] by measuring the time from start of the launch (acceleration pedal unequal zero) to synchronization (clutch slip speed equal zero) to optimize the driving behavior of the launch. The vehicle launch is optimized by Bachinger et al. [83] with the clutch closing time and a scaling factor for clutch slip speed as optimization

objectives using the differential evolution algorithm for vehicles with DCTs. By using a multi-objective evolutionary algorithm on a dynamic model, Huang [37] introduces the concept of model-based automated calibration. The algorithm approaches a desired trajectory that is controlled by a sliding mode controller to achieve the best match of comfort and sportiness criteria for the shift evaluation. In general, to improve the clutch engagement of a transmission to optimize the driving behavior GAs have been [84].

Also, to optimize the calibration parameters of the ECU GAs have been applied.

With a special DOE to identify an initial population and a novel GA based on existing GAs the calibration parameters of the ECU are optimized by Zaglauer [85] (only the method is knowledge-free the DoE is model-based). The determined initial population has the advantage of reducing the evaluation time. The engine is modeled using a gray-box model-based on a neural networks [85].

But GAs also have some drawbacks. The computation time and the quality of the evaluation is directly affected by the parameter population size (number of individuals per generation) [86]. There is also the possibility of remaining in a local minimum [87] and a long computation time [88]. Also, choosing the right parameters for crossover and mutation is influencing the quality of the solution of a GA [89].

3.3 Outcome

The drawbacks of RL and GA approaches in the optimization of the launch behavior through optimizing calibration parameters makes these approaches difficult to use in the regular development process of DCTs. To overcome these issues, the TSO algorithm has been introduced in [1] and [2]. The TSO algorithm is also knowledge-free and based on ML as a hybrid approach of RL and SL. The application of the TSO algorithm increases the efficiency of the calibration process within different test environments.

4 Optimization objectives

Typically, calibration parameters can be used to influence a vehicle and its component behavior. The software for the corresponding vehicle component includes these parameters, making it possible to adapt the software to various vehicle types without changing any software functions or the entire software. These parameters also provide the opportunity to modify the system behavior to fit customer requirements. To ensure a functioning vehicle with the highest possible quality and by fulfilling any requirement in the widest range of circumstances, the calibration parameters must be set iteratively under various environmental conditions. As a result, a lot of calibration parameters needs to be implemented into the software, which increases the calibration effort. These calibration parameters are implemented in the TCU software to affect the way a vehicle behaves. Experienced engineers optimize the driving behavior based on their subjective feelings with these parameters [26]. Due to the bias introduced by personal preferences, the evaluation of the driving behavior can therefore differ between engineers. This might result in additional adjustment loops and a greater calibration effort [90]. Therefore, the automated optimization of the drivability is strived to be implemented in the development process in order to lessen the workload of an engineer. To enable the optimization algorithms to monitor the quality of driving behavior without the involvement of an engineer, subjective feelings must be converted into objective measurements [39]. The research in [1] and [2] and the present study itself uses four different launch behavior objectives to measure the transfer of these subjective feelings. The objectives are: (1) the acceleration objective, (2) the reaction time objective, (3) the engine speed objective, and (4) the clutch torque objective.

These objectives are examined within a test subject study which is carried out and documented by He et al. [3]. The test subject study took place in a driving simulator before being applied in the automated optimization. The use of a driving simulator makes it possible to conduct studies on test subjects and examine the influence of the engine noise, maximum acceleration, acceleration build-up (mean jerk: first-time derivative of the acceleration) and reaction time on the evaluation of the vehicle launch. Based on these values the vehicle launch can be assessed regarding its sportiness, comfort and jerkiness [3].

The driving simulator has the ability to simulate the dynamics of the vehicle's longitudinal direction and was created for the study of human perception during longitudinal drive maneuvers. By combining translatory and rotatory motions of the driver cabin, it can simulate acceleration and deceleration [91]. By using virtual reality technology, simulating the driving environment (such as the roads, traffic signs, and landscape), and synthesizing the engine sound, the test subjects experience a driving situation that is close to reality. In addition to the simulation of the auditory and visual senses, the haptic sense is also taken into account [78].

The optimization objectives are divided into two categories: customer objectives (Section 4.1) and discomfort objectives (Section 4.2). These objectives are evaluated within a Python script which is connected to the optimization algorithms.

4.1 Customer Objectives

Since customer requirements form the basis of the customer objectives $obj_{customer}$, a measured value (v_{meas}) is compared with a reference value (v_{ref}) that is defined by the customer. The absolute deviation is normalized to ensure that every customer objective is in a similar range:

$$obj_{customer} = \left| \frac{v_{meas} - v_{ref}}{v_{ref}} \right| \quad (4.1)$$

In order to equalize the weight of the objectives and make them comparable, the normalization is used, which opens up the possibility of setting a relative tolerance. This objective value is set to zero if it is less than 0.1. The result is the introduction of a 10% tolerance based on the reference value:

$$v_{ref} \cdot 0.9 \leq v_{meas} \leq v_{ref} \cdot 1.1 \quad (4.2)$$

As a result, the following equation is used to determine the customer objectives of Sections 4.1.1 and 4.1.2:

$$obj_{customer,Tolerance} = \begin{cases} 0 & v_{ref} \cdot 0.9 \leq v_{meas} \leq v_{ref} \cdot 1.1 \\ \left| \frac{v_{meas} - v_{ref}}{v_{ref}} \right| & otherwise \end{cases} \quad (4.3)$$

4.1.1 Acceleration Peak and Acceleration Build-Up Objective

While speed cannot be perceived by humans, acceleration and jerk can [92]. Since the purpose of a vehicle launch is to increase speed from standstill, the driver's perception is affected by the acceleration and the acceleration build-up during launch. The test subject study of He et al. [3] varies the maximum acceleration and the acceleration build-up in order to investigate the impact on the evaluation criteria of sportiness, comfort, and jerkiness. Figure 4.1 illustrates an exemplary acceleration profile during a vehicle launch.

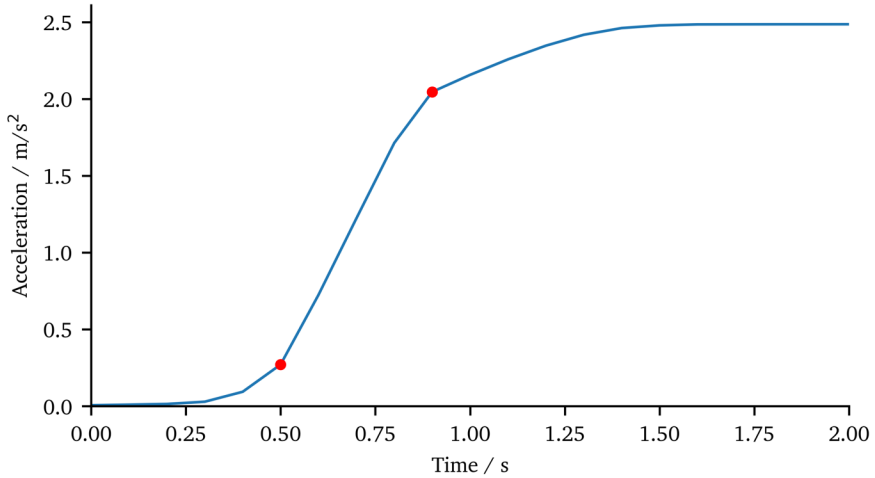


Figure 4.1. Acceleration profile of a launch [3].

In the test subject study [3] the acceleration build-up only refers to the mean jerk during the acceleration's rise. The transitions from standstill to a steadily increasing acceleration and then the transition to the maximal acceleration are not included in the calculation of the mean jerk. It is calculated from 15% and ends at 85% of the maximal acceleration according to [90]. Figure 4.1 highlights the beginning and ending locations in red. In the study, contrary effects of the acceleration build-up on comfort and jerkiness can be observed. The launches are evaluated jerkier and less comfortable the higher the maximum acceleration and the acceleration build-up are [3].

Further, conflicts exist between the demands for a sporty and a comfortable launch [26]. The comfort of the passengers is improved by reducing the maximum acceleration and the acceleration build-up [93]. In contrast, rapid accelerations and mean jerks (with shorter build-up times) tend to be more sportier [39]. The maximum acceleration and mean jerk must both be greater than zero in order to increase a vehicle's speed, but they must also not be too high to avoid discomfort. The measured values maximal acceleration a_{max} and the acceleration build-up \hat{a}_{build} are compared with its corresponding reference values ($a_{max,ref}$ and $\hat{a}_{build,ref}$) which should be chosen with regard on the subject study of [3]. So, using the equation below, the maximal acceleration objective $obj_{max,acc}$ is calculated:

$$obj_{max,acc} = \begin{cases} 0 & a_{max,ref} \cdot 0.9 \leq a_{max} \leq a_{max,ref} \cdot 1.1 \\ \left| \frac{a_{max} - a_{max,ref}}{a_{max,ref}} \right| & otherwise \end{cases} \quad (4.4)$$

And thus, using the equation below, the acceleration build-up objective $obj_{build,acc}$ is determined:

$$obj_{build,acc} = \begin{cases} 0 & \dot{a}_{build,ref} \cdot 0.9 \leq \dot{a}_{build} \leq \dot{a}_{build,ref} \cdot 1.1 \\ \left| \frac{\dot{a}_{build} - \dot{a}_{build,ref}}{\dot{a}_{build,ref}} \right| & otherwise \end{cases} \quad (4.5)$$

As stated in Section 4.1, if the measured values fall within a 10% range of the corresponding reference values, the objectives are zero. Since these reference values are defined by the customer both objectives are customer objectives. Nevertheless, the reference values should be chosen with respect to the study by He et al. [3].

4.1.2 Reaction Time Objective

According to Simon [90], a noticeable acceleration during launch occurs if 15% of the maximum acceleration is reached. So, the reaction time is the time between the driver actuating the accelerator pedal and reaching the noticeable acceleration.

He also investigated in a study similar to [3] the influence of the reaction time on the criteria of sportiness and comfort and published the results in Schmiedt et al. [2]. In the study the reaction time is varied with the three values 250, 550, and 850 ms. The reaction time is applied to different positions of the accelerator pedal since the driver is expecting different vehicle reactions depending on this position. Therefore, the accelerator pedal position is also varied with the three values 20%, 40% and 60% during these tests. The study of He concluded that the reaction time has a significant influence on the perception of the vehicle launch regarding its sportiness but less on the comfort criterion. From the test subject study, it can be derived that at low accelerator pedal positions the driver is expecting higher reaction times. Contrary the expectation for lower reaction times are increasing with increased accelerator pedal position since the driver is expecting a sportier behavior [2]. In order to determine the reaction time objective obj_{rea} , the results of this study can be used to set the reference value $t_{rea,ref}$ (depending on the position of the accelerator pedal). According to the study the reference value $t_{rea,ref}$ for low accelerator pedal positions can be set higher than for high accelerator pedal position values. The reference value $t_{rea,ref}$ is compared to the measured reaction time $t_{rea,meas}$:

$$obj_{rea} = \begin{cases} 0 & t_{rea,ref} \cdot 0.9 \leq t_{rea,meas} \leq t_{rea,ref} \cdot 1.1 \\ \left| \frac{t_{rea,meas} - t_{rea,ref}}{t_{rea,ref}} \right| & otherwise \end{cases} \quad (4.6)$$

The reaction time objective is a customer objective as well as the acceleration and acceleration build-up objectives of Section 4.1.1.

4.2 Discomfort Objectives

The discomfort objectives differ from the customer objectives of Section 4.1 since they are not based on a reference value. Instead, the objectives must be zero to achieve the best results. Any value greater than zero has a negative impact on a driving behavior. In Sections 4.2.1 and 4.2.2, the discomfort objectives are described.

4.2.1 Engine Speed Objective

The driver's perception of the launch is influenced by the engine speed behavior, which has an impact on the acoustics. Therefore, the driving comfort during the vehicle launch is directly impacted by a dropping engine speed (Figure 4.2b). By varying the engine speed drops n_{drop} between 0, 250, 500, and 750 rpm during a test subject study, the impact on the driver's perception is assessed. The test subject study is again carried out in the driving simulator by He similarly to [3] and the results are presented in Schmiedt et al. [2]. The subject study concludes that the launch behavior can be improved by ensuring that the engine speed is solely increasing [94] (Figure 4.2a) or, at the least, not dropping by more than 250 rpm. As stated in Chapter 2.6 the clutch torque during clutch slip phases can influence the engine speed behavior [5]. Decreasing the engine speed is only possible by increasing the clutch torque over the engine torque or by requesting an engine intervention. Since increasing the clutch torque can cause discomfort [37] and requesting an engine intervention can increase pollutant emissions [95], both options for reducing the engine speed should be avoided during the vehicle launch. Therefore, achieving an engine speed that is solely increasing is further beneficial to ensure a comfortable launch while meeting legislative requirements and not only to improve the acoustic behavior. The optimization objective obj_n is formed with these requirements:

$$obj_n = \frac{n_{drop}}{1000 \text{ rpm}} \quad (4.7)$$

In order to maintain the objective value within a similar range as the customer objectives (see Section 4.1), the objective is normalized. Also, a tolerance is implemented for this objective since He et al. [3] investigated that an engine speed drop of less than 250 rpm is not critical. However, a stricter threshold of 100 rpm is desired for the evaluation. Therefore, the objective is normalized with 1000 rpm resulting in a tolerance value of 0.1 comparable to the customer's objectives.

$$obj_n = \begin{cases} 0 & \frac{n_{drop}}{1000 \text{ rpm}} \leq 0.1 \\ \frac{n_{drop}}{1000 \text{ rpm}} & \text{otherwise} \end{cases} \quad (4.8)$$

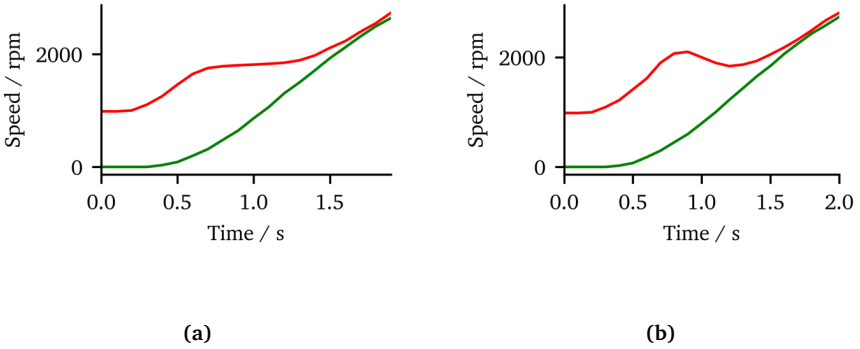


Figure 4.2. **(a)** Engine speed (red) does not drop, **(b)** Engine speed drops; Input-shaft speed is green.

The behavior of the engine speed is shown in Figure 4.2. The left measurement (a) is measured with a vehicle, the right one (b) is modified manually to illustrate the behavior.

4.2.2 Clutch Torque/Jerk Objective

Humans are even more sensitive to high jerks (first derivative of the acceleration) than to acceleration. High accelerations are uncomfortable, but a vehicle's change in acceleration is also related to passenger comfort [96]. During vehicle tests it turned out that the acceleration build-up of Equation (4.5) is suitable for the SiL environment but in the vehicle a bumpy feeling could occur anyhow despite fulfilling every objective. Therefore, the clutch torque objective is introduced for vehicle tests to identify changes in the acceleration.

Since the clutch torque (T_{clu}) is directly transmitted to the wheels during slip phases, the longitudinal acceleration of the vehicle is directly related to the clutch torque (see Figure 2.28). Due to the discomfort caused by inconsistencies during clutch engagement, it is mandatory to take care of the clutch torque gradient when controlling the clutch [37]. In order to determine whether there is a drop in the acceleration signal during launch, the clutch torque is analyzed during launch for local minima (Figure 4.4). If the first derivative of the clutch torque \dot{T}_{clu} is zero and its second derivative \ddot{T}_{clu} is greater than zero, a local minimum is discovered. To detect an uncomfortable driving behavior, the number of local minima obj_T is counted and should ideally be zero:

$$obj_T = \sum_{i=1}^n [\dot{T}_{clu}(t) = 0 \wedge \ddot{T}_{clu}(t) > 0]_i \quad (4.9)$$

Since the measured signal is sample-based, it is often not possible to find an exact zero of the

derivatives, which makes a numerical analysis difficult. Instead, Algorithm 4.1 is used to investigate the torque for local minima.

Algorithm 4.1. Local minima detection of the clutch torque.

```
Set gradient = gradient(signal)
Declare ExtremePoints
Set q = 0
Set i = 1

While i < length(gradient) do
  If (gradient[i] < 0 && gradient[i - 1] ≥ 0) || (gradient[i] ≥ 0 && gradient[i - 1] < 0) then
    Set ExtremePoints[q] = i
    Set q++
  End If
End While

Declare LowPoints
Set l = 0

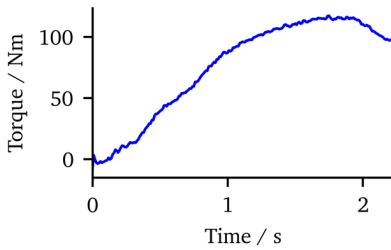
For Each e in ExtremePoints do
  If signal[e - 1] > signal[e] then
    Set LowPoints[l] = e
    Set l++
  End If
End For

Set objT = length(LowPoints)

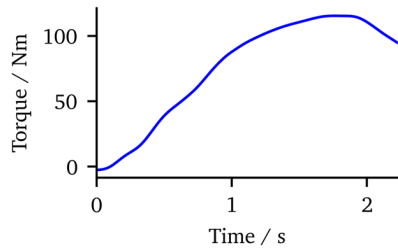
Return objT
```

If a sign change is detected in the derivative of the torque (*ExtremePoints*) an extremum is detected. An extremum is a local minimum if the value is lower than the value of the previously discovered extrema. The value of the objective is equal to the quantity of local minima.

The measured clutch torque in a real vehicle is a noisy signal with multiple local minima, making it challenging to apply the algorithm above, even though it is theoretically correct. To lessen the noise, the signal is therefore filtered using a one-dimensional gaussian filter [97]. Figure 4.3 illustrates the impact of the filtering.



(a)



(b)

Figure 4.3. Influence of the filtering on the signal **(a)** unfiltered, **(b)** filtered.

Due to the unfiltered signal's noisy behavior, there are 25 local minima (counted with Algorithm 4.1) on the left in Figure 4.3a which indicates an uncomfortable driving behavior. However, a behavior like this with small amplitudes is not noticeable and does not have a negative impact on the evaluation. Algorithm 4.1 on the filtered signal, on the other hand, predicts a driving behavior without discomfort because no local minimum is found (Figure 4.3b). In contrast in Figure 4.4, a typical clutch torque behavior that causes discomfort is illustrated.

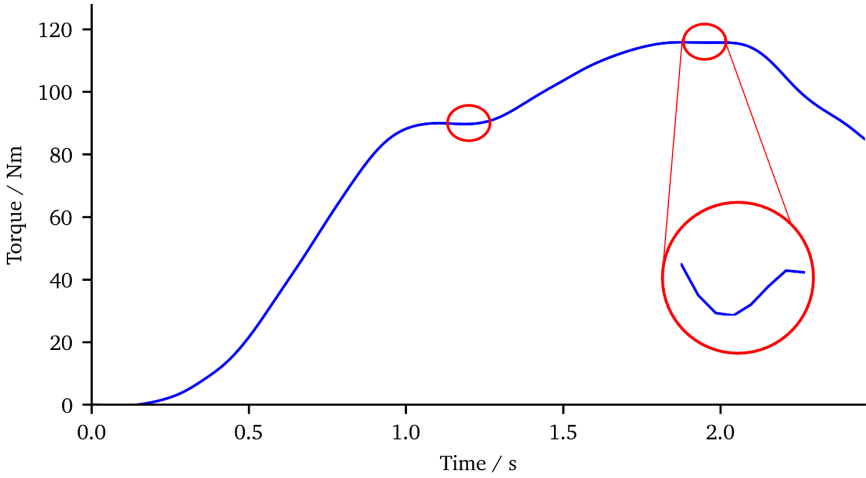


Figure 4.4. Clutch torque with one local minimum.

The red marker indicates the local minima of the clutch torque which would be noticed as discomfort. The second local minimum is highlighted to visualize the change of the gradient.

Unlike the other objectives, obj_T is not normalized so if an inconsistency occurs, the objective value is equal to or greater than one. Therefore, the possibility of a successful launch in the event of an objective value deviating from zero is eliminated.

4.3 The Reward

As aforementioned, the reward in RL algorithms is introduced to measure the quality of an action. The algorithm aims to maximize the reward based on the state of the environment. The reward in this case is the negated sum of the objectives. A launch is successful if the state is equal to the target state. If this happens all objectives are zero and hence the reward is zero too. This would be the highest (and best) value the reward can reach. The reward within the software in the loop (SiL) environment and thus the quality of a calibration is determined as follows:

$$reward = -(obj_n + obj_{max,acc} + obj_{build,acc} + obj_{rea}) \quad (4.10)$$

According to Table 4.1, the target state for any optimization within the SiL environment has been set with regard to the study by He et al. [3] as follows:

Table 4.1. Target state for the SiL environment.

Objective	Value	Unit
Engine Speed Drop (n_{drop})	0.00	rpm
Acceleration Peak ($a_{max,ref}$)	3.25	m/s ²
Acceleration Build-up ($\hat{a}_{build,ref}$)	6.50	m/s ³
Reaction Time ($t_{rea,ref}$)	0.50	s

Every benchmark optimization algorithm of Section 7.2 has been tested using this target state, which has been reproducibly reached in simulations.

The application of the algorithms within a test vehicle was initially carried out with the same target state as that shown in Table 4.1.

From a practical point of view, it turned out that some launches that met the requirement felt bumpy. Also, some launches that had an acceleration build-up that was different from the reference value could have also produced subjectively satisfying launches. As a result, the discomfort objective of Section 4.2.2 has replaced the acceleration build-up objective of Section 4.1.1 to prevent jerky launches during optimization. Therefore, the reward function for the test vehicle changes to:

$$reward = -(obj_n + obj_{max,acc} + obj_T + obj_{rea}) \quad (4.11)$$

The summed-up reward is in addition also used as an assessment of the individuals of the GA and the actions of the TSO algorithm.

5 Test environments

To optimize the driving behavior according to the objectives of Chapter 4 there are different test environments available.

5.1 Test Vehicle

Usually, the calibration of any control unit takes place in test vehicles. This kind of vehicles can be equipped with engines, transmissions or infotainment systems which are not necessarily production ready but used for developing the hardware or software solutions. With these vehicles e.g., the control software of embedded systems is adjusted during test trips. Since the vehicles and its parts during development phase are prototypes the costs to produce the test vehicles are high and an efficient usage is required.

To optimize the launch behavior the embedded software solution (illustrated in Section 2.5.2) is implemented on the TCU. The optimization of the launch behavior at sea level, without slope and in first gear is primarily driven through five calibration parameters. The driver's requested torque, which is determined by the position of the accelerator pedal, usually affect (see Table 5.1) these parameters. The calibration parameters affect the different phases of the vehicle launch of Chapter 2.6 which are shown in Figure 2.26:

- Parameter 1–phase 1: To increase the engine speed to phase 2, a certain percentage of the engine torque should be applied (a low value results in a quick vehicle reaction, whereas high value indicates a rapid increase in engine speed but with a worsen vehicle reaction).
- Parameter 2–phase 1: Minimum engine torque that should be applied to speed up the engine to phase 2 (active if the value of parameter 1 is too low).
- Parameter 3: To control how the engine speed behaves in relation to the engine target speed, the P-gain control value is used.
- Parameter 4–phase 3: parameter setting the time until the slip speed should be 0.
- Parameter 5–phase 3: parameter setting the engine speed which should be reached at the end of phase 3.

These parameters are, as previously mentioned, dependent from one or more input signals e.g., the driver request torque. With the input signals the parameters can be interpreted as a one-dimensional curve or a two-dimensional map. The latter example of a map is illustrated in Table 5.1.

Table 5.1. Example of a calibration map–Parameter 3: P-gain.

P-gain	Engine Speed Error / rpm					
	-500	-250	0	250	500	
Driver request / Nm	0	Z ₁₁	Z ₁₂	Z ₁₃	Z ₁₄	Z ₁₅
	150	Z ₂₁	Z ₂₂	Z ₂₃	Z ₂₄	Z ₂₅
	300	Z ₃₁	Z ₃₂	Z ₃₃	Z ₃₄	Z ₃₅

The difference between the engine target speed and the engine speed that is actually measured is the engine speed error. The value of the P-gain can be varied for various engine speed errors, as shown in Table 5.1 (the values are interpolated between the nodes).

During the optimization in this study only the values of the nodes that have an impact on the driving behavior are changed. For example, if the driver request is 150 Nm, altering values of the nodes of 0 Nm or 300 Nm does not affect the driving behavior hence these values are not considered during optimization. The marked row of Table 5.1 is illustrating this exemplary. Only the marked row needs to be adjusted to best suit the desired driving maneuver. Therefore, 22 values that need to be changed during optimization are derived from these five calibration parameters.

Since the calibration of a component requires access to the corresponding control unit, the optimization algorithms as well as the evaluation algorithms communicate with the programming interface of the common calibration software INCA of the company ETAS. Therefore, the calibration parameters are adjusted by the optimization algorithms and after the measurement is finished the required signals are evaluated regarding the objectives of Section 4.

5.2 Software in the Loop (SiL) Environment

In a SiL environment, as mentioned in Section 4.3, the different algorithms are tested. The software solution deployed in the test vehicle (implied in Section 5.1) is equal to the one in the SiL environment. Therefore, software functions can be tested without the use of expensive prototypes. Testing optimization algorithms in a virtual vehicle has the advantage of reproducible results, free from environmental influences. The tool Silver, a Synopsys product, is a virtual ECU platform used to configure the SiL environment and has a programming interface which can be accessed with Python [98], [99]. Within a SiL environment it is possible to develop some tasks without the use of hardware. Since the SiL environment is entirely modeled beside the TCU software also the entire powertrain (engine, transmission, etc.) is modeled. As a torque source the ICE model is used which also, takes auxiliary torques into account. In order to come as close to a digital twin of a real vehicle as possible, the hardware models of the vehicle itself for calculating the driving resistances and the transmission model with its inertias are also implemented. The models cannot be fully illustrated due to confidential reasons.

The SiL environment is not well suited for calibration tasks because the system behavior of the powertrain model differs from a real vehicle. To illustrate the clutch torque behavior of both test environments Figure 5.1 is introduced.

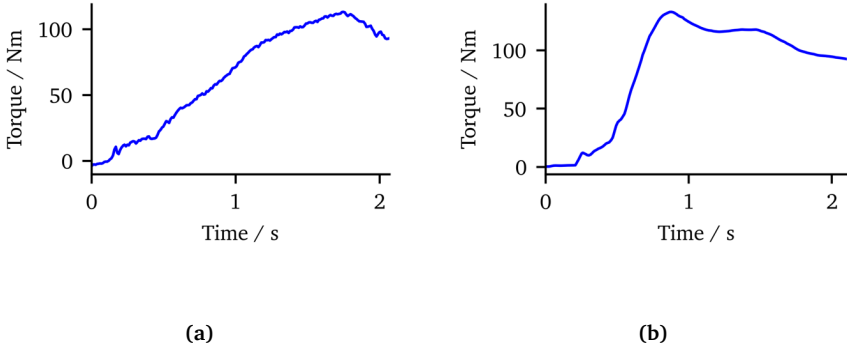


Figure 5.1. Clutch torque behavior of **(a)** a real vehicle and **(b)** the SiL environment.

It is observable that the behavior of the clutch torque is different between the two test environments. The maximum clutch torque of the real vehicle **(a)** occurs at about second 1.75 while the maximum acceleration within the SiL environment is measured earlier. Therefore, also the gradient of the clutch torque until the maximum acceleration is reached is much steeper in the SiL environment. Since as explained the clutch torque and the engine torque of the environments are different this also affects the behavior of the engine speed and the input-shaft which is illustrated in Figure 5.2.

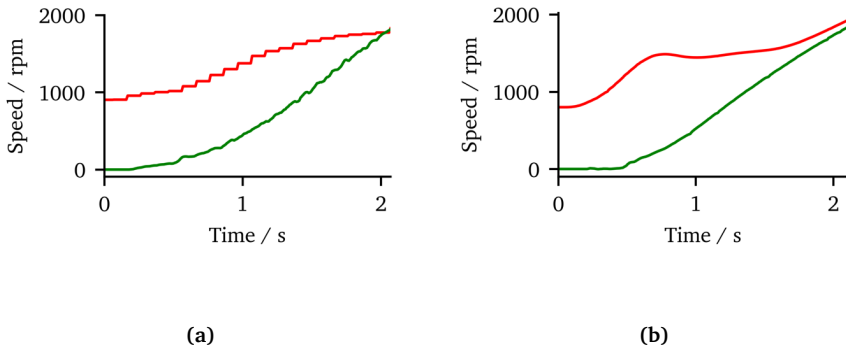


Figure 5.2. Engine speed (red) and input-shaft speed behavior of **(a)** a real vehicle and **(b)** the SiL environment.

The input-shaft speed of the SiL environment **(b)** is remaining longer at 0 rpm compared to the input-shaft speed of the real vehicle **(a)** which indicates that the reaction time is probably worse in the SiL environment compared with the real vehicle. Although the calibration of the environments was different, the example of Figure 5.1 and Figure 5.2 illustrates the fact that the underlying vehicle models of the SiL environment are hardly to be adjusted to fit the behavior of the test vehicle. Therefore, the calibration is not transferable between the vehicle and its digital twin.

Nevertheless, the SiL environment has the advantage of having the ability to produce repeatable results. This enables the possibility to test various calibration techniques for automating the calibration process effectively. All tested algorithms of Section 7.2 are tested within the SiL environment. Those algorithms which are most promising are further tested in a test vehicle. The results are illustrated in Section 7.3. All of the tests are run/simulated at sea level, at 20 °C, and with zero slope.

5.3 Simplified model

Since the embedded TCU software cannot be presented due to confidential reasons a simplified model is set up with Python. The model is introduced to illustrate the embedded software of the TCU and the influence of calibration parameters (described in Section 5.1). The model contains the engine control, the vehicle control and the transmission control shown in Figure 5.3. The powertrain parameters which affect the vehicles behavior are defined in Table 2.1.

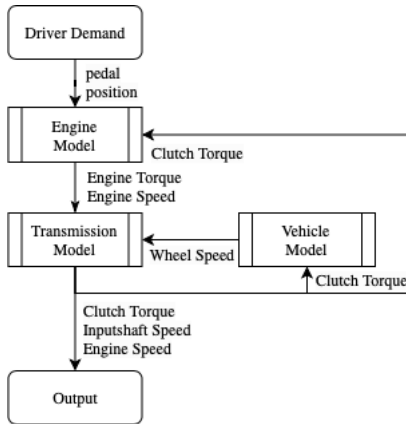


Figure 5.3. Simplified vehicle model.

5.3.1 Simplified Engine Model

The simplified engine model is basically a model for generating a torque signal as an input for the transmission model. Further also the engine specific constants are defined. These are the engine inertia and the engine idle speed. According to Bulatovi et al. [6] the engine inertia can range depending on the engine geometry between 0.168 kgm^2 and 1.012 kgm^2 [6]. For the exemplary simulation an engine inertia of 0.3 kgm^2 is chosen (see Section 2.2.3). The engine idle speed is set to 850 rpm.

Since the model is a fictive one the engine torque behavior is not modeled accurately. To approach the behavior a factor (fac_T) is introduced (starting with 1) which is decreased at each step.

$$fac_T(i) = 0.98 \cdot fac_T(i - 1) \quad (5.1)$$

The decreasing factor of 0.98 is only an estimation for approaching the fictive engine torque to a behavior similar to Figure 2.5 or the clutch torque during launch of Figure 4.3. With the torque factor (fac_T) the engine torque (T_e) is determined by multiplying it with the torque request (T_{req}) (which is set to 100 Nm).

$$T_e = (1 - fac_T) \cdot T_{req} \quad (5.2)$$

The resulting engine torque is illustrated in Figure 5.4.

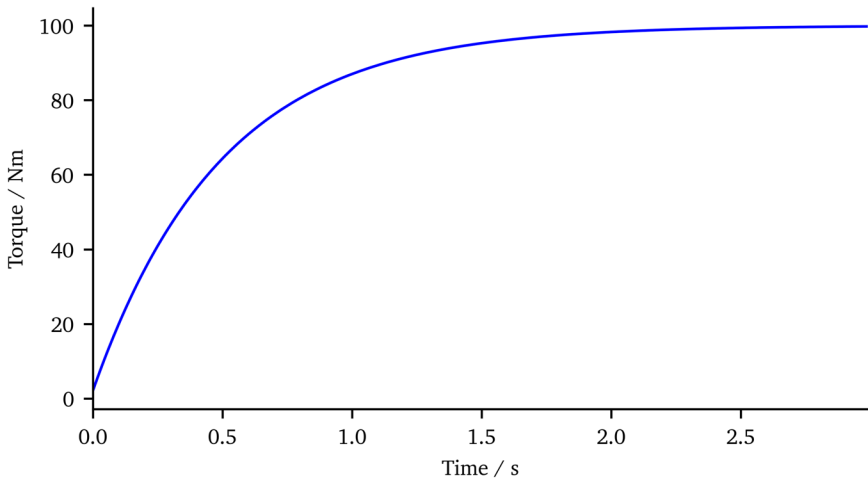


Figure 5.4. Engine torque of the simplified engine model.

With the difference of the engine torque (T_e) and the clutch torque (T_{clu}) as well as with the engine inertia J_e the gradient of the engine speed (\dot{n}_e) is determined.

$$\dot{n}_e = \frac{(T_e - T_{clu})}{J_e} \quad (5.3)$$

With the gradient of the engine speed the engine speed can be determined by integrating the gradient over time with the engine idle speed as the starting condition. The pseudocode of the simplified engine model is illustrated in Algorithm 5.1. As stated earlier the powertrain parameters e.g. the engine inertia can be found in Table 2.1.

Algorithm 5.1. Simplified Engine Model.

Declare EngineInertia = 0.3
Declare EngineSpeedIdle = 850
Declare TorqueRequest = 100
Declare TorqueFactor = 1

Function EngineStep

Pass In Variables: TorqueClutch, time

Calculate TorqueFactor = 0.98 * TorqueFactor

Calculate TorqueEngine(t) = (1 - TorqueFactor) * TorqueRequest

Calculate EngineSpeedGradient = (TorqueEngine(t) - TorqueClutch(t)) / EngineInertia

Calculate SpeedEngine(t) = **Integrate** EngineSpeedGradient **over** time **with** EngineSpeedIdle **as Starting Condition**

Return Variables SpeedEngine, TorqueEngine

Return Constants EngineSpeedIdle, EngineInertia

End Function

The engine torque and speed are further used in the transmission model.

5.3.2 Simplified Vehicle Model

Beside the engine model also a vehicle model is required to illustrate the calibration of the transmission. In the vehicle model the constants of Section 2.2.3 are defined. These constants are the total ratio of the first gear of the transmission and the differential transmission (i – estimated with 16), the efficiency of the powertrain (η_{pwt} – estimated with 0.98), the radius of the wheel (r_{dyn} – estimated with 0.3 m) and the mass of the vehicle (m_v – estimated with 1.800 kg).

With these constants and the clutch torque, the torque at the wheels (T_w), the corresponding force (F_w) and the acceleration (a) can be determined.

$$T_w = T_{clu} \cdot i \cdot \eta_{pwt} \quad (5.4)$$

$$F_w = T_w \cdot r_{dyn} \quad (5.5)$$

$$a = \frac{F_w}{m_v} \quad (5.6)$$

With the acceleration the velocity is determined through integrating the acceleration over time.

The vehicle velocity is further used to determine the wheel speed. This simplified vehicle model is illustrated in Algorithm 5.2.

Algorithm 5.2. Simplified Vehicle Model.

Declare RatioTotal = 16
Declare EfficiencyPowertrain = 0.98
Declare RadiusWheel = 0.3
Declare MassVehicle = 1800

Function VehicleStep

Pass In Variables: TorqueClutch, time

Calculate TorqueWheel = TorqueClutch(t) * RatioTotal * PowertrainEfficiency
Calculate ForceWheel = TorqueWheel / RadiusWheel
Calculate Acceleration = ForceWheel / MassVehicle
Calculate Velocity = **Integrate** Acceleration **over** time
Calculate SpeedWheel(t) = (Velocity * 2 * Pi) / RadiusWheel

Return Variables SpeedWheel

Return Constants RatioTotal

End Function

5.3.3 Simplified Transmission Model

The simplified transmission model requires the input of the vehicle and the engine model to determine its torques and speeds. With the total ratio and the wheel speed the input-shaft speed is determined. As mentioned prior the clutch torque is influenced by the desired engine speed which is shaped by the calibration parameters.

The simplified transmission model has two calibration parameters which are similar (SpeedEngineGradient) or equal (P-Gain) to the illustrated parameters of Section 5.1:

- SpeedEngineGradient: increases the desired engine speed at each time step until the SpeedDesiredLaunch has been reached.
- P-Gain: defines the deviation of the clutch torque and the engine torque to control the engine speed. The P-Gain is dependent on the engine speed error (deviation of the desired engine speed and the engine speed).

Further the desired engine speed is also dependent on the input-shaft-speed. After the input-shaft-speed has increased close to the desired engine speed the desired engine speed is increasing with the input-shaft-speed (but with an offset – green area) to ensure that the engine speed is greater than the input shaft speed. The gap between the desired engine speed and the input-shaft-speed

is dependent on the desired slip of 10 rpm (between the engine speed and the input-shaft-speed) as well as the calibrated P-Gain value (observable in Algorithm 5.3). This enables a smooth engagement of the clutch. The desired engine speed is illustrated in Figure 5.5.

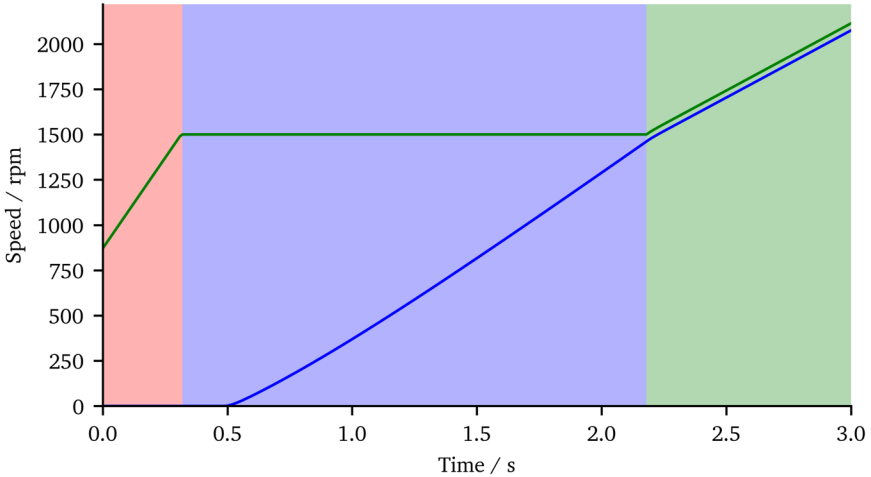


Figure 5.5. Desired engine speed of the simplified model.

It is observable that the desired engine speed is increasing at each time step from the engine idle speed until the desired launch speed is reached (red area). The desired engine speed remains at this level (blue area) until the input-shaft-speed (including the offset) exceeds the desired launch speed and becomes the desired engine speed itself (green area).

With the P-Gain, the engine inertia and the engine speed error the torque is determined which is required to increase the engine speed to the desired engine speed. This torque is subtracted from the engine torque to determine the clutch torque. Therefore, the desired engine speed has an influence on the clutch torque.

The determination of the clutch torque and the desired engine speed is illustrated in Algorithm 5.3.

Algorithm 5.3. Simplified Transmission Model.

Declare SpeedDesiredLaunch = 1500

Declare SpeedDesiredSlip = 10

Function TransmissionStep

Pass In Variables: SpeedWheel, SpeedEngine, TorqueEngine, time as t

Pass In Constants: RatioTotal, EngineSpeedIdle, EngineInertia

Pass In Parameters: SpeedEngineGradient, PGain

Calculate SpeedInputShaft(t) = SpeedWheel(t) * RatioTotal

Calculate SpeedSlip = SpeedEngine – SpeedInputShaft(t)

Calculate SpeedDesiredMinimum = max(SpeedEngineDesired(t-1), EngineSpeedIdle)

Calculate EngineSpeedDesiredRise = SpeedDesiredMinimum + SpeedEngineGradient

Calculate SpeedDesiredLaunch = min(SpeedDesiredLaunch, EngineSpeedDesiredRise)

Calculate SpeedEngineEngage = SpeedInputShaft(t) + SpeedDesiredSlip

Calculate SpeedEngineDesired(t) = max(SpeedDesiredLaunch, SpeedEngineEngage)

Calculate SpeedError = max(SpeedEngine(t), EngineSpeedIdle) – SpeedEngineDesired(t)

If SpeedSlip < 100 / PGain(SpeedError) **Then**

Calculate SlipDifference = SpeedDesiredSlip – SpeedSlip

 SpeedEngineDesired(t) = SpeedEngineDesired(t) + SlipDifference / PGain(SpeedError)

End If

If SpeedEngineDesired(t) < SpeedEngineDesired(t-1) **Then**

Calculate SpeedInputShaftGradient = SpeedInputShaft(t) – SpeedInputShaft(t-1)

 SpeedEngineDesired(t) = SpeedEngineDesired(t-1) + SpeedInputShaftGradient

End If

Calculate SpeedError = max(SpeedEngine(t), EngineSpeedIdle) – SpeedEngineDesired(t)

Calculate EngineSpeedControlTorque = EngineInertia * PGain(SpeedError) * SpeedError

Calculate TorqueClutch(t) = max(TorqueEngine(t) – EngineSpeedControlTorque, 0)

Return TorqueClutch

End Function

5.3.4 Main Loop

Each of the functions of Section 5.3.1 to 5.3.3 are called at each time step. To perform a vehicle, launch with the simplified model the main loop has to be triggered which is illustrated in Algorithm 5.4.

Algorithm 5.4. Main Loop of the Simplified Model.

```
Initialize time = 0
Declare EndTime = 3
Declare TimeStep = 0.01

While time < EndTime then
  Call EngineStep pass ClutchTorque, time
  Call Vehicle Step pass ClutchTorque, time
  Call TransmissionStep pass SpeedWheel, SpeedEngine, TorqueEngine, time

  Set time = time + TimeStep
End While
```

5.3.5 Influence of the Calibration Parameters

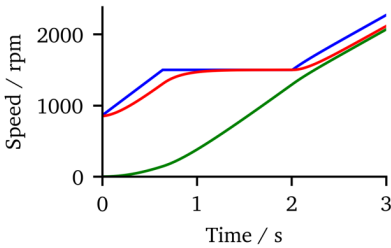
The SpeedEngineGradient is a scalar value which determines the gradient of the desired engine speed. In contrast the P-Gain is a curve similarly to Table 5.1 which is as mentioned dependent on the engine speed error. The P-Gain is illustrated in Table 5.2.

Table 5.2. P-Gain of the simplified model

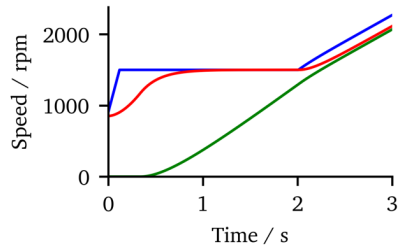
Engine Speed Error / rpm	-500	-250	0	250	500
P-Gain	Z11	Z12	Z13	Z14	Z15

Since the calibration parameters influence the engine speed as well as the clutch torque there is also an influence regarding the driving behavior (see Chapter 4) e.g., by influencing the reaction time.

Varying the calibration parameter “SpeedEngineGradient” results in the speeds of Figure 5.6:



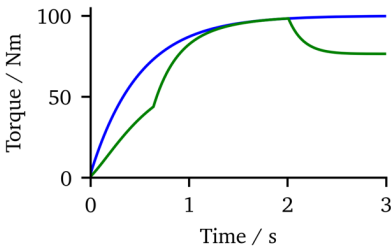
(a)



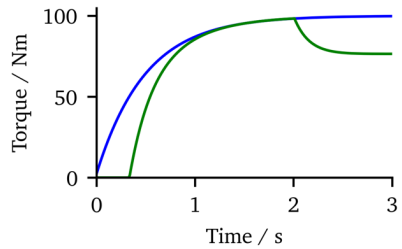
(b)

Figure 5.6. Influence of the "SpeedEngineGradient"; **(a/b)**: desired engine speed (blue), engine speed (red), input-shaft speed (green).

The corresponding torque is illustrated in Figure 5.7.



(a)

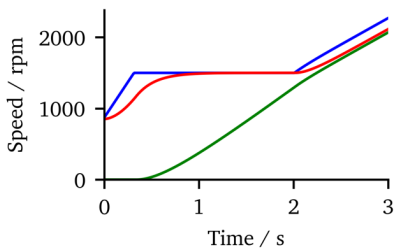


(b)

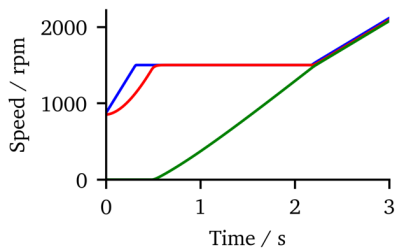
Figure 5.7. Influence of the "SpeedEngineGradient"; **(a/b)**: engine torque (blue), clutch torque (green).

The P-gain in both situations is set to 0.5 while the "SpeedEngineGradient" is varied in Figure 5.6 and Figure 5.7. On the left side (Figure 5.6a / Figure 5.7a) the calibration parameter is set to 10 while it is set to 50 on the right side (Figure 5.6b / Figure 5.7b). It is observable that the gradient of the desired engine speed is steeper on the right which results in a worsened response (the clutch torque is rising delayed in (Figure 5.7b) compared to (Figure 5.6b)) but the target speed of 1500 rpm is reached faster compared to (Figure 5.6a).

Figure 5.8 illustrates the influence of the P-Gain on the speeds.



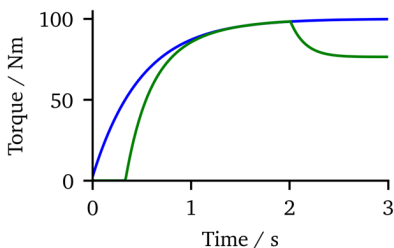
(a)



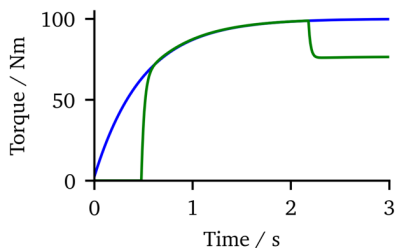
(b)

Figure 5.8. Influence of the P-gain; **(a/b)**: desired engine speed (blue), engine speed (red), input-shaft speed (green).

Again, the corresponding torque is illustrated in Figure 5.9.



(a)



(b)

Figure 5.9. Influence of the P-gain **(a/b)**: engine torque (blue), clutch torque (green).

The “SpeedEngineGradient” in both situations are set to 20 while the P-gain is varied in Figure 5.8 and Figure 5.9. On the left (Figure 5.8a / Figure 5.9a) the P-gain is set again to 0.5 and on the right (Figure 5.8b / Figure 5.9b) the P-gain is set to 3. The behavior is different on the right since the engine speed is following the desired engine speed closer compared to the left side. Also, the response is worsened on the right indicated by the delayed clutch torque.

This simple example illustrates how calibration parameters affect the driving behavior. Nevertheless, the model is very simplified and not comparable to e.g., the SiL environment but illustrates the scheme of the models behind the embedded software solution.

6 Target State Optimization

The TSO approach suggested in [1] and [2] is motivated by the stated drawbacks of the existing knowledge-free self-learning algorithms of Section 3.3 and combines the advantages of RL and SL.

The interaction with an environment is comparable to RL (Figure 3.1). The action of the TSO agent affects the environment, which then reacts with a state. Similar to the RL algorithm DQL the agent is a feedforward neural network model which is trained during optimization based on the dataset that is created in parallel. In contrast to RL and TSO an existing dataset is used in SL, to train and optimize a neural network.

The quality of an action is measured with the reward (as explained in Section 4.3) but it is not used to optimize the actions in contrast to RL. Anyway, if the reward is zero it is indicating a successful optimization. The flowchart of

Figure 6.1 illustrates the main loop of the TSO algorithm. The generation of the action is further explained in Section 6.1. To illustrate the relevance of the hyperparameters of neural networks a brief introduction in neural networks is outlined in Section 6.2. These hyperparameters as well as the weights of the model are optimized during optimization. The optimization of the neural network model is further explained in Section 6.3. Another hyperparameter is the activation function of a neural network which is illustrated in Section 6.4. The batch size of neural networks influences the training process and is introduced in Section 6.5. The robustness of the algorithm is tested in Section 6.6.

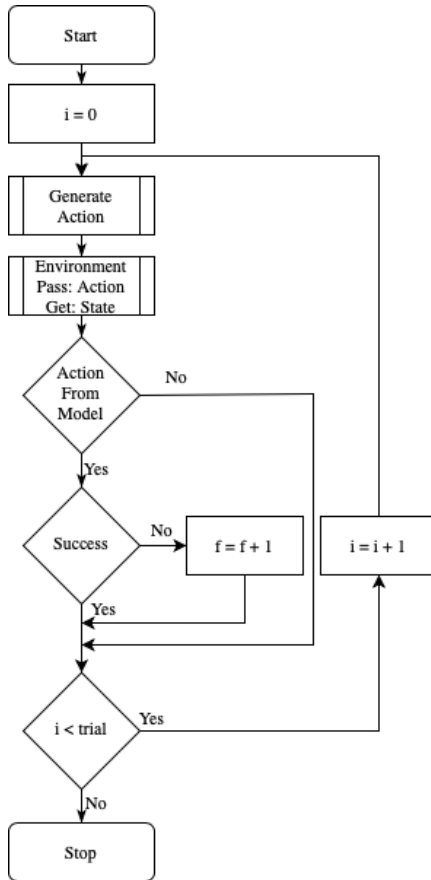


Figure 6.1. Flowchart of the TSO algorithm: main loop.

The entire algorithm is implemented in Python to enable the option to communicate with the SiL environment as well as with the programming interface of the calibration software INCA.

6.1 Generation of the Action

The trade-off between exploration and exploitation affects optimization algorithms like GAs [100] and RL [101] algorithms. An optimization algorithm tries to find (explore) good solutions in the state space. After a good solution has been found one option is to exploit the state space to find similar solutions which are maybe better. A possible disadvantage is that new solutions are not

being explored because the optimization is stuck in a local minimum [100].

To solve this problem, Mnih [52], [56] considered the ϵ -Greedy approach for DQL to ensure a trade-off between exploration and exploitation Mnih [56]:

$$\epsilon(i) = \max(\epsilon_{dec}^i, \epsilon_{min}) \tag{6.1}$$

The equation above illustrates the ϵ -Greedy approach where ϵ is decreasing with ϵ_{dec} in each iteration i until ϵ_{min} is reached. Epsilon's progression for different values is shown in Figure 6.2.

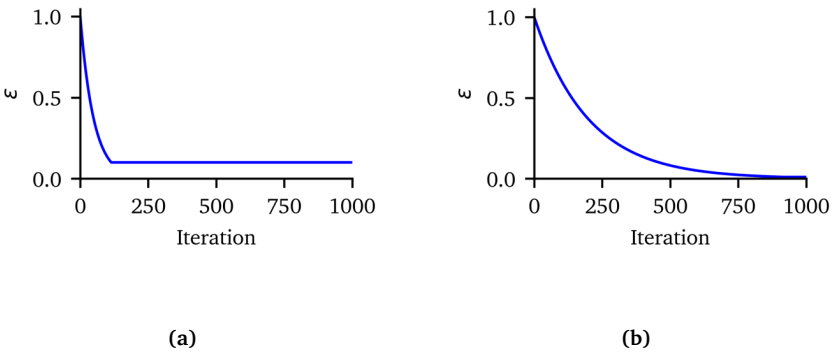


Figure 6.2. Progress of ϵ with **(a)** $\epsilon_{dec} = 0.98$ and $\epsilon_{min} = 0.1$, **(b)** $\epsilon_{dec} = 0.995$ and $\epsilon_{min} = 0.01$.

The action type is determined in parallel by generating a random number r between 0 and 1 in each iteration. If the random number is greater than $\epsilon(i)$, the neural network generates an action, otherwise an action is generated randomly. Therefore, the first actions are primarily generated randomly (to explore the state space) because ϵ is decreasing over time, whereas the model generates later actions with exploitation in focus.

The RL algorithm described in Mnih [52] is introduced to optimize combinatorial problems (where an optimization can last for multiple steps) and tries to predict an action, which maximizes the reward depending on the previous state of the environment. After executing the action another state is achieved and the next action is predicted. In contrast, the TSO algorithm is not intended to optimize combinatorial problems. Therefore, during an optimization with the TSO algorithm the initial state in each step is equal. The neural network learns from previous state-action pairs after a certain number of iterations (it), by mapping the environmental states to the actions that caused them. The neural network is supposed to learn the system behavior of the environment through the early random actions that are assumed to cover a large state space. Depending on the ϵ -Greedy approach (if an action is requested from the neural network) the neural network receives the optimization objectives as a target state to forecast an action. The environmental state resulting

from the action is expected to be equivalent the target state (based to the previous assumptions of having learned prior state-action pairs). An indication that the neural network needs to be optimized or more random actions are required is given if the state from the environment is not equal to the target state. The optimization is successful, and the reward is zero if the resulting state is equal to the target state. The generation of the action within the TSO algorithm is illustrated in the flowchart of Figure 6.3.

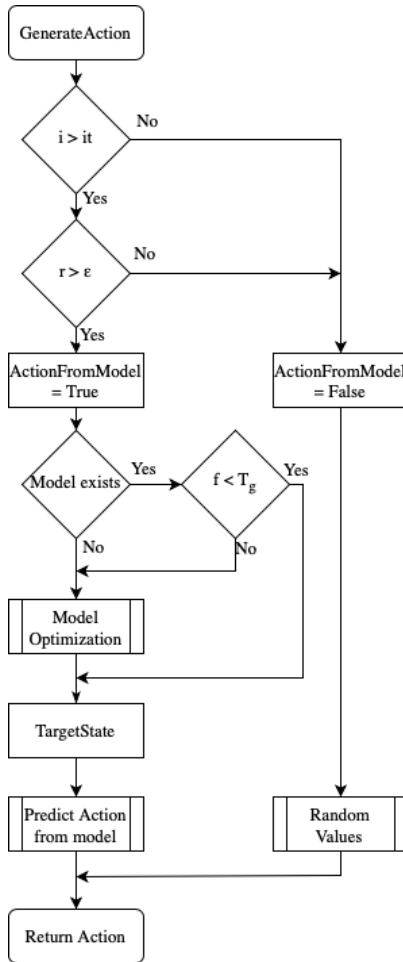


Figure 6.3. Flowchart of the TSO algorithm: generation of the action.

If an action is taken by the model, which has not led to a successful result the fail count (f) is incremented. The model has to be created or optimized if the model is not existing or the threshold T_g is exceeded by the fail count. To understand the model optimization Section 6.2 is introduced to get a brief overview over the hyperparameters of a neural network.

6.2 Brief Introduction into Neural Networks

This section is introduced since the TSO algorithms uses hyperparameter optimization and should therefore provide a brief introduction into neural networks.

A neural network computes the relationship between its input and its output, and hence it is acting as a function estimator [102]. The number of units (neurons) of each layer and the number of layers are the model design hyperparameters [103] which are detected during hyperparameter-tuning in Section 6.3.

According to Yang and Shami [104] all neural network models have an input and an output layer. The number of hidden layers influences the complexity of a neural network [104]. Figure 6.4 illustrates such a neural network architecture [103].

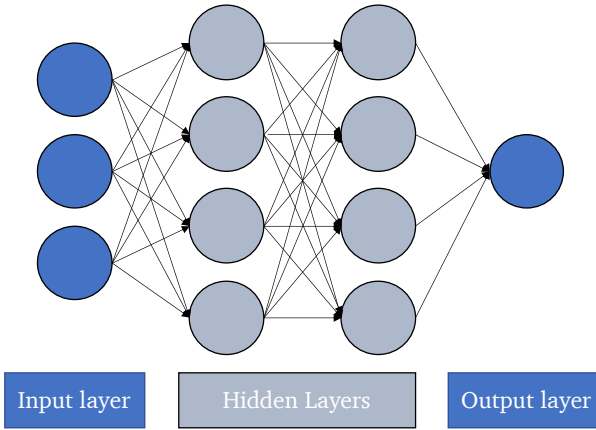


Figure 6.4. Neural network architecture according to [103].

To determine the input of a single unit a of a neural network a set of input variables d is weighted with w_i [105]:

$$a = \sum_{i=1}^d w_i \cdot x_i + w_0 \quad (6.2)$$

The processing of the unit a within a nonlinear activation function g yields the output z of a unit:

$$z = g(a) \quad (6.3)$$

The representation changes for multiple units with the same inputs (single layer neural network) for $x_0 = 1$ to:

$$z_j = g \left(\sum_{i=0}^d w_{ji} \cdot x_i \right), \quad (6.4)$$

In cases where there are multiple layers, the output of layer $(j + 1)$ is based on the output of the previous layer j , which is represented in:

$$z_{j+1} = g \left(\sum_{j=0}^m w_{(j+1)j} \cdot z_j \right), \quad (6.5)$$

Thus, the notation of z_{j+1} is represented for a neural network with two layers as follows:

$$\hat{y} = z_{j+1} = g \left(\sum_{j=0}^m w_{(j+1)j} \cdot g \left(\sum_{i=0}^d w_{ji} \cdot x_i \right) \right) \quad (6.6)$$

The weights of the neural network are established within training using the available input (x) and output (y) data. The number of data points is denoted with n . The predictions of the neural network \hat{y} based on the input x are compared with its true value y [105]. This process is called forward propagation due to the fact that the data is passed through the network. The predicted (\hat{y}) and true value (y) are compared using an error (or cost) function like the mean squared error E :

$$E = \frac{1}{n} \sum_{q=1}^n (\hat{y}(x_q, w) - y_q)^2 \quad (6.7)$$

Since Equation (6.6) is insertable into Equation (6.7) the cost function is depending on the weights. By minimizing the cost function with its derivative, the weights are determined during backpropagation. This takes place by taking fixed steps (learning rate) in the direction of negative gradients (gradient descent) until a minimum is approached [105]. The trade-off between faster learning but a potential lack of convergence (with large learning rates) and more precise predictions but longer computation times (with small learning rates) drives the selection of learning rate. This behavior is illustrated with a simplified example in Figure 6.5 according to [104].

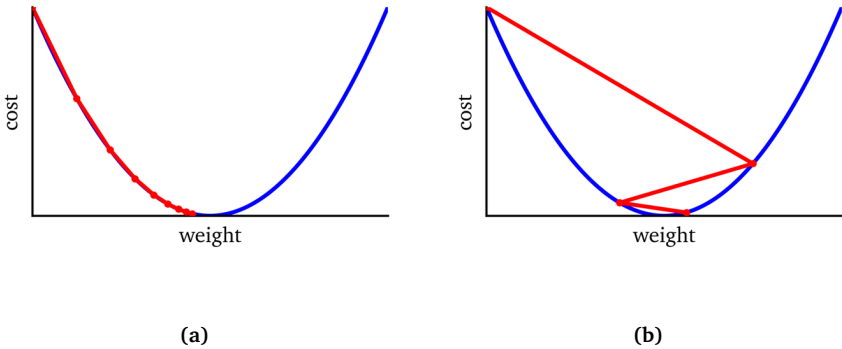


Figure 6.5. Cost function with **(a)** small learning rate, **(b)** large learning rate.

Due to its importance the learning rate [106] but also the number of layers and the number of neurons per layer are optimized within the TSO algorithm. The optimization is illustrated in Section 6.3.

6.3 Model Optimization and Hyperparameter-Tuning

The model parameters (updated during learning) and hyperparameters (which cannot be learned from the data) are two crucial different kind of parameters in ML [104]. The hyperparameters have a major influence on the accuracy of the ML model [107] and have to be defined before starting the training process [108]. For optimizing the predictions of a ML model the hyperparameters have to be adjusted to a dataset [104].

As mentioned previously, the three hyperparameters of Section 6.2 (learning rate, number of layers and the number of neurons per layer) are usually adjusted to a dataset the model should be trained on. Different to SL where hyperparameters are adjusted to an existing dataset, the dataset in RL algorithms as well as in the TSO algorithm is growing with the optimization progress. Therefore, the hyperparameters in RL algorithms are typically not adjusted to the dataset during optimization (according to the research carried out about RL algorithms by this date). In an optimization algorithm this can result in an unsatisfying performance since the accuracy of a neural network can be negatively affected by a misleading set of initial hyperparameters.

As described in Section 6.1 the data is collected with randomly generated actions for a certain number of iterations (iT) to bypass the absence of a dataset. During these iterations the neural network in TSO does not exist. After these iterations and if an action should be generated by the neural network (requested by the ϵ -greedy approach – Figure 6.2), the hyperparameter optimization is carried out to generate the neural network model. The hyperparameter-tuning is carried out with the Bayesian optimization based on Gaussian Processes and the collected state-action

pairs. The Bayesian optimization outperforms other hyperparameter-tuning methods (e.g., grid-search, random-search, etc.) [109] and is hence used within TSO.

If the neural network's predictions are still inaccurate (after the hyperparameter-tuning), it is optimized again.

For indicating the need of an optimization and to differentiate between optimization methods the fail counter (f) is introduced as well as the gradient threshold (T_g), and the tuning threshold (T_t). The fail counter f is incremented if the action is generated by the neural network and the target state is not equal to the environmental state. If a neural network already exists and this behavior occurs multiple times ($f > T_g$) it is re-trained with the new collected data. The hyperparameter-tuning is repeated and a new model n is created if the model still fails ($f > T_t$) to predict an action which is leading to the target state. The tuning threshold (T_t) is greater than the gradient threshold (T_g) to ensure that the model is first re-trained before the hyperparameter-tuning is performed. Since the hyperparameter-tuning is very expensive regarding computation time, the optimization is only performed m times. Every new created model is additionally compared with the ones which have already been created during previous optimization processes.

Figure 6.6 illustrates the model optimization process of the TSO algorithm.

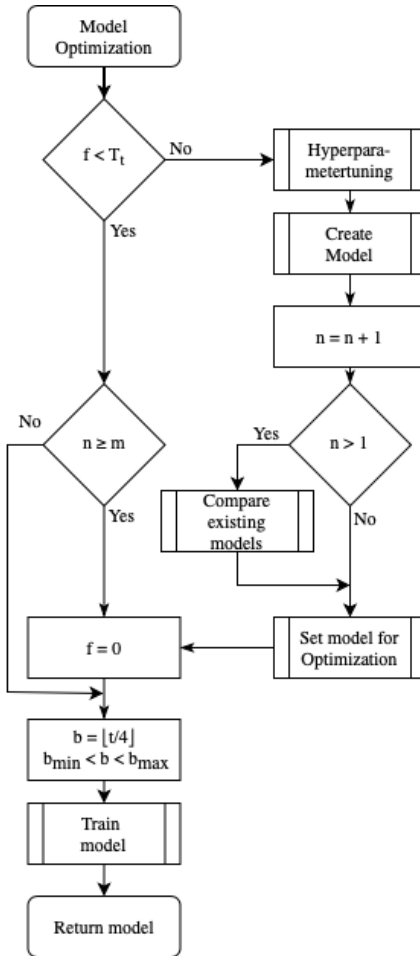


Figure 6.6. Flowchart of the model optimization.

The development of the hyperparameters due to hyperparameter-tuning is exemplary illustrated in Table 6.1 and Table 6.2. In this example two test runs in the SiL environment (Section 5.2) are carried out. In each test run 1000 iterations (vehicle launches) are performed. The optimization of the hyperparameters is carried out three times in this example.

Table 6.1. Hyperparameter propagation test run 1.

Test Run / Iteration	Layer	Neurons	Learning Rate
7	1	512	0.0237
16	1	25	0.0545
23	1	512	0.0237

In Table 6.1 it is observable that in any optimization a neural network with one layer is preferred. Further it is observable that in the third hyperparameter-tuning process a neural network architecture is generated which has not reached a better performance than the neural network architecture of the first tuning process (the hyperparameter set is equal to the first one after the model comparison).

Table 6.2. Hyperparameter propagation test run 2.

Test Run / Iteration	Layer	Neurons	Learning Rate
6	6	130	0.0247
16	1	180	0.0414
20	4	425	0.0012

Other than in Table 6.1 in Table 6.2 it is observable that the hyperparameter-tuning in each tuning process led to a different neural network architecture.

Although the optimization objectives in both test runs are equal the hyperparameters are different. This issue is probably caused by the fact that the first iterations are performed with randomly chosen action values leading to a different exploration of the state space. Therefore, despite equal targets the state-action pairs for the hyperparameter-tuning are different.

6.4 Relevance of the Activation Function

As part of the error function (shown in Section 6.2) the activation function affects the training. To reduce the computation time the activation function is not determined during hyperparameter-tuning of Section 6.3 (although it is also a hyperparameter), instead three common activation functions are investigated in this Section. These are: (a) the linear identity activation function, (b) the rectified linear unit (ReLU) and (c) the logistic sigmoid activation function [110] illustrated in Figure 6.7:

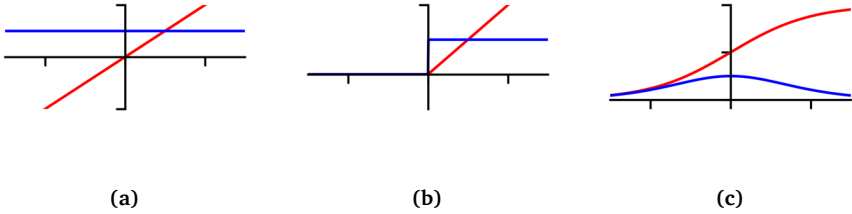


Figure 6.7. **(a)** linear, **(b)** ReLU, and **(c)** logistic sigmoid activation function (red) and its derivative (blue).

The following equation is representing the linear activation function:

$$g(x) = x \tag{6.8}$$

The linear activation function simplifies the neural network into a simple matrix multiplication since it reduces the neural network to a single-layer network. Thus, results in limited computational capabilities. For more sophisticated tasks a neural network with multiple layers is required (deep neural network) [105] which prerequisites a non-linear activation function like the ReLU function [111]:

$$g(x) = \max(0, x) \tag{6.9}$$

or the logistic sigmoid activation function [105]:

$$g(x) = \frac{1}{1 + e^{-x}} \tag{6.10}$$

Using the logistic sigmoid function comes with the drawback of having a greater computational effort compared to implementing the rectified linear unit (ReLU). The issue becomes more relevant with a growing number of weights due to added layers and/or neurons since the weight estimation becomes computationally more expensive [111]. To investigate the influence of the activation function on the performance of the optimization the TSO algorithm is tested with both activation functions within the SiL environment. For this purpose, each activation function is tested within five test runs (one test run is containing 1000 iterations). The performance is determined with the average number of successful iterations \bar{x}_5 and the average number of the first successful iteration \bar{x}_F out of the five test runs. For both activation functions equal optimization targets are proposed. The results are illustrated in Table 6.3.

Table 6.3. Comparison of the activation functions.

Activation function	\bar{x}_S	\bar{x}_F
Sigmoidal	7.8	221.6
ReLU	250.0	20.6

It is observable that the performance of the TSO algorithm is significantly affected by the chosen activation function. Figure 6.8 illustrates the difference of the behavior:

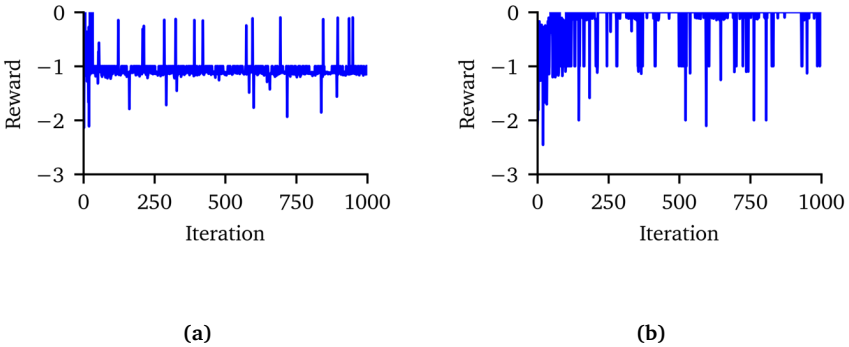


Figure 6.8. Comparison of the influence of the activation functions of the TSO algorithm on the results. **(a)** sigmoidal activation function, **(b)** ReLU activation function.

The results indicate that the ReLU activation function is preferable to be used for the TSO algorithm.

6.5 Batch Size

The number of training samples used to update the neural network model is determined by another hyperparameter called batch size [112]. The batch size influences the behavior of the neural network in terms of accuracy of the predictions but also during training. Thus, a bad generalization and inaccurate predictions may be the outcome of a batch size which is too large. A better quality but longer computation times, in contrast, are the result of a too small batch size [113]. Since the dataset is increasing its size (during the optimization with the TSO algorithm), the batch size is not kept at a constant value in contrast to e.g., SL or RL. An increasing computation time would be observable with an increasing dataset size if the batch size is small during the entire optimization process. In contrast the possibility of misleading predictions could occur over a longer time period with a large batch size. Thus, the batch size b in TSO is dependent on the size of the dataset

t and determined as follows:

$$b = \left\lfloor \frac{t}{4} \right\rfloor \quad (6.11)$$

$$b_{min} < b < b_{max} \quad (6.12)$$

The dataset size, t , is used to dynamically determine the batch size based on the optimization progress. The batch size's minimum and maximum values are b_{min} and b_{max} .

6.6 Robustness

Since in real world environments it is difficult to find stationary conditions, optimization algorithms should be robust. Robustness means the ability of a system to function under perturbations and hence under changing conditions according to Kitano [114]. The optimization of the launch behavior is usually not carried out in a stationary environment (outside simulations) since roads or test tracks are not entirely flat. For a desired optimization of the launch at flat conditions obviously roads with high slopes should be avoided. Nevertheless, roads without any slope are hardly available. However, small changes in the driving resistance urge the system to behave differently compared to optimal conditions. To test the robustness of the TSO algorithm a real-world behavior is simulated within the SiL environment by altering the slope randomly at each vehicle launch between -1% and 1% as well as between -2% and 2% . This test is repeated four times with 100 launches each. Table 6.4 illustrates the results.

Table 6.4. Successful iterations of the robustness test with the TSO algorithm.

Test run	$\pm 1 \%$	$\pm 2 \%$
1	20	21
2	78	33
3	17	74
4	7	0
Average	30.5	32

As shown in Table 6.4 the results vary strongly. With the outliers of 78 and 74 successful launches and one outlier without any successful launch the behavior is very different. The average number of successful launches in both test cases are very similar anyway (30.5 for $\pm 1\%$, 32 for $\pm 2\%$ slope). Despite the varying results it indicates that minor changes in the system behavior are acceptable for the optimization with the TSO algorithm.

7 Results of the calibration of the launch behavior

To find a calibration resulting in a driving behavior which matches the objectives of Chapter 4 several self-learning algorithms have been applied. First, tests have been carried out with the TSO algorithm deployed on the simplified model of Section 5.3 to evaluate the potential of the algorithm. In the SiL environment tests are carried out with existing RL algorithms and the GA NSGA-II. These results are compared with the performance of the TSO algorithm in the SiL environment. To verify the most promising algorithms of the SiL environment the tests are repeated in different test vehicles.

In every optimization the values of the calibration parameters are limited by experienced calibration engineers to a defined range. Therefore, the search space is limited to avoid long optimization times and to ensure the plausibility of the parameters after the optimization. The latter issue is important since sometimes after the optimization it is required to adjust the parameters by calibration engineers. Thus, if a parameter set which is leading to a successful launch is found it is ensured that the parameters are in a similar range to those the calibration engineer would have chosen. Otherwise, it would be difficult for the engineers to modify the optimized parameters.

The range of the parameters are equal for every algorithm. The range of the parameters for the optimization and the resulting calibration parameters itself are illustrated for the simplified model in Section 7.1. The behavior is equal for the other test environments but not illustrated in detail due to confidentiality reasons.

7.1 Simplified Model

To investigate the potential of the TSO algorithm it is tested with the ReLU activation function (see Section 6.4) within the simplified model of Section 5.3. For testing the algorithm only two of the four objectives of Chapter 4 are used for optimization: (1) the reaction time objective of Section 4.1.2 and (2) the clutch torque objective of Section 4.2.2. The other objectives remain constant due to the simplified behavior and are therefore neglected. While the clutch torque objective should be zero (since it is a discomfort objective) the reference value of the reaction time objective has to be chosen. An illustration of different driving behaviors is shown in Figure 5.7a with a reaction time of 0.1 s and in Figure 5.9b with a reaction time of 0.4 s. Within these two figures the differences of the clutch torque affected by the reaction time is clearly visible.

Generalization tests

To test the generalization of the algorithm and therefore to prove the ability of the TSO algorithm to learn the system behavior despite varying the test set up the reference value of the reaction time objective is varied between 0.35 s and 0.5 s. For each set of objectives, the TSO algorithm is applied five times on the simplified model. Every test run of these five test runs contains 1000 launch iterations (same test scenario as in Section 6.4 Relevance of the Activation Function). If the reward of one launch is zero, the objectives are met, and the target is reached by the TSO algorithm. Table 7.1 illustrates the performance of the TSO algorithm regarding the test scenario. The average number of the first successful iteration is denoted with \bar{x}_f which is the iteration the reward is zero the

first time. The standard deviation of this value is σ_F . The average number of successful iterations (average number of iterations where the reward is zero of the 1000 iterations) is denoted with \bar{x}_S . Accordingly, the standard deviation is illustrated with σ_S . The outliers are indicated with min_S (lowest number of successful iterations out of the five test runs) and max_S (highest number respectively).

Table 7.1. TSO algorithm applied on the simplified model.

Reaction Time	\bar{x}_S	σ_S	min_S	max_S	\bar{x}_F	σ_F
0.35 s	759.2	27.1	732.0	804.0	138.2	57.4
0.4 s	713.2	114.9	588.0	829.0	106.6	64.1
0.45 s	734.6	52.1	650.0	781.0	122.4	58.8
0.5 s	0.0	-	0.0	0.0	-	-

It turns out that a set of calibration parameters for meeting the objectives can be found by the TSO algorithm often in almost every test run. Only the objective combination of a reaction time of 0.5 s while meeting the clutch torque objective is not approachable by the algorithm in this test environment.

Further investigation of the 1000 launches of the fourth of the five test runs for the reaction time objective of 0.4 s

An example of such a test run is illustrated in Figure 7.1. The figure illustrates the results (of the fourth of five test runs) of the TSO algorithm applied to the simplified model with a reaction time objective of 0.4 s. The red marks are indicating where a successful result occurs. In the example below 794 of the 1000 launches were successful. The first successful launch occurs at iteration 48.

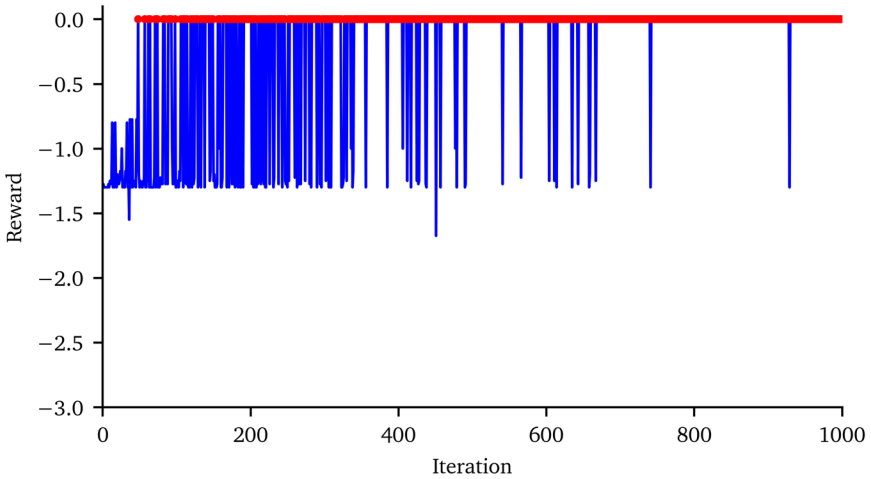


Figure 7.1. Results of the fourth of five test runs of the TSO algorithm applied to the simplified model with a reaction time objective of 0.4 s. The red dots indicate successful launches.

The 794 successful launches of Figure 7.1 have been achieved with five sets of parameters. The parameter sets are all in the range of Table 7.2.

Table 7.2. Parameter range of the simplified model for the optimization.

Parameter	minimum	maximum
“SpeedEngineGradient”	3 rpm/10ms	500 rpm/10ms
P-Gain	0.001	10

The P-Gain is in contrast to the “SpeedEngineGradient” not a scalar value. Instead, it is a curve which is dependent on the speed error of the engine speed and the desired engine speed. This curve is illustrated in Table 5.2.

The parameter sets achieving the successful results are illustrated in Table 7.3. The iteration the parameter set is leading to a successful iteration the first time is denoted with x_P . The overall number of successful iterations of a parameter set is denoted with x_S .

Table 7.3. Parameter sets of the fourth of five test runs leading to successful results in the simplified model for a reaction time objective of 0.4s.

No	x_F	x_S	"SpeedEngine-Gradient"	P-Gain over Engine Speed error				
				-500	-250	0	250	500
1	48	790	33.38	1.14	0.74	0.69	0.90	0.76
2	114	1	283.39	0.97	0.52	0.03	6.37	6.57
3	198	1	397.19	9.79	0.59	0.19	9.08	5.07
4	273	1	496.26	6.36	0.39	0.37	0.52	8.41
5	538	1	75.72	5.32	0.20	0.82	5.39	6.01

In Table 7.3 it is observable that the first parameter set is leading to the greatest number of successful results. The other parameter sets are leading to one successful result only.

Anyway, to validate these five parameter sets which are leading to 794 successful launches of the 1000 launches of the fourth test run and to ensure that these parameter sets did not accidentally lead to a successful result (especially the ones with only one successful result) the parameter sets are again tested within the simplified model multiple times. The measured reaction times of these parameter sets are illustrated in Table 7.4. The reaction time values have been generated reproducibly by the parameter sets. The tolerance thresholds of 10% have not been reached for the reaction time objective hence the objective obj_{rea} is zero for every parameter set as well as the clutch torque objective obj_T .

Table 7.4. Results of the different successfully applied parameter sets (generated by the TSO algorithm in the fourth test run on the simplified model) for a reaction time objective of 0.4 s.

No	Reaction Time
1	0.43 s
2	0.40 s
3	0.41 s
4	0.40 s
5	0.39 s

Epsilon Greedy

As stated in Section 6.1 the ϵ -Greedy approach of Mnih [52], [56] is introduced to TSO (to determine whether an action is generated by the neural network model or randomly) to avoid being stuck in a local minima. The action distribution of randomly generated actions and actions generated by the neural network model (according to Section 6.1) is illustrated in Figure 7.2. As shown previously in Figure 6.2 the blue line indicates the value of epsilon during the optimization progress. The randomly generated number to decide if an action is generated randomly or by the neural network model is indicated by the 1000 dots. The green dots indicate where actions are generated by the neural network model. The red dots indicate where actions are generated randomly.

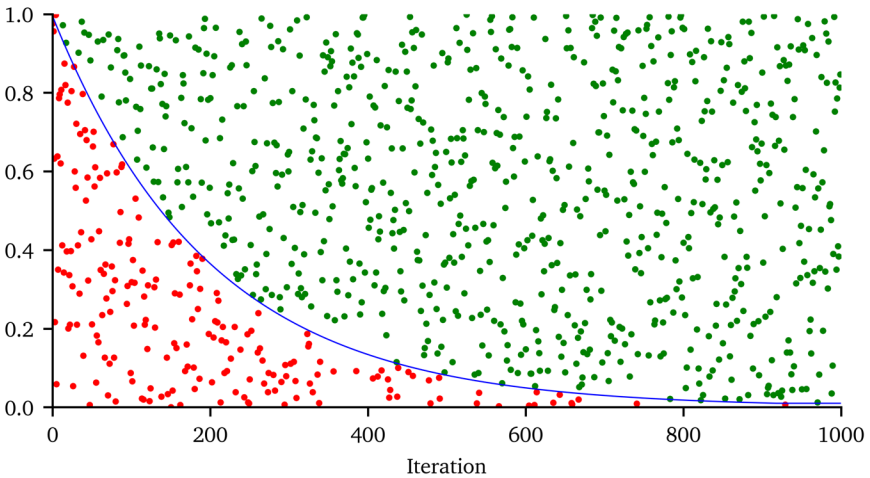


Figure 7.2. Action distribution generated by the TSO algorithm in the fourth test run on the simplified model for a reaction time objective of 0.4 s; red: randomly generated actions, green: predicted actions of the neural network, blue: epsilon threshold.

It is observable that iterations in an advanced optimization process are more likely to be determined by the neural network model as mentioned previously. It is also observable that latter randomly generated actions (red dots between iteration 700 and iteration 1000) lead to an exploration of the state space without leading to successful iterations (observable at the corresponding iterations in Figure 7.1 – reward is below zero). This behavior is also observable in early stages of the optimization of Figure 7.1.

The influence of ϵ -Greedy approach is also observable in Table 7.3. Except the first parameter set the other parameter sets are only leading to one successful result. Figure 7.3 is illustrating only

the successful iterations of the optimization in the same manner as in Figure 7.2.

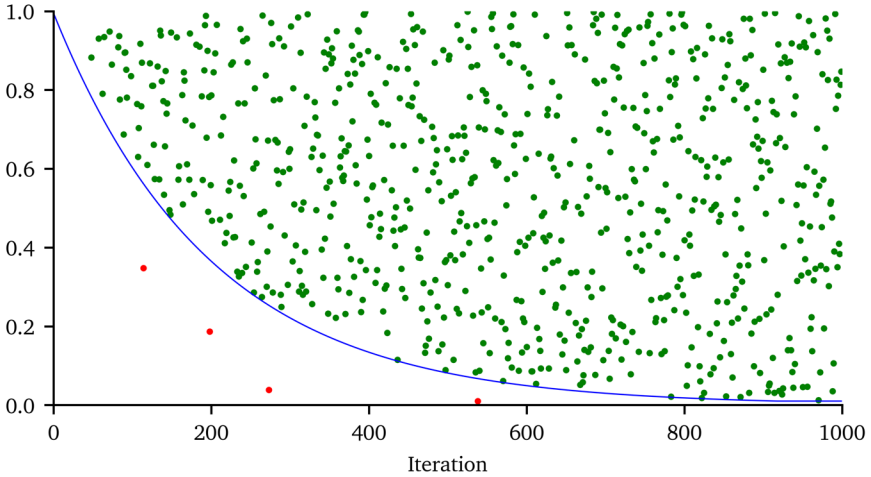


Figure 7.3. Action distribution of the successful iterations; red: randomly generated actions, green: predicted actions of the neural network, blue: epsilon threshold.

The four red dots are indicating the four successful results which have occurred only one time (randomly chosen). The green dots indicate the 790 iterations leading to successful results through the neural network model.

Anyway, although the first parameter set is leading to a measured reaction time with the greatest deviation compared to the reference reaction time (among the other parameter sets) it produced by far the most successful results. Further, it is observable that many parameter sets can produce a satisfying result.

Choosing the parameter set

Despite the possibility of choosing several parameter sets out of Table 7.3 which would lead to a successful launch behavior the advice for the calibration engineer in this and the following cases of Section 7.2 and Section 7.3 is to choose the one with the most repetitive successful results from the optimization. In Table 7.3 this is the first parameter set with 790 successful launches out of 1000. Therefore, a robust calibration is guaranteed.

Anyway, although the robustness is a criterion of choosing the parameter set generated by the TSO algorithm it is worth taking an insight in the distribution of the P-Gain. Since calibration engineers selecting calibration parameters not only driven by data (unlike TSO), the probability of calibrating the parameter set in the same manner manually is very unlikely. The P-Gain of the first and the

fourth parameter set of Table 7.3 are illustrated in Figure 7.4.

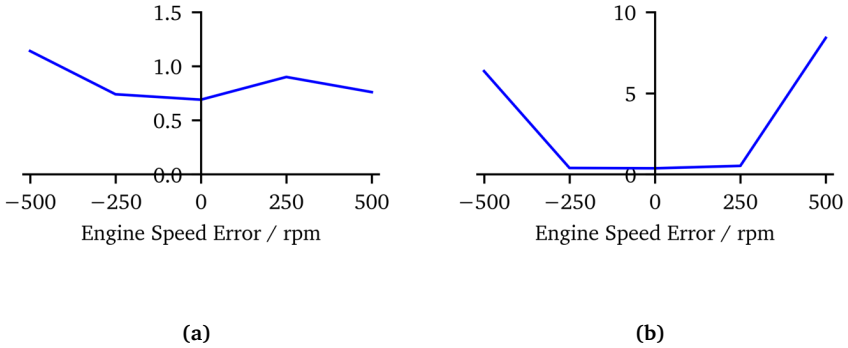


Figure 7.4. P-Gain of the different parameter sets of Table 7.3. (a) Parameter set 1 recommended to be chosen ($x_S = 790$), (b) Parameter set 4 ($x_S = 1$).

While the P-Gain of the first parameter set does not follow a pattern the P-Gain of the fourth parameter set does. In the fourth parameter set for high absolute values of the engine speed error the P-Gain value is also high. In contrast for low values of the engine speed error the P-Gain is low. These parameters are more likely to be chosen manually since a calibration engineer chooses parameters driven by domain knowledge and therefore would eventually treat negative and positive deviations of the engine speed similarly. Anyway, since the TSO algorithm and the other evaluated self-learning optimization algorithms do not have any domain knowledge the calibration engineer should use (as stated previously) the calibration leading to the most successful results.

Conclusion

Concluding, it can be observed that the algorithm is producing repeatedly successful results within the simplified model despite varying the optimization objectives. Therefore, the algorithm is further applied within the more complex SiL environment which includes the same TCU software solution as a test vehicle.

7.2 Software in the Loop (SiL) environment

The TSO algorithm has shown its potential in Section 7.1. Regardless of the promising results a comparison with existing established algorithms has to be carried out in a more sophisticated development environment. Nevertheless, to ensure reproducible results the virtual vehicle of Section 5.2 is used for comparing these algorithms. The ability to reproduce results without external influences (environmental conditions) comes with the advantage of only having an impact on the driving behavior through varying the parameters of the TCU (comparable to Table 5.1). Beside the

TSO algorithm the following self-learning algorithms are applied within the SiL environment to determine the calibration parameters: NSGA-II (GA), the RL algorithms: Deep Deterministic Policy Gradient (DDPG) [58], Proximal Policy Optimization (PPO) [115], Advantage Actor-Critic (A2C) [116] and Soft Actor-Critic (SAC) [117].

As mentioned, the launch behavior is optimized with regard on the objectives of Sections 4.1 and 4.2. To monitor the performance of each algorithm in scope of the optimization problem – comparable to Section 7.1 – the number of successful iterations as well as the iteration where the first successful iteration occurs are compared. To reduce the influence of having accidentally favorable or unfavorable starting conditions each algorithm is tested within five test runs (as previously shown in Section 6.4 and Section 7.1). This method is applied since the first iterations are usually performed with a set of randomly chosen parameters (which also influences the hyperparameters – see Section 6.3). Since self-learning algorithms are making decisions based on experience the starting conditions can influence an optimization towards a more positive or negative progress. To overcome this issue Zaglauer [85] identified the starting population with a DOE. The DOE is introduced to find several solutions within the search space to shorten the evaluation time with the gained knowledge [85]. To set up such a DOE it is required to have domain knowledge which makes the method difficult to generalize (the algorithm should also find solutions properly for different system behaviors like different combustion engine behaviors). Since the optimization method should work e.g., in different environmental conditions and for different optimization problems (and as stated in Section 3 knowledge-free) a DOE is not intended.

The mentioned test runs to measure the performance of an algorithm contains 1000 iterations and therefore in this case 1000 vehicle launches within the SiL environment. If a successful launch has been recognized, the optimization however proceeds until all 1000 iterations have been carried out to avoid the chance of having a success with randomly generated calibration parameters. Table 7.5 illustrates the results of the five test runs. As previously shown in Table 7.1 the outliers are indicated with the columns min_S for the lowest number of successful iterations out of the five test runs and max_S with the highest number respectively. The number of successful iterations is averaged in \bar{x}_S and the standard deviation is denoted in σ_S . Accordingly, the first success (reward is zero for the first time) is illustrated with \bar{x}_F and the standard deviation is illustrated in column σ_F . The results are based on the target state of Table 4.1.

Table 7.5. Comparison of TSO with existing algorithms.

Algorithm	\bar{x}_S	σ_S	min_S	max_S	\bar{x}_F	σ_F
NSGA-II	104.2	15.37	84	126	21.0	13.8
DDPG	0.0	-	0	0	-	-
PPO	1.2	1.30	0	3	363.0	261.6
A2C	2.0	1.87	0	5	332.8	274.3
SAC	28.8	4.15	22	32	46.4	31.1
TSO	250.0	160.4	36	442	20.6	11.3

The GA NSGA-II outperforms the RL algorithms, and the standard deviation indicated reproducible results. Therefore Table 7.5 proves the mentioned drawbacks of Section 3.3 regarding RL. Since RL algorithms need many iterations to learn the policy [62], [66] 1000 iterations in each test run without known starting conditions from a DOE turns out to be not enough to gain promising results. This indicates that the tested RL algorithms might not be the right choice for optimizing the behavior of a vehicle in a non-simulated environment and hence in a time-consuming development process. Despite the drawbacks of RL algorithms, the SAC algorithm led to some promising results even though they were not comparable to those of the GA. Figure 7.5 illustrates the promising results of the NSGA-II algorithm.

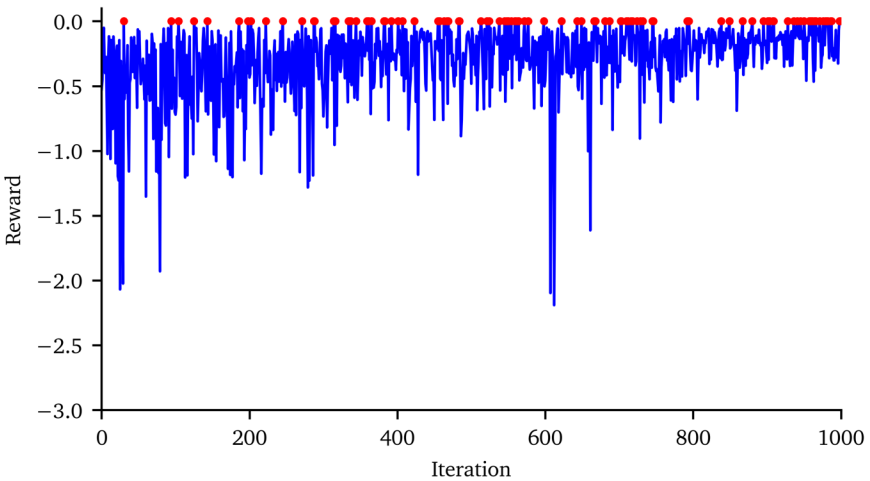


Figure 7.5. Results of an NSGA-II test run in the SiL environment. The red dots indicate successful launches.

The standard deviation of the TSO algorithm for the average successful iterations is greater compared to the other algorithms since one test run produced one outlier with only 36 successful iterations. Nevertheless, the TSO algorithm had in average more successful results compared to the other algorithms. The result of one test run is illustrated in Figure 7.6.

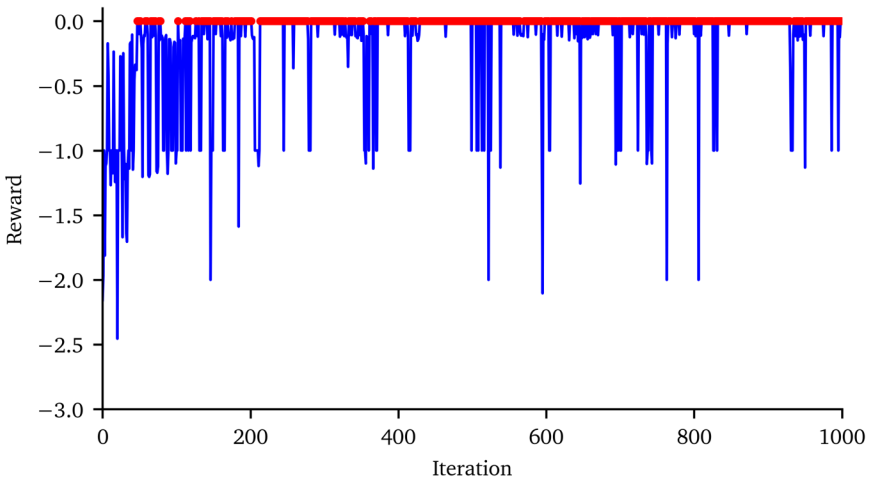


Figure 7.6. Results of the TSO algorithm within the SiL environment. The red dots indicate successful launches.

Further, the results regarding the first successful iteration are comparable to the NSGA-II algorithm.

Although the NSGA-II algorithm has some drawbacks (possibility of a worsened performance with an increasing number of calibration parameters, noisy reward with many outliers – Figure 7.5) it will be applied within different test vehicles as well as the TSO algorithm due to its promising results. In the test vehicles the hyperparameter optimization did not take place to save computation time. Therefore, the TSO algorithm is adjusted and applied with the last combination of the hyperparameters of the SiL environment (layers = 4, neurons = 425, learning rate = 0.0012 – Table 6.2) within the different test vehicles.

7.3 Test vehicles

To validate the NSGA-II and the TSO algorithm in the test vehicles the target values of the optimization objectives are equal. Table 7.6 is introduced to illustrate the target state:

Table 7.6. Optimization targets.

Objective	Value	Unit
Engine Speed Drop	0.00	rpm
Acceleration Peak	3.25	m/s ²
Number of Clutch Torque Minima	0	-
Reaction Time	0.50	s

Test Vehicle: Three-cylinder Otto – FWD – HDT

To validate the algorithms a vehicle with a three-cylinder Otto engine equipped with an HDT is used (the HDT is explained in Section 2.4.3). The vehicle has a front-wheel drive (FWD). Since the time spent in a test vehicle should be minimized (for reducing costs) the number of iterations for the optimization is strived to be as low as possible in contrast to the 1000 iterations per test run of Section 7.2. To realize a lower number of iterations with the NSGA-II algorithm two different hyperparameter configurations are applied. The hyperparameters of the NSGA-II algorithm are explained in Section 3.2 and are set for the first test to a population size of four and a generation size of five. Thus, in total 20 iterations are performed. Since this number is significantly lower compared to the tests in the SiL environment, another hyperparameter set is tested. The generation size is again five, but the population size is set to 12 resulting in 60 iterations. Figure 7.7 illustrates the results:

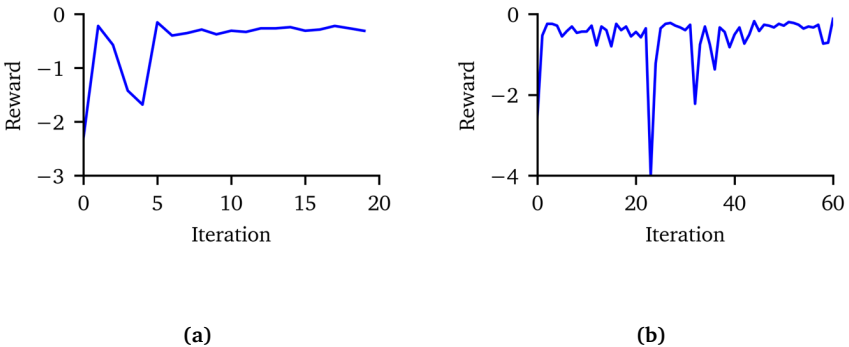


Figure 7.7. NSGA-II results tested in a vehicle with an HDT and a three-cylinder Otto engine :**(a)** Generations: 5, Populations: 4; **(b)** Generations: 5, Populations: 12.

The target state has never been reached with any of the NSGA-II configurations which is observable with the reward which is always below zero.

For testing the TSO algorithm also, a lower number of iterations is applied. To compare the results with the one of the NSGA-II algorithms (of Figure 7.7) only 50 iterations are desired for the optimization. Further, also the test vehicle is the same and the environmental conditions are almost equal (slight differences could be possible regarding the ambient temperature – test is carried out later the same day). As mentioned in Section 7.2 the optimization in the test vehicle is performed with the hyperparameters of the neural network of the SiL environment (number of layers = 4, neurons per layer = 425, learning rate = 0.0012). Figure 7.8 illustrates the results:

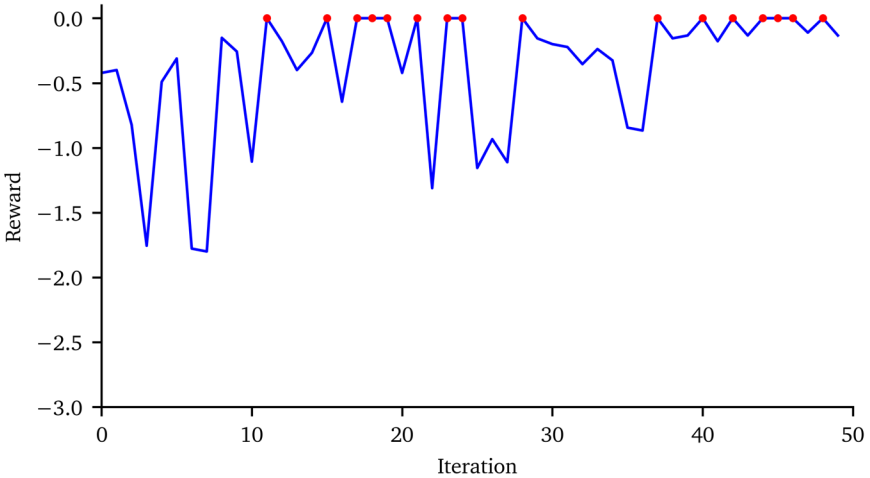
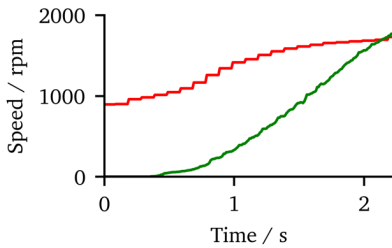


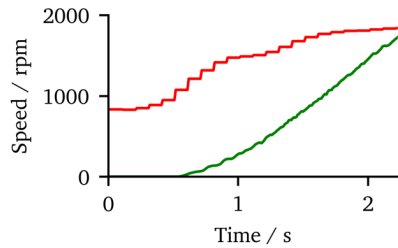
Figure 7.8. Results of the TSO algorithm tested in a test vehicle with an HDT and a three-cylinder Otto engine. The red dots indicate successful launches.

It is observable that the TSO algorithm outperforms the NSGA-II algorithm since the first successful result has been achieved after 11 iterations and overall 16 iterations have been successful out of 50.

Figure 7.9 shows an example of a launch with a calibration leading to a successful result (reward = 0) and a failed launch.



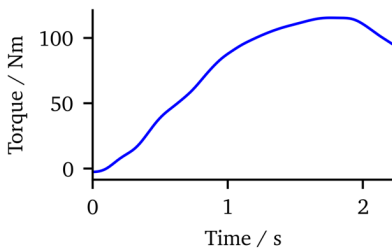
(a)



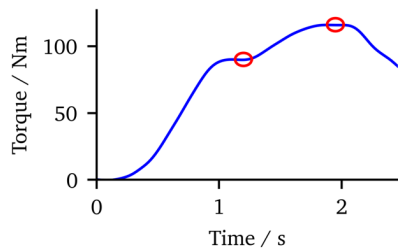
(b)

Figure 7.9. **(a)** Successful launch (TSO: iteration 11 of Figure 7.8), **(b)** Failed launch (NSGA-2: iteration 0 of Figure 7.7a); Engine Speed (red); Input-shaft speed (green).

Figure 7.10 illustrates the corresponding clutch torque of these launches.



(a)



(b)

Figure 7.10. Clutch torques of the different launches accordingly to Figure 7.9: (a), desired clutch torque behavior (b) clutch torque with local minima.

Regarding the engine speed objective of Section 4.2.1 Figure 7.9a illustrates a behavior without discomfort since the engine speed does not drop. Also, Figure 7.10a does not show any local minimum of the clutch torque and hence complies with the clutch torque objective of Section 4.2.2. The customer objectives of Section 4.1 are also met since a maximum acceleration of 3.26 m/s^2 and a reaction time of 0.45 s is measured and hence is within the 10% tolerance of the target values defined in Table 7.6. The second launch (illustrated in Figure 7.9b) is also not negatively

evaluated regarding the engine speed objective since although the engine speed has some inconsistencies the engine speed does not drop. In contrast two local minima of the clutch torque (marked red) are determined in Figure 7.10b while only the one at timestep 1.18 s is clearly observable. Nevertheless, the maximum acceleration measured with 3.24 m/s^2 is within the tolerance in contrast to the reaction time measured with 0.59 s. The reward of the launch illustrated in Figure 7.9b and Figure 7.10b is -2.31 and hence the launch is not successful.

Test Vehicle: Three-cylinder Otto – FWD – DCT

To verify the results the tests have been repeated in another vehicle with a three-cylinder Otto engine but instead of an HDT, a conventional DCT is part of the powertrain. The vehicle again has a FWD. The results of the NSGA-II algorithm are illustrated in Figure 7.11:

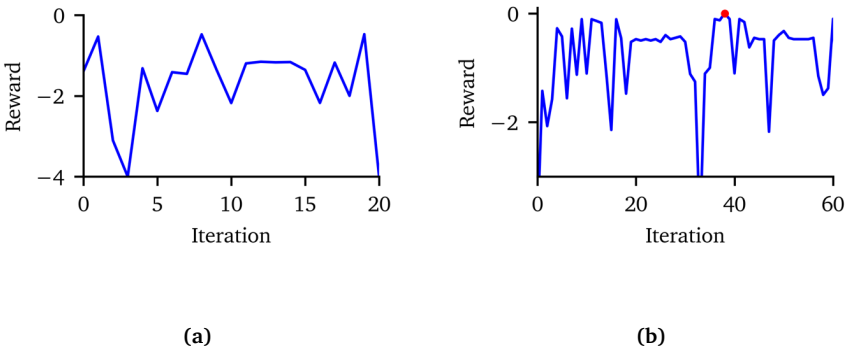


Figure 7.11. NSGA-II results tested in a test vehicle with a conventional DCT and a three-cylinder Otto engine: **(a)** generations: 5, population: 4; **(b)** generations: 5, population: 12. The red dot indicate a successful launch.

Figure 7.12 illustrates the results of the TSO algorithm applied to the same vehicle.

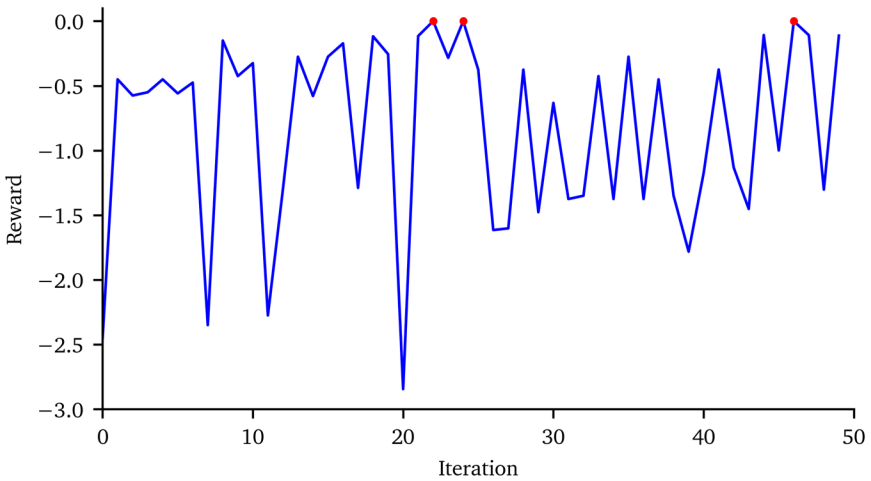


Figure 7.12. Results of the TSO algorithm tested in a test vehicle with a conventional DCT and a three-cylinder Otto engine. The red dots indicate successful launches.

It is observable that the performance of the NSGA-II algorithm is still worse than the performance of the TSO algorithm. Anyway, the TSO algorithm has not achieved as many successful results as illustrated in Figure 7.8 since the target values of the optimization objectives have not been adjusted to the different powertrain configuration. This is necessary since the behavior of the different engines and vehicles can vary strongly despite an equal expression of the driver demand with the same pedal positions.

Test Vehicle: Four-cylinder diesel – AWD – HDT

Since the TSO algorithm outperformed the NSGA-II algorithm twice (and hence it is assumed that it is proven that the NSGA-II algorithm is outperformed in scope of the optimization problem for low numbers of iterations generally) only the TSO algorithm is tested in the third vehicle to verify the ability of the algorithm to generalize for different system behaviors. The third test vehicle differs from the previous test vehicles since the powertrain is equipped with a four-cylinder diesel engine, an HDT and an all-wheel drive (AWD). The first test in this vehicle is again carried out with the target state of Table 7.6 (reference acceleration 3.25 m/s^2 , reference reaction time 0.5 s).

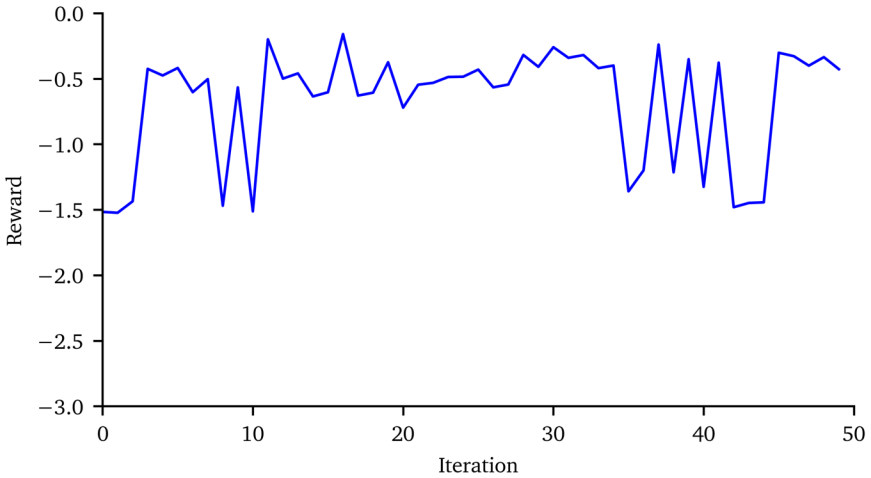


Figure 7.13. Results of the TSO algorithm tested in a test vehicle with an HDT and a four-cylinder diesel engine (Acceleration target: 3.25 m/s², reaction time target: 0.5 s).

Since the engine torque behavior and also the power of a four-cylinder Diesel engine is different compared to the one of the three-cylinder Otto engines the target state of Table 7.6 has not been reached by the TSO algorithm which is illustrated in Figure 7.13 (the accelerator pedal position was set to the same value as in previous tests). Therefore, the target state needs to be adjusted to the physical behavior of the new engine-transmission combination. Different target states with its results are illustrated in Table 7.7:

Table 7.7. Comparison of the TSO algorithm with different optimization targets within the vehicle with a four-cylinder diesel-engine and an HDT.

Acceleration	Reaction Time	Successful	First Success
		Iterations	
3.25 m/s ²	0.5 s	0	-
3.5 m/s ²	0.3 s	16	13
3.5 m/s ²	0.4 s	11	4
4 m/s ²	0.3 s	1	13

It is observable that for the applied accelerator pedal position (and hence the applied engine load)

most successful launches have been determined for the acceleration target of 3.5 m/s^2 and a reaction time target of 0.3 s which is illustrated in Figure 7.14.

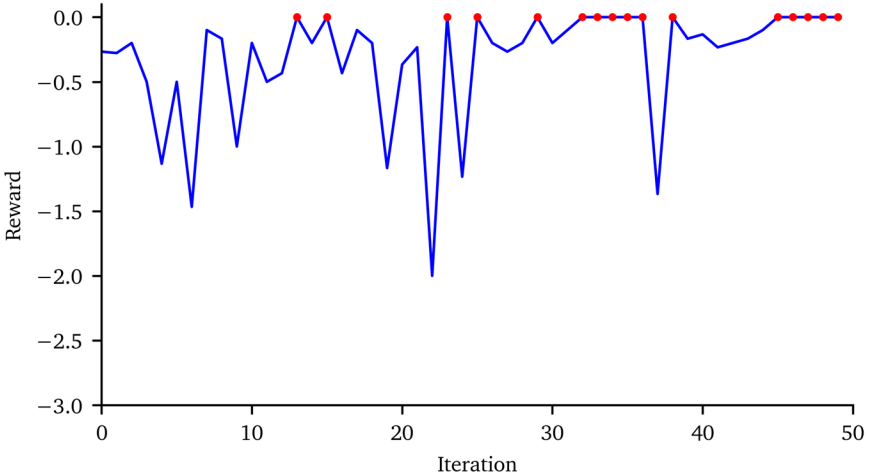


Figure 7.14. Results of the TSO algorithm tested in a test vehicle with an HDT and a four-cylinder diesel engine (Acceleration target: 3.5 m/s^2 , reaction time target: 0.3 s).

If the maximum acceleration is varied lower or higher the success of the optimization is influenced negatively. Varying the reaction time target instead also leads to a slightly worse but acceptable result.

The results illustrate that the TSO algorithm is able to optimize the calibration of the parameters influencing the behavior of the launch of DCT faster than currently available optimization algorithms formerly applied in this domain. Further it is shown that the TSO algorithm is able to generalize by being applied to different vehicles (if the customer objectives are chosen in a way that is reachable by the system behavior).

8 Discussion

The study revealed that the TSO algorithm [1], [2] was able to optimize the calibration parameters of the TCU of DCT regarding the launch behavior faster than common RL algorithms and the commonly used GA NSGA-II (illustrated in Table 7.5). In the SiL environment the performance of the TSO algorithm in terms of successful results showed that the NSGA-II algorithm is outperformed since the TSO algorithm achieved two times more successful results. The best tested RL algorithm (SAC) had an even worse performance in the domain of TCU parameter optimization. The TSO algorithm had nine times more successful results (see Section 7.2). Within the tests with test vehicles the NSGA-II algorithm only had one successful result which is shown in Figure 7.11b (vehicle with the conventional DCT and a three-cylinder Otto engine) while the other tests have not been successful. The TSO algorithm achieved a better performance in both vehicles with the three-cylinder Otto engines (HDT Figure 7.8 and DCT Figure 7.12) but could not find a successful parameter set in the first place for the vehicle with the four-cylinder Diesel engine (see Section 7.3).

Table 7.7 illustrates that some domain knowledge of the system behavior (regarding the vehicle which should be calibrated) is required since as mentioned the calibration of the vehicle with the four-cylinder Diesel engine came with a different system behavior compared to the other test vehicles. Therefore, although the calibration process can be successfully automated, the setting of the customer objectives (see Section 4.1) still requires experienced engineers. The setting of the optimization objectives is not only an issue in the optimization with the TSO algorithm in fact the issue also exists for any other optimization algorithm. To further shorten the optimization time an option could be generally setting the objective values of the customer objectives (Section 4.1) to zero if the reaction time is below its reference value (see Section 4.1.2) or if the maximum acceleration exceeds its reference value (Section 4.1.1). Both cases would lead to an improved driving behavior although the tolerance thresholds are exceeded. Also, the dynamical adjustment of the reference values dependent on the system behavior could be an approach for further studies as long as the other objectives are still reachable.

In Section 7.1 the selection of the parameter set from the ones leading to successful results is illustrated. Therefore, the selection is driven by the robustness and hence the parameter set leading to the highest number of successful results should be chosen by the calibration engineer. Anyway, this could lead to the drawback of having a parameter set which is not likely to be chosen by a calibration engineer which makes it difficult to re-calibrate manually. Although this issue is existing for any of the tested algorithms a solution for further research could be analyzing the monotony of curves and maps to align the parameters to the domain knowledge of the calibration engineer.

A benefit of the TSO algorithm is the fact that it has the ability to adjust its hyperparameters during optimization which is illustrated in the SiL environment. Nevertheless, in Section 6.3 it is outlined that the hyperparameters can differ in each optimization after tuning if the initial state-action pairs are different (which is the case in every optimization since the initial actions are chosen randomly). This behavior does not occur in common RL algorithms since the hyperparameters are set before the optimization starts (according to the research carried out about RL algorithms by this date). Although the latter option is not beneficial (pointed out in Section 6.3) the behavior is indicating that also an optimization of the neural network can result in a misleading set of hyperparameters.

Such a behavior might be observable in the outlier of the results of the SiL environment (Section 7.2) and have to be considered especially when it is desired to transfer hyperparameters from the SiL environment to the test vehicle.

The tests carried out in the test vehicles (Section 7.3) were promising and indicate a good generalization of the TSO algorithm regarding the calibration of the launch behavior of vehicles equipped with DCT. The generalization in this domain was in general observable since the TSO algorithm was applied successfully in different test environments (simplified model, SiL environment, test vehicle), for different optimization targets, for different combustion engine configurations (Diesel and Otto engines with different amounts of cylinders), for different hybrid configurations (with and without electric motor applied) and for different drivetrain configurations (FWD and AWD configuration). Anyway, the robustness tests of Section 6.6 indicate that some further investigation should be carried out.

Another disadvantage is found in the fact that each optimization step is taking more time the further the optimization proceeds since every state-action pair is learned at each step. In contrast in DQL only a certain amount (batch size) of samples is learnt at each step which are chosen randomly from the experience buffer. Beside the disadvantage regarding the evaluation time the current implementation can also lead to an overfitted neural network. Tests with an adjusted learning behavior did not lead to an increased or equal performance of the optimization. Since the TSO algorithm solved the optimization problem comparably fast (see Table 7.5) worsened optimization time is accepted and hence a maximum number of 50 iterations in the test vehicle is desired.

9 Conclusion

Although some methods have been proposed to automatize the calibration process of control units it is usually still driven by calibration engineers in a time-consuming iterative manner. As mentioned in Section 3.1 the gaining popularity of machine learning methods led to new approaches for solving existing problems. One approach to solve the optimization problem in calibration tasks is illustrated in this study with the TSO algorithm. The TSO algorithm is a hybrid solution combining techniques applied in SL with the ongoing optimization of the machine learning model in RL. Different to RL algorithms in TSO it is strived to achieve a pre-defined target within tolerance boundaries with a single step. In contrast in RL a reward is maximized within multiple steps to solve a combinatorial problem. To achieve the target the TSO algorithm has the ability to adjust its underlying neural network architecture by using Bayesian optimization based on Gaussian processes and the developing dataset. Thus, the network is not only trained to predict actions based on its experience it also re-configures itself if a lack in performance is detected (see Section 6.3). The optimization is carried out using the ReLU activation function which is determined in Section 6.4 as beneficial for the optimization.

Beside the automated optimization also the evaluation of the vehicle launch needs to be automatized. Therefore, the objectives of Chapter 4 are introduced and divided into customer objectives and discomfort objectives. The objectives are introduced to transfer subjective feelings into objective measurements. Therefore, the discomfort objectives are determined by evaluating the signal of the engine speed for negative gradients (to determine the engine speed drop Section 4.2.1) and the clutch torque for local minima (Section 4.2.2) which should optimally lead to an objective value of zero. The determination of the customer objectives is different since they are introduced to influence the driving behavior of a vehicle based on customer requirements e.g., sporty or comfortable. Therefore, to determine the customer requirements a measured value is compared to a reference value. The customer objectives are the acceleration objective (Section 4.1.1) and the reaction time objective (Section 4.1.2). Choosing the reference value and deciding if it e.g. leads to a comfortable or sporty launch is driven by the test subject study of He et al. [3].

Compared to the other tested algorithms of Section 7.2 the TSO algorithm sets a new benchmark in scope of optimizing calibration parameters of control units such as the TCU for DCT in this study. With the fast achievement of successful results, it is shown that the TSO algorithm has the ability to be implemented into existing development processes. Therefore, the TSO algorithm without hyperparameter optimization can be used directly during the development in test vehicles (e.g. during test trips) and is not limited to optimization procedures on test benches or in simulations.

Further the TSO algorithm is able to be deployed not only in different types of test environments but also in different vehicle settings. The algorithm found a suitable calibration successfully in vehicles with an AWD and vehicles with a FWD. Also, the combustion engine type did not limit the TSO algorithm and successful calibrations could be found for vehicles with Diesel and Otto engines. In addition, the application in modern hybrid vehicles was possible since some of the test vehicles were equipped with the HDT transmission (see Section 2.4.3). Also, different optimization targets have been reached in case the system behavior of the corresponding test environment in general was able to reach these targets. The fact that the TSO algorithm is able to find calibration values to positively influence the launch behavior of vehicles equipped with DCT indicates a good

generalization by learning the system behavior with the neural network despite multiple differences in the test set up. Therefore, the TSO algorithm has the ability to increase the efficiency of the time spent in expensive test vehicles and possibly reduce costs during development by simplifying the work of calibration engineers.

Bibliography

- [1] M. Schmiedt, A. Pawlenka, und S. Rinderknecht, „AI-based parameter optimization method - applied for vehicles with dual clutch transmissions“, in *22. Internationales Stuttgarter Symposium*, M. Bargende, H.-C. Reuss, und A. Wagner, Hrsg., in Proceedings, vol. 2. Wiesbaden: Springer Fachmedien, 2022, S. 337–353.
- [2] M. Schmiedt, P. He, und S. Rinderknecht, „Target State Optimization: Drivability Improvement for Vehicles with Dual Clutch Transmissions“, *Appl. Sci.*, Bd. 12, Nr. 20, Art. Nr. 20, Okt. 2022, doi: 10.3390/app122010283.
- [3] P. He, E. Kraft, und S. Rinderknecht, „Objektivierung subjektiver Kriterien für die Bewertung von Anfahrvorgängen“, in *Digital-Fachtagung VDI-MECHATRONIK 2022*, T. Bertram, B. Corves, K. Janschek, und S. Rinderknecht, Hrsg., Darmstadt, Germany: Universitäts- und Landesbibliothek Darmstadt, 2022. doi: 0.26083/tuprints-00020963.
- [4] S. Pischinger und U. Seiffert, *Vieweg Handbuch Kraftfahrzeugtechnik*, 8. Aufl. in ATZ/MTZ-Fachbuch. Wiesbaden: Springer Vieweg, 2016. [Online]. Verfügbar unter: https://www.springer.com/de/book/9783658095277?gclid=Cj0KCQiA6t6ABhDMARIsAONIYyytYWWXz-PAYlci4L76IEFhbz_UWjHA0prVGIhnOcX3Ybwu3rcr8RHEaAu3NEALw_wcB
- [5] R. Fischer, F. Küçükay, G. Jürgens, und B. Pollak, *Das Getriebebuch*, 2. Aufl. in Der Fahrzeugantrieb. Wiesbaden: Springer Vieweg, 2016. [Online]. Verfügbar unter: https://www.springer.com/de/book/9783658131036?gclid=Cj0KCQiA6t6ABhDMARIsAONIYyz-IXWr4s0zFx05RcfDXhFGcnNoYZObGOB9Q7yeNvZrT84_ssGINYaAi4LEALw_wcB
- [6] Ž. M. Bulatović, M. V. Tomić, D. M. Knežević, und M. R. Cvetić, „Evaluation of variable mass moment of inertia of the piston–crank mechanism of an internal combustion engine“, *Proc. Inst. Mech. Eng. Part J. Automob. Eng.*, Bd. 225, Nr. 5, S. 687–702, Mai 2011, doi: 10.1177/2041299110394918.
- [7] T. S. Schmidt und S. Sewerin, „Technology as a driver of climate and energy politics“, *Nat. Energy*, Bd. 2, Nr. 6, Art. Nr. 6, Mai 2017, doi: 10.1038/nenergy.2017.84.
- [8] T. Gersdorf, P. Hertzke, P. Schaufuss, und S. Schenk, „McKinsey Electric Vehicle Index: EV Market Trends & Sales | McKinsey“, *Automot. Assem. Pract.*, S. 12, 2020.
- [9] E. Köhler und R. Flierl, *Verbrennungsmotoren: motormechanik, berechnung und auslegung des hubkolbenmotors*, 6. Aufl. in ATZ/MTZ-Fachbuch. Wiesbaden: Springer Fachmedien Wiesbaden GmbH, 2011. [Online]. Verfügbar unter: <https://www.springer.com/de/book/9783834883094>
- [10] G. H. Neugebauer, *Kräfte in den Triebwerken schnellaufender Kolbenkraftmaschinen*, 2. Aufl. in

Konstruktionsbücher, no. 2. Berlin Heidelberg: Springer-Verlag Berlin Heidelberg, 1952. [Online]. Verfügbar unter: <https://www.springer.com/de/book/9783662220719>

[11] R. van Basshuysen und F. Schäfer, Hrsg., *Handbuch Verbrennungsmotor*. Wiesbaden: Springer Fachmedien, 2017. doi: 10.1007/978-3-658-10902-8.

[12] H. Naunheimer, B. Bertsche, J. Ryborz, W. Novak, und P. Fietkau, *Fahrzeuggetriebe: Grundlagen, Auswahl, Auslegung und Konstruktion*, 3. Aufl. Berlin Heidelberg: Springer Vieweg, 2019. [Online]. Verfügbar unter: https://www.springer.com/de/book/9783662588826?gclid=Cj0KCQiA6t6ABhDMARIsAONiYyx282U87enR66AfoJ8UYU8SNf7lROn2pf53z-_AYAcC8-a0wHSxsuEaAl3-EALw_wcB

[13] H. Kerle, R. Pittschellis, und B. Corves, *Einführung in die Getriebelehre*. Wiesbaden: Teubner, 2007. doi: 10.1007/978-3-8351-9082-5.

[14] B. Gao, Q. Liang, Y. Xiang, L. Guo, und H. Chen, „Gear ratio optimization and shift control of 2-speed I-AMT in electric vehicle“, *Mech. Syst. Signal Process.*, Bd. 50–51, S. 615–631, Jan. 2015, doi: 10.1016/j.ymssp.2014.05.045.

[15] P. J. Kollmeyer, J. D. McFarland, und T. M. Jahns, „Comparison of class 2a truck electric vehicle drivetrain losses for single- and two-speed gearbox systems with IPM traction machines“, in *2015 IEEE International Electric Machines & Drives Conference (IEMDC)*, Mai 2015, S. 1501–1507. doi: 10.1109/IEMDC.2015.7409261.

[16] O. F. Vynakov, E. V. Savolova, und A. I. Skrynnyk, „MODERN ELECTRIC CARS OF TESLA MOTORS COMPANY“, *Autom. Technol. Bus. Process.*, Bd. 8, Nr. 2, Aug. 2016, doi: 10.15673/atbp.v8i2.162.

[17] G. Sieklucki, „An Investigation into the Induction Motor of Tesla Model S Vehicle“, in *2018 International Symposium on Electrical Machines (SME)*, Juni 2018, S. 1–6. doi: 10.1109/ISEM.2018.8442648.

[18] F. Di Nicola, A. Sorniotti, T. Holdstock, F. Viotto, und S. Bertolotto, „Optimization of a Multiple-Speed Transmission for Downsizing the Motor of a Fully Electric Vehicle“, *SAE Int. J. Altern. Powertrains*, Bd. 1, Nr. 1, S. 134–143, 2012.

[19] L. Nicoletti, F. Ostermann, M. Heinrich, A. Stauber, X. Lin, und M. Lienkamp, „Topology analysis of electric vehicles, with a focus on the traction battery“, *Forsch. Im Ingenieurwesen*, 2020, doi: 10.1007/s10010-020-00422-1.

[20] M. Kebriaei, A. H. Niasar, und B. Asaei, „Hybrid electric vehicles: An overview“, in *2015 International Conference on Connected Vehicles and Expo (ICCVE)*, Okt. 2015, S. 299–305. doi: 10.1109/ICCVE.2015.84.

[21] C. C. Chan, A. Bouscayrol, und K. Chen, „Electric, Hybrid, and Fuel-Cell Vehicles: Architectures and Modeling“, *IEEE Trans. Veh. Technol.*, Bd. 59, Nr. 2, S. 589–598, Feb. 2010, doi: 10.1109/TVT.2009.2033605.

- [22] K. Reif, K.-E. Noreikat, und K. Borgeest, Hrsg., *Kraftfahrzeug-Hybridantriebe*. Wiesbaden: Vieweg+Teubner Verlag, 2012. doi: 10.1007/978-3-8348-2050-1.
- [23] O. Dingel u. a., „Benchmarking Hybrid Concepts: On-Line vs. Off-Line Fuel Economy Optimization for Different Hybrid Architectures“, *SAE Int. J. Altern. Powertrains*, Bd. 2, S. 456–470, Mai 2013, doi: 10.4271/2013-24-0084.
- [24] H. Tschöke, P. Gutzmer, und T. Pfund, Hrsg., *Elektrifizierung des Antriebsstrangs: Grundlagen - vom Mikro-Hybrid zum vollelektrischen Antrieb*. Berlin, Heidelberg: Springer, 2019. doi: 10.1007/978-3-662-60356-7.
- [25] J. Jing, Y. Liu, J. Wu, W. Huang, B. Zuo, und G. Yang, „Research on drivability control in P2.5 hybrid system“, *Energy Rep.*, Bd. 7, S. 1582–1593, Nov. 2021, doi: 10.1016/j.egy.2021.09.065.
- [26] S. Kahlbau, „Mehrkriterielle Optimierung des Schaltablaufs von Automatikgetrieben“, BTU Cottbus - Senftenberg, Cottbus, 2013. [Online]. Verfügbar unter: <https://opus4.kobv.de/opus4-btu/frontdoor/index/index/year/2013/docId/2751>
- [27] B. Matthes, „Dual Clutch Transmissions - Lessons Learned and Future Potential“, in *SAE 2005 Transactions Journal of Engines*, JSTOR, 2005, S. 941–952. doi: 10.4271/2005-01-1021.
- [28] Z. Sun und K. Hebbale, „Challenges and opportunities in automotive transmission control“, in *Proceedings of the 2005, American Control Conference, 2005.*, Evanston, IL, Piscataway, N.J.: IEEE, 2005, S. 3284–3289. doi: 10.1109/ACC.2005.1470477.
- [29] U. C. Blessing, J. Meissner, M. Schweiher, und T. Hoffmeister, „Skalierbares Hybrides Doppelkupplungsgetriebe“, *ATZ - Automob. Z.*, Bd. 116, Nr. 12, S. 12–17, Dez. 2014, doi: 10.1007/s35148-014-2009-3.
- [30] U. Blessing und J. Roth-Stielow, „Possibilities of hybridization of a double clutch system - a comparing evaluation; Hybridisierungsmöglichkeiten des Doppelkupplungs-getriebes - eine vergleichende Bewertung“, Juli 2008, Zugegriffen: 19. März 2023. [Online]. Verfügbar unter: <https://www.osti.gov/etdweb/biblio/21234595>
- [31] J. Gindele und M. Diehl, „Systemansatz für einen dedizierten Hybridantrieb“, *ATZ - Automob. Z.*, Bd. 121, Nr. 5, S. 44–51, Mai 2019, doi: 10.1007/s35148-019-0038-7.
- [32] K. Jankov, „Beitrag zur automatisierten Steuerkennfeld-Applikation bei Fahrzeug-Dieselmotoren“, TU Berlin, Berlin, 2008. [Online]. Verfügbar unter: <https://depositonce.tu-berlin.de/handle/11303/2252>
- [33] N. M. Morris, *Microprocessor and Microcomputer Technology*. London: Palgrave, 1981. [Online]. Verfügbar unter: <https://www.springerprofessional.de/microprocessor-and-microcomputer-technology/7122160>
- [34] I. Susnea und M. Mitescu, *Microcontrollers in Practice*, 1. Aufl., Bd. 18. in Springer Series in Advanced Microelectronics, vol. 18. Berlin Heidelberg: Springer-Verlag, 2005. doi: 10.1007/3-540-28308-0.

- [35] P. Cappelletti, C. Golla, P. Olivo, und E. Zanoni, Hrsg., *Flash Memories*, 1. Aufl. New York: Springer US, 2013. doi: 10.1007/978-1-4615-5015-0.
- [36] H. Wallentowitz und K. Reif, Hrsg., *Handbuch Kraftfahrzeugelektronik: Grundlagen - Komponenten - Systeme - Anwendungen*, 2. Aufl. in ATZ/MTZ-Fachbuch. Wiesbaden: Vieweg+Teubner Verlag, 2011. Zugegriffen: 1. Februar 2021. [Online]. Verfügbar unter: <https://www.springer.com/de/book/9783834807007>
- [37] H. Huang, „Model-based calibration of automated transmissions“, TU Berlin, Berlin, 2016. [Online]. Verfügbar unter: <https://pdfs.semanticscholar.org/7b9f/7ca311a304de065e05121958e3ade249c1a0.pdf>
- [38] A. Dutta u. a., „Model-based and model-free learning strategies for wet clutch control“, *Mechatronics*, Bd. 24, Nr. 8, S. 1008–1020, Dez. 2014, doi: 10.1016/j.mechatronics.2014.03.006.
- [39] K. Wehbi, D. Bestle, und J. Beilharz, „Automatic calibration process for optimal control of clutch engagement during launch“, *Mech. Based Des. Struct. Mach.*, Bd. 45, Nr. 4, S. 507–522, Okt. 2017, doi: 10.1080/15397734.2016.1250221.
- [40] K. van Berkel, T. Hofman, A. Serrarens, und M. Steinbuch, „Fast and smooth clutch engagement control for dual-clutch transmissions“, *Control Eng. Pract.*, Bd. 22, S. 57–68, Jan. 2014, doi: 10.1016/j.conengprac.2013.09.010.
- [41] J. Liao, *Generische automatische Applikation für die Vorentwicklung von Hybridgetrieben in Rapid-Prototyping-Umgebung*. in Wissenschaftliche Reihe Fahrzeugtechnik Universität Stuttgart. Wiesbaden: Springer Fachmedien, 2022. doi: 10.1007/978-3-658-36814-2.
- [42] J. Hudson u. a., „Iterative Model and Trajectory Refinement for Launch Optimization of Automotive Powertrains“, in *IFAC Proceedings Volumes*, in 15th IFAC Workshop on Control Applications of Optimization, vol. 45. Rimini, Jan. 2012, S. 146–151. doi: 10.3182/20120913-4-IT-4027.00010.
- [43] Z. Zhao, L. He, Z. Zheng, Y. Yang, und C. Wu, „Self-adaptive optimal control of dry dual clutch transmission (DCT) during starting process“, *Mech. Syst. Signal Process.*, Bd. 68–69, S. 504–522, Feb. 2016, doi: 10.1016/j.ymsp.2015.06.012.
- [44] G. Pinte, B. Depraetere, W. Symens, J. Swevers, und P. Sas, „Iterative learning control for the filling of wet clutches“, *Mech. Syst. Signal Process.*, Bd. 24, Nr. 7, S. 1924–1937, Okt. 2010, doi: 10.1016/j.ymsp.2010.05.016.
- [45] A. C. Van Der Heijden, A. F. A. Serrarens, M. K. Camlibel, und H. Nijmeijer, „Hybrid optimal control of dry clutch engagement“, *Int. J. Control*, Bd. 80, Nr. 11, S. 1717–1728, Nov. 2007, doi: 10.1080/00207170701473995.
- [46] J. Horn, J. Bamberger, P. Michau, und S. Pindl, „Flatness-based clutch control for automated manual transmissions“, *Control Eng. Pract.*, Bd. 11, Nr. 12, S. 1353–1359, Dez. 2003, doi: 10.1016/S0967-0661(03)00099-6.

- [47] P. Dolcini, H. Bechart, und C. C. de Wit, „Observer-based optimal control of dry clutch engagement“, in *Proceedings of the 44th IEEE Conference on Decision and Control*, Seville, Dez. 2005, S. 440–445. doi: 10.1109/CDC.2005.1582195.
- [48] T. Naumann, „Wissensbasierte Optimierungsstrategien für elektronische Steuergeräte an Common-Rail-Dieselmotoren“, TU Berlin, Berlin, 2002. Zugegriffen: 1. Februar 2021. [Online]. Verfügbar unter: <https://depositonce.tu-berlin.de/handle/11303/851>
- [49] A. Hagerodt, *Automatisierte Optimierung des Schaltkomforts von Automatikgetrieben*, 1., Edition. Aachen: Shaker, 2003.
- [50] J. Schmidt, M. R. G. Marques, S. Botti, und M. A. L. Marques, „Recent advances and applications of machine learning in solid-state materials science“, *Npj Comput. Mater.*, Bd. 5, Nr. 1, Art. Nr. 1, Aug. 2019, doi: 10.1038/s41524-019-0221-0.
- [51] Y. LeCun, Y. Bengio, und G. Hinton, „Deep learning“, *Nature*, Bd. 521, Nr. 7553, S. 436–444, Mai 2015, doi: 10.1038/nature14539.
- [52] V. Mnih u. a., „Playing Atari with Deep Reinforcement Learning“, *ArXiv13125602 Cs*, Dez. 2013, Zugegriffen: 8. Dezember 2021. [Online]. Verfügbar unter: <http://arxiv.org/abs/1312.5602>
- [53] R. S. Sutton und A. G. Barto, *Reinforcement Learning: An Introduction*, 2. Aufl. in Adaptive Computation and Machine Learning series. Cambridge, MA, USA: A Bradford Book, 2018.
- [54] D. Michie und R. A. Chambers, „BOXES: An experiment in adaptive control“, *Mach. Intell.*, Bd. 2, Nr. 2, S. 137–152, 1968.
- [55] Y. Li, „Deep Reinforcement Learning: An Overview“. arXiv, 25. November 2018. doi: 10.48550/arXiv.1701.07274.
- [56] V. Mnih u. a., „Human-level control through deep reinforcement learning“, *Nature*, Bd. 518, Nr. 7540, S. 529–533, Feb. 2015, doi: 10.1038/nature14236.
- [57] L.-J. Lin, „Reinforcement learning for robots using neural networks“, phd, Carnegie Mellon University, USA, 1992. [Online]. Verfügbar unter: <https://apps.dtic.mil/sti/pdfs/ADA261434.pdf>
- [58] T. P. Lillicrap u. a., „Continuous control with deep reinforcement learning“. arXiv, 2015. doi: 10.48550/arXiv.1509.02971.
- [59] I. Bello, H. Pham, Q. V. Le, M. Norouzi, und S. Bengio, „Neural Combinatorial Optimization with Reinforcement Learning“, *ArXiv161109940 Cs Stat*, Jan. 2017, Zugegriffen: 9. Dezember 2021. [Online]. Verfügbar unter: <http://arxiv.org/abs/1611.09940>
- [60] L. M. Gambardella und M. Dorigo, „Ant-Q: A Reinforcement Learning approach to the traveling salesman problem“, in *Machine Learning Proceedings 1995*, A. Prieditis und S. Russell, Hrsg., San Francisco (CA): Morgan Kaufmann, 1995, S. 252–260. doi: 10.1016/B978-1-55860-377-6.50039-6.

- [61] L. Xiaohui, G. Bingzhao, und C. Hong, „Q-learning based adaptive PID controller design for AMT clutch engagement during start-up process“, in *Proceedings of the 31st Chinese Control Conference*, Juli 2012, S. 3131–3136.
- [62] M. Gagliolo u. a., „Policy search reinforcement learning for automatic wet clutch engagement“, in *15th International Conference on System Theory, Control and Computing*, Okt. 2011, S. 1–6.
- [63] K. Van Vaerenbergh u. a., „Improving wet clutch engagement with reinforcement learning“, in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, Juni 2012, S. 1–8. doi: 10.1109/IJCNN.2012.6252825.
- [64] T. Brys, K. V. Moffaert, K. V. Vaerenbergh, und A. Nowé, „On the Behaviour of Scalarization Methods for the Engagement of a Wet Clutch“, in *2013 12th International Conference on Machine Learning and Applications*, Dez. 2013, S. 258–263. doi: 10.1109/ICMLA.2013.52.
- [65] A. Lampe, C. Gühmann, R. Serway, und L. G. Siestrup, „Artificial Intelligence in Transmission Control Clutch Engagement with Reinforcement Learning“, *VDI-Berichte*, Bd. 2354, S. 899–918, 2019.
- [66] W. Genders und S. Razavi, „Using a Deep Reinforcement Learning Agent for Traffic Signal Control“, *ArXiv161101142 Cs*, Nov. 2016, Zugegriffen: 10. Dezember 2021. [Online]. Verfügbar unter: <http://arxiv.org/abs/1611.01142>
- [67] D. E. Goldberg, „Genetic and evolutionary algorithms come of age“, *Commun. ACM*, Bd. 37, Nr. 3, S. 113–120, März 1994, doi: <https://doi.org/10.1145/175247.175259>.
- [68] E. Zitzler und L. Thiele, „Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach“, *IEEE Trans. Evol. Comput.*, Bd. 3, Nr. 4, S. 257–271, Nov. 1999, doi: 10.1109/4235.797969.
- [69] K. Deb, A. Pratap, S. Agarwal, und T. Meyarivan, „A fast and elitist multiobjective genetic algorithm: NSGA-II“, *IEEE Trans. Evol. Comput.*, Bd. 6, Nr. 2, S. 182–197, Apr. 2002, doi: 10.1109/4235.996017.
- [70] C. Darwin und W. F. Bynum, *The origin of species by means of natural selection: or, the preservation of favored races in the struggle for life*. New York: AL Burt New York, 1894.
- [71] T. Bäck und H. Schwefel, „An Overview of Evolutionary Algorithms for Parameter Optimization“, *Evol. Comput.*, Bd. 1, Nr. 1, S. 1–23, März 1993, doi: 10.1162/evco.1993.1.1.1.
- [72] H. Spencer, *A System of Synthetic Philosophy: The principles of biology. I*. New York: D. Appleton, 1873.
- [73] D. H. Reed und R. Frankham, „Correlation between Fitness and Genetic Diversity“, *Conserv. Biol.*, Bd. 17, Nr. 1, S. 230–237, 2003, doi: <https://doi.org/10.1046/j.1523-1739.2003.01236.x>.
- [74] R. Knippers, *Molekulare Genetik*. Stuttgart: Georg Thieme Verlag, 2006. [Online]. Verfügbar unter:

https://www.thieme.de/shop/Biologie/Nordheim-Knippers-Droege-Meister-Schiebel-Molekulare-Genetik-9783132426375/p/000000000139920111?utm_campaign=semwhoosh-2021&utm_source=google&utm_medium=cpc&utm_content=21k1h7_2114pv_21ows5&gclid=EAIaIQobChMI4Z_O0NjJ7gIVRertCh1AiwZuEAQYASABeg-JstPD_BwE

[75] W. Janning und E. Knust, *Genetik: Allgemeine Genetik - Molekulare Genetik - Entwicklungsgenetik*, 2. Edition. Stuttgart, New York: Thieme, 2008. [Online]. Verfügbar unter: https://books.google.de/books?hl=de&lr=&id=6hWjDWnYAbQC&oi=fnd&pg=PR1&dq=Genetik:+Allgemeine+Genetik+-+Molekulare+Genetik+-+Entwicklungsgenetik&ots=WE06_-_Y6h&sig=klWRLH3CfsEszSy8Z8OReDcK2WA#v=onepage&q=Genetik%3A%20Allgemeine%20Genetik%20-%20Molekulare%20Genetik%20-%20Entwicklungsgenetik&f=false

[76] M. J. McDonald, D. P. Rice, und M. M. Desai, „Sex speeds adaptation by altering the dynamics of molecular evolution“, *Nature*, Bd. 531, Nr. 7593, Art. Nr. 7593, März 2016, doi: 10.1038/nature17143.

[77] M. Mitchell, *An Introduction to Genetic Algorithms*. 1998. doi: 10.7551/mitpress/3927.001.0001.

[78] O. Kramer, „Genetic Algorithms“, in *Genetic Algorithm Essentials*, O. Kramer, Hrsg., in Studies in Computational Intelligence. , Cham: Springer International Publishing, 2017, S. 11–19. doi: 10.1007/978-3-319-52156-5_2.

[79] B. Gadhvi, V. Savsani, und V. Patel, „Multi-Objective Optimization of Vehicle Passive Suspension System Using NSGA-II, SPEA2 and PESA-II“, *Procedia Technol.*, Bd. 23, S. 361–368, Jan. 2016, doi: 10.1016/j.protcy.2016.03.038.

[80] A. Koziolok, H. Koziolok, S. Becker, und R. H. Reussner, „Automatically improve software architecture models for performance, reliability, and cost using evolutionary algorithms“, *undefined*, 2010, Zugriffen: 7. Dezember 2021. [Online]. Verfügbar unter: <https://www.semanticscholar.org/paper/Automatically-improve-software-architecture-models-Koziolok-Koziolok/776d2d0865265eed93f35ce1cdb9d1a13577688f>

[81] S. Kahlbau und D. Bestle, „Optimal Shift Control for Automatic Transmission“, *Mech. Based Des. Struct. Mach.*, Bd. 41, Nr. 3, S. 259–273, Apr. 2013, doi: 10.1080/15397734.2012.756719.

[82] C. Desai, „Design and Optimization of Hybrid Electric Vehicle Drivetrain and Control Strategy Parameters Using Evolutionary Algorithms“, masters, Concordia University, 2010. Zugriffen: 7. Dezember 2021. [Online]. Verfügbar unter: <https://spectrum.library.concordia.ca/id/eprint/7496/>

[83] M. Bachinger, B. J. Knauder, und M. Stolz, „Automotive vehicle launch optimization based on differential evolution (DE) approach for increased driveability“, in *International Conference on Engineering Optimization*, Rio de Janeiro. , 2012, S. 1–12. Zugriffen: 1. Februar 2021. [Online]. Verfügbar unter: <https://graz.pure.elsevier.com/de/publications/automotive-vehicle-launch-optimization-based-on-differential-evol>

- [84] Y. Zhong, B. Wyns, R. De Keyser, G. Pinte, und J. Stoev, „An implementation of genetic-based learning classifier system on a wet clutch system“, in *Applied Stochastic Models and Data Analysis Conference, 14th, Proceedings*, Rome, 2011, S. 1431–1439. Zugegriffen: 1. Februar 2021. [Online]. Verfügbar unter: <http://hdl.handle.net/1854/LU-1996820>
- [85] S. Zaglauer, „Methode zur multikriteriellen Optimierung des Motorverhaltens anhand physikalisch motivierter Modelle“, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, 2014. Zugegriffen: 1. Februar 2021. [Online]. Verfügbar unter: <https://opus4.kobv.de/opus4-fau/frontdoor/index/index/docId/5248>
- [86] A. Piszcz und T. Soule, „Genetic programming: optimal population sizes for varying complexity problems“, in *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, in GECCO '06. New York, NY, USA: Association for Computing Machinery, Juli 2006, S. 953–954. doi: 10.1145/1143997.1144166.
- [87] S. Katoch, S. S. Chauhan, und V. Kumar, „A review on genetic algorithm: past, present, and future“, *Multimed. Tools Appl.*, Bd. 80, Nr. 5, S. 8091–8126, Feb. 2021, doi: 10.1007/s11042-020-10139-6.
- [88] S.-F. Hwang und R. He, „A hybrid real-parameter genetic algorithm for function optimization“, *Adv Eng Inform.*, 2006, doi: 10.1016/j.aei.2005.09.001.
- [89] A. Hassanat, K. Almohammadi, E. Alkafaween, E. Abunawas, A. Hammouri, und V. B. S. Prasath, „Choosing Mutation and Crossover Ratios for Genetic Algorithms—A Review with a New Dynamic Approach“, *Information*, Bd. 10, Nr. 12, Art. Nr. 12, Dez. 2019, doi: 10.3390/info10120390.
- [90] D. Simon, „Entwicklung eines effizienten Verfahrens zur Bewertung des Anfahrverhaltens von Fahrzeugen“, Universität Rostock, Rostock, 2010. [Online]. Verfügbar unter: http://rosdok.uni-rostock.de/file/rosdok_disshab_0000000705/rosdok_derivate_0000004687/Dissertation_Simon_2011.pdf
- [91] E. Kraft, A. Viehmann, P. Erler, und S. Rinderknecht, „Virtuelle Fahrerproben von Antriebssystemen im Fahrsimulator“, *ATZ-Automob. Z.*, Bd. 123, Nr. 3, S. 42–47, 2021.
- [92] H. Bellem, T. Schönenberg, J. F. Krems, und M. Schrauf, „Objective metrics of comfort: Developing a driving style for highly automated vehicles“, *Transp. Res. Part F Traffic Psychol. Behav.*, Bd. 41, S. 45–54, Aug. 2016, doi: 10.1016/j.trf.2016.05.005.
- [93] M. Elbanhawi, M. Simic, und R. Jazar, „In the passenger seat: investigating ride comfort measures in autonomous cars“, *IEEE Intell. Transp. Syst. Mag.*, Bd. 7, Nr. 3, S. 4–17, 2015.
- [94] K. Nowatschin u. a., „Multitronic—das neue Automatikgetriebe von Audi“, *ATZ-Automob. Z.*, Bd. 102, Nr. 9, S. 746–753, 2000.
- [95] C. Hirzel, „Ein Beitrag zur Synthese und Analyse elektrifizierter Fahrzeuggetriebebestrukturen aus einer

Kombination von Stirnrad- und Planetengetrieben mit Fokus auf die systematische Realisierung einer hinreichenden Gangverteilung“, Otto-von-Guericke-Universität Magdeburg, Magdeburg, 2018. [Online]. Verfügbar unter: https://opendata.uni-halle.de/bitstream/1981185920/13503/1/Hirzel_Cathleen_Dissertation_2018.pdf

[96] L. L. Hoberock, „A survey of longitudinal acceleration comfort studies in ground transportation vehicles“, University of Texas at Austin, Austin, Texas, 40, 1976. [Online]. Verfügbar unter: https://repositories.lib.utexas.edu/bitstream/handle/2152/20856/cats_rr_40.pdf?sequence=2

[97] G. Deng und L. W. Cahill, „An adaptive Gaussian filter for noise reduction and edge detection“, in *1993 IEEE Conference Record Nuclear Science Symposium and Medical Imaging Conference*, San Francisco: IEEE, Okt. 1993, S. 5. doi: 10.1109/NSSMIC.1993.373563.

[98] A. Junghanns, J. Mauss, und M. Seibt, „Faster Development of AUTOSAR compliant ECUs through simulation“, gehalten auf der Embedded Real Time Software and Systems (ERTS2014), Toulouse, France, 2014, S. 5. [Online]. Verfügbar unter: <https://hal.archives-ouvertes.fr/hal-02272183>

[99] Synopsys, „Silver – Virtual ECU | Synopsys Verification“. Zugegriffen: 6. August 2023. [Online]. Verfügbar unter: <https://www.synopsys.com/verification/virtual-prototyping/virtual-ecu/silver.html>

[100] J. H. Holland, „Genetic Algorithms“, *Sci. Am.*, Bd. 267, Nr. 1, S. 66–73, 1992.

[101] S. Ishii, W. Yoshida, und J. Yoshimoto, „Control of exploitation–exploration meta-parameter in reinforcement learning“, *Neural Netw.*, Bd. 15, Nr. 4, S. 665–687, Juni 2002, doi: 10.1016/S0893-6080(02)00056-4.

[102] P. Sibi, S. Jones, und P. Siddarth, „Analysis of different activation functions using back propagation neural networks“, *J. Theor. Appl. Inf. Technol.*, Bd. 47, Nr. 3, S. 1264–1268, 2005.

[103] A. Koutsoukas, K. J. Monaghan, X. Li, und J. Huan, „Deep-learning: investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data“, *J. Cheminformatics*, Bd. 9, Nr. 1, S. 42, Juni 2017, doi: 10.1186/s13321-017-0226-y.

[104] L. Yang und A. Shami, „On hyperparameter optimization of machine learning algorithms: Theory and practice“, *Neurocomputing*, Bd. 415, S. 295–316, Nov. 2020, doi: 10.1016/j.neucom.2020.07.061.

[105] C. M. Bishop, „Neural networks and their applications“, *Rev. Sci. Instrum.*, Bd. 65, Nr. 6, S. 1803–1832, Juni 1994, doi: 10.1063/1.1144830.

[106] Y. Ozaki, M. Yano, und M. Onishi, „Effective hyperparameter optimization using Nelder-Mead method in deep learning“, *IPSJ Trans. Comput. Vis. Appl.*, Bd. 9, Nr. 1, S. 20, Nov. 2017, doi: 10.1186/s41074-017-0030-7.

[107] F. Pedregosa, „Hyperparameter optimization with approximate gradient“, in *Proceedings of The 33rd International Conference on Machine Learning*, PMLR, Juni 2016, S. 737–746. Zugegriffen: 14. Dezember 2021. [Online]. Verfügbar unter: <https://proceedings.mlr.press/v48/pedregosa16.html>

-
- [108]I. Goodfellow, Y. Bengio, und A. Courville, *Deep learning*. Cambridge, Massachusetts: MIT press, 2016.
- [109]C. E. Rasmussen, „Gaussian Processes in Machine Learning“, in *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2 - 14, 2003, Tübingen, Germany, August 4 - 16, 2003, Revised Lectures*, O. Bousquet, U. von Luxburg, und G. Rätsch, Hrsg., in Lecture Notes in Computer Science. , Berlin, Heidelberg: Springer, 2004, S. 63–71. doi: 10.1007/978-3-540-28650-9_4.
- [110]Y. Wang, Y. Li, Y. Song, und X. Rong, „The Influence of the Activation Function in a Convolution Neural Network Model of Facial Expression Recognition“, *Appl. Sci.*, Bd. 10, Nr. 5, Art. Nr. 5, Jan. 2020, doi: 10.3390/app10051897.
- [111]J. Schmidt-Hieber, „Nonparametric regression using deep neural networks with ReLU activation function“, *Ann. Stat.*, Bd. 48, Nr. 4, S. 1875–1897, Aug. 2020, doi: 10.1214/19-AOS1875.
- [112]P. M. Radiuk, „Impact of Training Set Batch Size on the Performance of Convolutional Neural Networks for Diverse Datasets“, *Inf. Technol. Manag. Sci.*, Bd. 20, Nr. 1, Jan. 2017, doi: 10.1515/itms-2017-0003.
- [113]N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, und P. T. P. Tang, „On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima“, *ArXiv160904836 Cs Math*, Feb. 2017, Zugegriffen: 14. Dezember 2021. [Online]. Verfügbar unter: <http://arxiv.org/abs/1609.04836>
- [114]H. Kitano, „Biological robustness“, *Nat. Rev. Genet.*, Bd. 5, Nr. 11, S. 826–837, Nov. 2004, doi: 10.1038/nrg1471.
- [115]J. Schulman, F. Wolski, P. Dhariwal, A. Radford, und O. Klimov, „Proximal Policy Optimization Algorithms“. arXiv, 28. August 2017. doi: 10.48550/arXiv.1707.06347.
- [116]V. Mnih u. a., „Asynchronous Methods for Deep Reinforcement Learning“. arXiv, 16. Juni 2016. doi: 10.48550/arXiv.1602.01783.
- [117]T. Haarnoja, A. Zhou, P. Abbeel, und S. Levine, „Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor“. arXiv, 8. August 2018. doi: 10.48550/arXiv.1801.01290.