

Safe Reinforcement Learning-Based Control of Voltage-Forming Grid Inverters

**From the Faculty of Electrical Engineering, Computer Science
and Mathematics of Paderborn University**

to obtain the academic degree

Doctor of Engineering (Dr.-Ing.)

approved dissertation
by

Daniel Weber

First Reviewer: Prof. Dr.-Ing. Joachim Böcker

Second Reviewer: Prof. Dr.-Ing. Oliver Wallscheid

Date of the dissertation defense: 20 July 2026

Paderborn 2026

Diss. EIM-E/402

Acknowledgment

This work was created during my time as a research assistant at the Department of Power Electronics and Electric Drives (LEA) at Paderborn University in the context of the project Learning Algorithms for the Automatization of Intelligent Power Electronic Systems (LEA-AI), funded by the Federal Ministry of Education and Research (grant: 03VP11210).

First and foremost, I would like to thank Annika, for her constant support. She has been there for me through both difficult times and wonderful moments, which has been a great support for me. I would also like to thank my family, especially mother and my sister.

I have spent most of my working life so far at LEA, and the last few years in particular have been the best. This is only possible in an outstanding working environment, created by the people you meet every day. I would therefore like to take this opportunity to thank all my current and former colleagues at LEA. Especially, Thorsten, who supervised me during my master's thesis and subsequently motivated me to pursue a doctorate. Special thanks also go to my former office colleague Karl Stephan, Oliver and Wilhelm P. for their daily support and for helping me settle in at LEA, as well as for their technical support at the beginning, and during the, partly due to funding requirements, odyssey-like search for a topic, which has now come to a happy end. In working on this, I would like to especially thank Jarren, Maximilian, and Dominik for their technical support and for the welcome change after work, which has resulted in many good friendships for me. The same thanks go to Matti, Stefan, Marc, Benjamin, Wilhelm K. and Sören.

I would also like to thank my colleagues at IW-hall, Daniel, Michael and Michael, both for the relaxed daily atmosphere and for listening to (sometimes loud) music together, as well as Elke, who was always available to help with administrative and everyday issues.

Last but not least, I would like to thank Prof. Dr.-Ing. Joachim Böcker and Prof. Dr.-Ing. Oliver Wallscheid for supervising this work, for the academic freedom I was given during the process, and for their support.

Special thanks go to my father, Wolfgang Weber, to whom I dedicate this work.

Abstract

The increasing expansion of local generation units, such as photovoltaic systems or wind turbines, within the electric power grid is leading to a prevalence of inverter-based resources. This change is motivating the establishment of microgrids, which are inverter-based local distribution systems in which storage units allow local demand and local generation to be matched as a degree of freedom. To make optimal use of local generation capabilities and ensure high-quality supply, high demands are placed on the control engineering to provide fast reactions to transients. The requirements for accurate system modeling, the effort involved in controller design, and the necessary human expertise make conventional approaches such as PI control or optimal approaches such as model predictive control disadvantageous when they need to be scaled to a large number of individual systems. These challenges can be overcome by reinforcement learning, a data-driven optimal control approach. In reinforcement learning, a control is learned from interaction experiences with the technical system – in this thesis an inverter. Benefiting from the data-driven behavior, a model-free, self-adaptive, and largely automatic controller development process with minimum human effort can be enabled.

In order to profit from the data-driven behavior, the experiences must be gained from an interaction in a real-world application. This poses a challenge in terms of safety during learning to protect the inverter from damage. To address this challenge, this thesis extends a standard reinforcement learning algorithm with a safety component that allows inverter control to be learned safely in a real-time, real-world application without a priori parameter knowledge and human intervention. In such an experiment, safe learning of voltage control via reinforcement learning is presented with a three-phase 250kVA-capable inverter.

In a proof of concept, supplying a constant, resistive load, the safety extension is didactically introduced and validated. The reinforcement learning-based controller is compared to model predictive controllers and PI-based controllers, highlighting the benefits regarding transients, steady state, compensating for non-linearities and unbalances. Extending the safety approach and the learning experiment to arbitrary loads, enables a general use case in an electric grid application. Finally, this makes the developed data-driven voltage controller applicable in a decentralized, inverter-based microgrid. Based on learning algorithms and general design rules, this setup enables fully automatic and model-free commissioning of an inverter-based microgrid without human intervention, which is experimentally demonstrated in a 1.25 MVA-capable system.

Zusammenfassung

Die zunehmende Verbreitung dezentraler Stromversorgung führt zu einer stärkeren Verbreitung umrichterbasierter Systeme. Diese Veränderung motiviert die Etablierung von Microgrids, bei denen es sich um umrichterbasierte, lokale Energienetze handelt, in denen ein elektrischer Energiespeicher als Freiheitsgrad genutzt wird, um lokal Bedarf und Erzeugung zu synchronisieren. Um die lokalen Erzeugungskapazitäten optimal zu nutzen und eine hohe Versorgungsqualität zu gewährleisten, muss das Regelverhalten hohen Anforderungen gerecht werden. Die Anforderungen an eine genaue Systemmodellierung, der Aufwand für die Regelungsauslegung und das erforderliche Fachwissen stellen Herausforderungen bei herkömmlichen Ansätzen wie Proportional-Integral(PI)-Reglern oder optimalen Regelungsansätzen wie der modellprädiktive Regelung dar, wenn sie auf eine große Anzahl einzelner Systeme skaliert werden müssen. Diese Herausforderungen können durch Reinforcement Learning datenbasiert bewältigt werden. Beim Reinforcement Learning wird eine Regelung mit aus der Interaktion mit dem technischen System (hier Wechselrichter) gewonnenen Daten trainiert. Durch das datenbasierte Verfahren wird ein modellfreier, selbstadaptiver und weitgehend automatischer Reglerentwicklungsprozess mit minimalem menschlichem Aufwand ermöglicht.

Um vom datenbasierten Verfahren zu profitieren, müssen die Daten aus der Interaktion mit einer realen Hardwareanwendung gewonnen werden. Dies stellt eine Herausforderung hinsichtlich der Sicherheit des Wechselrichters während des Lernprozesses dar. Um dieser zu begegnen, wird in dieser Arbeit ein Standardalgorithmus für Reinforcement Learning um eine Sicherheitskomponente erweitert, was es ermöglicht, die Regelung eines Wechselrichters in einer Echtzeit-Anwendung in der realen Welt ohne vorherige Parameterkenntnisse und ohne menschliches Eingreifen sicher zu erlernen. In einem solchen Experiment wird das sichere Erlernen der Spannungsregelung durch Reinforcement Learning mit einem dreiphasigen 250-kVA-Wechselrichter vorgestellt.

Im Zuge eines Machbarkeitsnachweises, bei dem eine konstante ohmsche Last versorgt wird, wird die Sicherheitserweiterung erläutert und validiert. Der Reinforcement-Learning-Regler wird mit modellprädiktiven und PI-Reglern verglichen, wobei die Vorteile in Bezug auf Transiente, das stationäre Verhalten sowie die Kompensation von Nichtlinearitäten und Schiefasten hervorgehoben werden. Die Erweiterung von Sicherheitsansatz und Lernexperiment auf beliebige Lasten ermöglicht eine allgemeine Anwendung in elektrischen Energienetzen, wodurch der datenbasierte Spannungsregler in einem dezentralen, umrichterbasierten Microgrid anwendbar ist. Basierend auf Lernalgorithmen und allgemeinen Auslegungsregeln ermöglicht dieser Aufbau eine vollautomatische und modellfreie Inbetriebnahme, was experimentell in einem 1,25-MVA-System demonstriert wird.

Own Publications

Articles with Direct Significance to this Thesis

The content of this work is largely based upon the publications [Own1, Own2, Own3, Own4], whose main innovations and contents have been developed, investigated and documented by the author of this thesis. [Own1] covers a comparison to another method, which was developed and implemented by the second author Dominik Schmies. Reference is made to this in the relevant section of this thesis. Parts of the content of this thesis have been adapted from these publications in either modified or unmodified form, in terms of text, illustrations, data and mathematical formulas. Corresponding self-citations are not explicitly listed in this work for reasons of readability. [Own5, Own6] discuss purely simulative considerations that, although not explicitly part of this work, represent essential preliminary work.

- [Own1] D. Weber, D. Schmies, J. Lange, M. Schenke, and O. Wallscheid. “Optimal Control of Voltage Forming Grid Inverters by Model Predictive Control and Reinforcement Learning”. In: *(Under Review in IEEE Access)* (2026).
- [Own2] D. Weber, J. Lange, and O. Wallscheid. “Safe Reinforcement Learning-based Control for a Voltage Source Inverter Operating in an Unbalanced Grid”. In: *2025 IEEE Kiel PowerTech*. 2025, pp. 1–6. DOI: 10.1109/PowerTech59965.2025.11180230.
- [Own3] D. Weber, J. Lange, and O. Wallscheid. “Reinforcement Learning-Based Control of Voltage-Forming Grid Inverters With Arbitrary Loads”. In: *IEEE Journal of Emerging and Selected Topics in Industrial Electronics* (2026), pp. 1–12. DOI: 10.1109/JESTIE.2026.3654784.
- [Own4] D. Weber, J. Lange, and O. Wallscheid. “Model-Free Droop and Reinforcement Learning-Based Voltage Control of Decentralized Inverter-Based Microgrids”. In: *(Accepted for presentation at the 2026 IEEE Applied Power Electronics Conference and Exposition (APEC 2026))*. 2026.
- [Own5] D. Weber, M. Schenke, and O. Wallscheid. “Steady-State Error Compensation for Reinforcement Learning-Based Control of Power Electronic Systems”. In: *IEEE Access* 11 (2023), pp. 76524–76536. DOI: 10.1109/ACCESS.2023.3297274.
- [Own6] D. Weber, M. Schenke, and O. Wallscheid. “Safe Reinforcement Learning-Based Control in Power Electronic Systems”. In: *2023 International Conference on Future Energy Solutions (FES)*. IEEE. 2023. DOI: 10.1109/FES57669.2023.10182718.

Further Articles in the Broader Context of this Thesis

As Main Author

- [Own7] D. Weber, K. S. Stille, O. Wallscheid, and J. Böcker. “Energy Management for a Nano-CHP Unit and an Electrical Storage System in a Residential Application”. In: *2018 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)*. 2018, pp. 786–792. DOI: 10.1109/SPEEDAM.2018.8445296.
- [Own8] D. Weber, M. I. R. Jani, M. Grabo, et al. “Lifetime Extension of Photovoltaic Modules by Influencing the Module Temperature Using Phase Change Material”. In: *2018 IEEE 7th World Conference on Photovoltaic Energy Conversion (WCPEC) (A Joint Conference of 45th IEEE PVSC, 28th PVSEC, 34th EU PVSEC)*. 2018, pp. 0784–0789. DOI: 10.1109/PVSC.2018.8548115.
- [Own9] D. Weber, S. Heid, H. Bode, J. H. Lange, E. Hüllermeier, and O. Wallscheid. “Safe Bayesian Optimization for Data-Driven Power Electronics Control Design in Microgrids: From Simulations to Real-World Experiments”. In: *IEEE Access* 9 (2021), pp. 35654–35669. DOI: 10.1109/ACCESS.2021.3062144.

As Co-Author

- [Co1] F. Bause, D. Weber, J. Rautenberg, and B. Henning. “Unsicherheitsanalyse eines Vorwärtsmodells zur Simulation transienter Wellenausbreitung im Hohlzylinder”. In: VDE Verlag, 2014.
- [Co2] T. Vogt, D. Weber, O. Wallscheid, and J. Böcker. “Prediction of residual power peaks in industrial microgrids using artificial neural networks”. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. 2017, pp. 3228–3235. DOI: 10.1109/IJCNN.2017.7966259.
- [Co3] M. Grabo, D. Weber, A. Paul, et al. “Entwicklung eines thermischen 1D-Simulationsmodells zur Bestimmung der Temperaturverteilung in Solarmodulen”. de. In: *RET.Con 2019: Tagungsband : 2. Regenerative Energietechnik-Konferenz in Nordhausen 7.-8. Februar 2019* (2019). Ed. by V. Wesselak, pp. 116–129. DOI: 10.22032/dbt.46054.
- [Co4] M. Grabo, D. Weber, A. Paul, et al. “Numerical Investigation of the Temperature Distribution in PCM-integrated Solar Modules”. In: *Chemical Engineering Transactions* 76 (Oct. 2019), pp. 895–900. DOI: 10.3303/CET1976150.
- [Co5] M. Grabo, D. Weber, A. Paul, T. Klaus, W. Bärmppohl, and E. Kenig. “Numerische Untersuchung der Temperaturverteilung in PCM-integrierten Solarmodulen”. In: Frankfurt am Main, 2019.
- [Co6] S. Heid, D. Weber, H. Bode, E. Hüllermeier, and O. Wallscheid. “OMG: A Scalable and Flexible Simulation and Testing Environment Toolbox for Intelligent Microgrid Control”. In: *Journal of Open Source Software* 5.54 (2020), p. 2435. DOI: 10.21105/joss.02435.
- [Co7] K. S. Stille, D. Weber, J. Lange, T. Vogt, O. Wallscheid, and J. Böcker. “Emulation of Microgrids for Research and Validation of Control and Opera-

- tion Strategies”. In: *International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM)*. 2020, pp. 324–329. DOI: 10.1109/SPEEDAM48782.2020.9161971.
- [Co8] M. Meyer, D. Weber, V. Chidananda, et al. “ElectricGrid.jl – Automated modeling of decentralized electrical energy grids”. In: *PESS 2023; Power and Energy Student Summit*. 2023, pp. 112–117.
- [Co9] S. Boshoff, J. Stenner, D. Weber, et al. “Hybrid control of interconnected power converters using both expert-driven droop and data-driven reinforcement learning approaches”. In: *PESS 2023; Power and Energy Student Summit*. 2023, pp. 124–129.
- [Co10] O. Wallscheid, S. Peitz, J. Stenner, et al. “ElectricGrid.jl - A Julia-based modeling and simulation tool for power electronics-driven electric energy grids”. In: *Journal of Open Source Software* 8.89 (2023), p. 5616. DOI: 10.21105/joss.05616.

Nomenclature

Abbreviations

ANN	Artificial neural network
AI	Artificial intelligence
CCS	Continuous control set
DDPG	Deep deterministic policy gradient
FCS	Finite control set
FPGA	Field programmable gate array
HIL	Hardware in the loop
HP	Hyperparameter
HPO	Hyperparameter optimization
IBR	Inverter-based resources
ID	System identification
IGBT	Insulated gate bipolar transistor
LTI	Linear time-invariant
LUT	look-up table
MAE	Mean absolute error
MDP	Markov decision process
MG	Microgrid
MG-lab	Microgrid laboratory
MPC	Model predictive control
MRE	Mean root error
MSE	Mean squared error
NPC	Neutral-point-clamped
ODE	Ordinary differential equation

Nomenclature

PI	Proportional integral
PLL	Phase-locked loop
PWM	Pulse width modulation
PR	Proportional resonant
PRBS	Pseudorandom binary sequence
PV	Photovoltaic
QP	Quadratic program
RCP	Rapid control prototyping
ReLU	Rectified linear unit
RES	Renewable energy sources
RL	Reinforcement learning
RLS	Recursive least squares
SSD	Symmetrical sequence decomposition
THD	Total harmonic distortion
UDP	User Datagram Protocol
VSG	Virtual synchronous generator
VSI	Voltage-source inverter

Symbols

$\mathbf{0}_{n \times m}$	Matrix of zeros with m rows and n columns
$\mathbf{1}_{n \times m}$	Matrix of ones with m rows and n columns
$\alpha_{q/\mu}$	Step size of the optimizer used for training (learning rate)
$\beta_{q/\mu}$	Parameter of the artificial neural network's activation function
γ	Discount factor
Γ	Covariance matrix of the identified parameter estimation
δ	Parameters of the actor's function approximator
ϵ	Regression for identification
ζ	Output of the identification
$\eta_{q/\mu}$	Number of neurons per layer
θ	Phase angle
ϑ	Slack variable
ι	Forgetting factor used within the identification
λ	Noise parameter stiffness
μ	Policy function
ν	Action noise
ξ	Experience vector
Ξ	Parameter vector for the identification
ρ	Parameters of the critic's function approximator
ς	Switch to connect an inverter to the microgrid
σ	Noise parameter variance
τ	Damping factor used during training
Υ	Dropout parameter
ϕ	Feature function
φ	Angle between grid voltage and load current
χ	Parameter vector for the identification
ψ	Punishment factor for using a slack variable
ω	Vector of system constraints
ω_g	Angular frequency

$\mathbf{A}$	Dynamic matrix of a discrete-time system
$\mathbf{b}, \mathbf{B}$	Input vector/matrix of a discrete-time system
$\mathcal{B}$	Batch of experiences
$\mathbf{C}$	Output matrix of a discrete-time system
C_f	Capacitance of the filter capacitor
$\mathbf{d}$	Disturbance vector
$\mathbb{D}$	Disturbance space
$\mathbf{e}_{\text{feas}}$	Constraint vector resulting from the feasible set
$\mathbf{f}$	State transition function
f_s	Switching frequency
$\mathbb{F}_0$	State-action space
$\mathbb{F}_{\infty \rightarrow 0}$	Final feasible set
$\mathbb{F}_{6 \rightarrow 0}$	Feasible state-action space considering $k_f = 6$ future time steps
$\mathbf{g}$	Disturbance feedthrough matrix of a discrete-time system
$\mathbf{G}_{\text{feas}}$	Constraint matrix resulting from the feasible set
g	Return
$\mathbf{h}$	Disturbance input vector of a discrete-time system
$i, \mathbf{i}$	Current/current vector
$\mathbf{I}$	Identity matrix
j	Imaginary number
k	Discrete time index
k_f	Prediction horizon
$K_{p/i}$	Proportional/integral gains
$K_{P/Q}$	Droop gain
l	Number of outputs
L_f	Inductance of the filter inductor
m	Number of actions
$\mathcal{M}$	Replay Buffer
n	Number of states
o	Number of disturbances
P	Active power

q	State-action value function
Q	Reactive power
r	Reward
R_{sC}	Internal series resistance of the filter capacitor
R_{sL}	Internal series resistance of the filter inductor
$\mathbb{R}$	Set of real numbers
$\underline{s}$	Laplace variable
S	Apparent power
t	Continuous time
T_{s}	Sampling time
$\mathbf{u}$	System input
$\mathbb{U}$	Action space
$v, \mathbf{v}$	Voltage/voltage vector
$\mathcal{V}$	Stage costs
$\mathbf{w}$	Output function
$\mathbf{W}$	Matrix defining the system constraints
$\mathbf{x}$	System state
$\mathbb{X}$	State space
$\mathbf{y}$	System output
$\mathbb{Y}$	Output space
$z_{q/\mu}$	Number of layers of an ANN

Notation

x	Scalar
$ x $	Absolute value of x
$\frac{dx}{dt}$	Time derivative of x
$\underline{x}$	Complex variable
$\text{Re}(\underline{x})$	Real part of $\underline{x}$
$x(t)$	Continuous-time variable
x_k	Discrete-time variable
$\mathbf{x}$	Row vector
$\mathbf{x}^\top$	Column vector
$\ \mathbf{x}\ $	2-norm/length of $\mathbf{x}$
$\mathbf{X}$	Matrix
$\mathbf{X}^{-1}$	Inverse of $\mathbf{X}$
$\hat{x}$	Estimate of x
$\bar{x}$	Mean of x
$\tilde{\mathbf{x}}$	Featurized $\mathbf{x}$
$\mathbf{x}^*$	Reference of x
$\mathbf{x} \in \mathbb{X}$	$\mathbf{x}$ is an element of the set $\mathbb{X}$
$\mathbb{X} \subseteq \mathbb{R}^n$	$\mathbb{X}$ is a subset of the n -dimensional set $\mathbb{R}$
$x \in [a, b[$	Right-open interval $\{x \in \mathbb{R} a \leq x < b\}$
$\mathcal{X}$	Condensed matrix of $\mathbf{X}$
$\text{proj}_{\mathbb{F}_0}(\mathbb{F}_{k_f})$	Projection of k_f future time steps onto the momentary time step
Δx	Deviation of x
$\Delta_k x_k$	Difference of x_k : $\Delta_k x_k = \frac{x_k - x_{k-1}}{T_s}$
$\mathbf{f} : \mathbb{X} \rightarrow \mathbb{R}$	Function f that maps the domain $\mathbb{X}$ to the codomain $\mathbb{R}$
$\mathbb{X} \times \mathbb{U}$	Cartesian product of the sets $\mathbb{X}$ and $\mathbb{U}$

Contents

Acknowledgment	i
Abstract	iv
Zusammenfassung	vi
Own Publications	viii
Own Publications	viii
Nomenclature	xi
Nomenclature	xi
Contents	xvii
1 Introduction	1
1.1 Motivation	1
1.2 State of the Art	3
1.3 Contribution	5
2 Fundamentals	7
2.1 Optimal Control for Voltage-Forming Inverters	7
2.2 Voltage Source Inverter	8
2.3 Reinforcement Learning Fundamentals	10
2.3.1 Function Approximation in Reinforcement Learning	12
2.3.2 Exploration vs. Exploitation	13
2.4 Safeguarding in Continuous State and Action Space	14
2.4.1 Derivation of the Relevant Constraints	14
2.4.2 Example: Double Integrator	18
2.4.3 Differential Constraints	21
2.4.4 System Identification	23
2.5 Power Sharing	24
3 RL-Based Control of Voltage-Forming Grid Inverters	27
3.1 Experimental Setup	28
3.2 Digital Control Delay Compensation	31
3.3 Reward Design	31

3.4	Featurization	33
3.5	Training via Edge Computing	34
3.6	Hyperparameter Optimization in Real-Time Application	34
4	Application: Constant, Resistive Load	37
4.1	State-Space Representation of the Plant	38
4.2	Safeguard in Real Time	39
4.3	Data-Driven Safeguard	41
4.4	Hyperparameter Configuration	43
4.5	Training Behavior	48
4.6	Steady-State Behavior	48
4.7	Transient Behavior	52
4.8	Black-Start Capability	54
4.9	Operation with Active Output Constraints	56
4.10	Computing Time Evaluation	57
4.11	Unbalanced Loads	58
4.11.1	Learning Behavior	60
4.11.2	Transient Behavior	61
4.11.3	Steady-State Behavior	64
5	Application: Arbitrary Loads	67
5.1	Safeguard Extension	68
5.2	Training Behavior	69
5.3	Steady-State Behavior	72
5.4	Transient Behavior	75
5.4.1	Active Load	75
5.4.2	Passive Linear Time-Variant Load	78
5.4.3	Black-Start and Emergency Shutdown Capability	81
6	Decentralized Inverter-Based Microgrid Application	83
6.1	Training Behavior	86
6.2	Islanded Grid Experiment	87
6.2.1	P-Droop Evaluation	87
6.2.2	Q-Droop Evaluation	90
6.2.3	Non-Linear Load	91
6.2.4	Failure Event	93
7	Conclusion and Outlook	95
7.1	Conclusion	95
7.2	Outlook	96
	Appendix	99
A.1	Deep Deterministic Policy Gradient Algorithm - Basics	99
A.2	Hyperparameter Settings of the Used DDPG Agents	100
A.2.1	HPs for the Constant, Resistive Load Application	100

A.2.2 HPs for the Arbitrary Load Application	102
A.3 Predictive Dead-Beat Controller	107
A.4 Artificial Intelligence use Declaration	108
Lists	109
List of Tables	109
List of Figures	109
References	115

1 Introduction

1.1 Motivation

The increased penetration of inverter-based resources (IBR), which enables the integration of local, decentralized generation units (e.g., photovoltaic (PV) systems or wind turbines) into the electrical energy grid, offers new possibilities and more flexibility, but also comes along with new challenges. Forming local electrical energy grids, so-called microgrids (MGs) consisting of sources, storage units and loads that can be operated coupled or decoupled from the supply grid, makes it possible to match local demand and generation [1]. This, e.g., can help to relieve the transmission lines and can reduce the need for grid expansion. Additionally, MGs permit increasing stability, improving the continuity of supply and can provide a broad scope of ancillary services, like, e.g., dispatch services, black-start restoration, reactive power supply, and voltage and frequency control [2]. A successful integration that optimally utilizes the IBR, places higher demands on the control engineering at the component level to benefit from the component's reaction capability. Standard feedback controllers such as proportional integral (PI) controllers or proportional resonant (PR) controllers often perform poorly when operated close to the system's limits, which becomes apparent in low responsiveness during transients and controller wind-up-based overshoots [3]. Therefore, optimal utilization of IBRs in such cases cannot be achieved with standard feedback controllers, as a controller design that enables a fast response could lead to possible overshoots and cannot be guaranteed to not violate the physical system constraints. These constraints result from physical system limitations to avoid, e.g., overcurrents and overvoltages within the IBR's power electronic or filter components. The compliance with these constraints during the control task is safety-critical, as exceeding the limits can lead to an emergency shutdown and thus to a supply interruption or physical damage to the hardware.

Optimal control approaches, such as model predictive control (MPC), can directly take such constraints into account [4]. Utilizing a model, MPC can directly consider these limitations during the system input calculation and allows operation at the system limits without further safety margins, resulting in maximized plant performance [3]. However, highly specialized engineers and a lot of working time are required to achieve this level of performance, since it requires accurate measurements and modelling as well as

computationally efficient implementation. When applied to a real system, deriving an accurate model becomes challenging. Real-world implementation poses several general challenges:

- model inaccuracy (e.g., with regard to data sheet specifications or aging effects),
- measurement noise,
- sensor calibration mismatch,
- limited sensor bandwidth,
- inverter non-linearities,
- computational time demand, and
- the prevention of safety-critical system states.

While safety-critical system states can be handled by MPC, the previous points can be a challenge for such a model-based control approach. Since MPC uses a model to predict future system trajectories to calculate the next optimal control action based on a real-time solver solution, model inaccuracy or measurement deviations can result in sub-optimal control behavior. This is especially the case during non-linear behavior of the plant system, since MPCs are typically based on a linear model, as they must be configured in real-time applications taking into account the trade-off between computational effort, model complexity and prediction horizon. Therefore, in addition to deviations in terms of model parameter accuracy, prediction errors may also arise from (knowingly or unknowingly) omitted system dynamics.

Instead, data-driven methods such as reinforcement learning (RL) utilize interaction data to learn the optimal control behavior. This results in several advantages over conventional, solver-based optimal control schemes such as MPC, specifically in terms of computational efficiency and independence from model availability [5, 6]. Thereby, RL learns an explicit optimal control policy, whereby only the learned policy, but not the learning process, must be capable of real-time execution. Accordingly, MPC and RL follow different solution concepts, but both strive for optimal control performance [7]. As a result, RL enables a model-free, self-adaptive control approach that paves the way for non-linear control and enables a fast and largely automatic controller development process with minimum human effort [8, 9]. These attributes render RL promising for evolving plants such as the electrical power grid [10]. The advantages allow resolving some of the issues listed above when transferring to a hardware application, e.g., non-linearities, model inaccuracy or sensor calibration mismatch. However, the advantages are offset by the challenge of safety while training control applications on technical systems such as in a power-electronic grid applications [11].

1.2 State of the Art

The utilization of RL in the concept of MGs is not completely new. However, publications mostly document applications at a higher control level with a significantly larger step size in simulative investigations. These focus mainly on economic optimization, planning or power flow control, rather than control on component level, such as current or voltage control, where fast reactions on transients are required [12]. E.g., in [13], RL is used for energy management in an MG to maximize economic benefits depending on a local storage system, while considering operational constraints. In [14], RL is applied for scheduling flexible loads in an MG, while balancing the interests of multiple stakeholders on a monetary basis. Both areas are combined in [15], whereby RL is utilized to adjust the control parameters at the power flow level in order to optimize the MG's operating costs. Furthermore, there exist several investigations on applications of RL at the power-control level to adjust the grid frequency and voltage amplitude. E.g., [16] applies RL to optimize the voltage magnitude in consideration of the reactive power in a simulative power flow scenario.

In [17], RL is used for adapting the coefficients of a classic controller on power flow level, while the subordinate controllers on component level are realized via classic PI controllers. Accordingly, RL already shows advantages over conventional methods at the higher control levels, whereas the lower-level controls are still mapped by standard feedback control methods, such as PI controllers. For optimal operation of an MG, however, fast-responding controls are required at component level, as can be achieved by optimal controllers such as RL. Most investigations of applying RL-based control in power electronic and grid applications are based on simulation, while the transfer to a real-time capable hardware application is focus of current research [18]. Learning directly on the real-world system is one of the current challenges of RL-based control for power electronic systems and is not covered in literature so far. Merely, the completely simulative pre-trained controller is applied to actual hardware [19]. This is shown in [20], where an RL-based approach is initially trained in offline simulation for a voltage control application for power electronic converters. Later on, this pre-trained agent is presented to be applicable on the corresponding hardware. However, the advantage of self-adaptation is lost when not training directly on the hardware, since the simulation model used for training does not normally reflect the challenging effects (e.g., calibration mismatch or non-linearities). Only interacting with the hardware to be controlled enables RL to overcome such challenges.

Safety is becoming increasingly important, especially when transferring to real-world applications and training in a hardware experiment [5, 21]. Here, the concept of safety is defined as compliance with constraints of the plant system. Depending on the application, these can be physically determined constraints, e.g., current and voltage limits of electrical components in an inverter control task, or power limits in a grid control task, or financially determined constraints in an economic optimization. The former represent safety-relevant constraints, the violation of which can result in damage to the system,

in contrast to financial constraints overruns. For the sake of completeness, all categories are covered in the following literature review. In order to examine compliance with these safety-relevant constraints of RL in electric grid applications in the existing literature, safety is categorized as follows:

1. *No* safety concept is considered in the investigation.
2. Safety is *promoted* by structural changes to the learning algorithm, whereby safety cannot be guaranteed. This can be achieved through several approaches:
 - Incorporating safety factors or constraint penalty terms within the feedback during training, e.g., [Own5, 22–24].
 - Embedding safety considerations within the policy via extending the cost function of the learning algorithm, e.g., [25–28].
3. Structural changes are made to the RL-framework by extending the control structure, e.g., [24, 29–34] where safety and compliance with physical constraints are *enforced*.

Tab. 1.1 summarizes the state-of-the-art literature of RL in electric grid applications, classified into three categories. They cover *economic optimization*, *grid control* at power level, such as droop control under power flow considerations and, lastly, the *inverter control* at component level, the current and voltage control regarding the filter components (inductors and capacitors). Economic considerations are usually examined on the basis of simulation calculations, since there are no issues such as non-linearities, modeling inaccuracies, or sensor calibration deviations. Therefore, the transfer to hardware is not considered in the table.

Tab. 1.1: Reinforcement learning-based control in electric grid applications

Training:		Simulation		Simulation		Hardware	
Application:		Simulation		Hardware		Hardware	
Safety:	No	Promoted	Enforced	No	Promoted	Enforced	Enforced
Economic operation	[13–15] [35]	[27, 28]	[29, 30] [36]	-	-	-	-
Grid control	[16–18] [37–43]	[44–49] [23–26]	[31, 32] [50–53]	[54, 55]			
Inverter control	[56–59] [60–65]	[Own5]	[Own6]	[66, 67]	[20]		[Own1] [Own3] [Own2] [Own4]

As can be derived from Tab. 1.1, a lot of applications focus on the economic and power control level. In order to enable optimal use of the IBR, safe and fast control is required at the component level, enabling optimal operation up to the limits of the system, as made possible by RL. In order to learn how to deal with, e.g., non-linearities, modeling inaccuracies, or sensor calibration deviations, training must be carried out directly on the hardware. As can be seen in Tab. 1.1, no literature (apart from the author's own publications forming this work) covers training on the hardware plant, which highlights the research gap that will be addressed by this work.

1.3 Contribution

The goal of this work is the investigation of a toolchain to train an RL-based controller for a control task at component level. The considered application is a grid-forming voltage control task in an MG hardware application without prior simulative training. The constraints have to be taken into account to adhere to physical system limits of the plant inverter. For this purpose, a standard RL algorithm is extended by a real-time capable safety layer that allows the application in a hardware experiment in an MG and, hence, enables a fast and largely automatic controller development process with minimum human effort. The resulting RL-based control framework does not require any prior system parameter knowledge regarding the specific inverter. To determine the best configuration and architecture for the RL algorithm, a hyperparameter optimization (HPO) framework is proposed that exploits the real-time simulation capabilities of rapid control prototyping (RCP) hardware. The steady-state and transient behavior of the trained RL-based controller is experimentally validated on a 250 kVA industrial-grade three-level four-wire inverter and compared with classic PI controllers, predictive dead-beat controllers and MPC. In a follow-up experiment, a model-free droop control layer for active and reactive power sharing is added to the RL-based voltage controller trained directly on the hardware without simulative prior training. By doing so, a fully automated control design for MGs is achieved, which does not require any prior parametrization knowledge of the plant system, enabling an automatic and model-free commissioning of inverter-based MGs.

The main contributions of this work are:

- Real-time capable implementation of an RL-based voltage controller at component level within a system in the three-digit kVA range, rendering this investigation highly relevant for industrial applications.
- Introduction of an edge learning toolchain to outsource the training of the embedded RL-based controller from real-time hardware to high-performance workstations.
- Empirical proof for the safe application of the RL-based controller with special respect to its data-driven artificial neural network (ANN)-based premise. Hence, corresponding measurements are retrieved without prior assumptions about the plant

dynamics, but by initializing the required safeguarding technique on the basis of identified plant behavior [68].

- Setup of an HPO framework, executable with hardware-in-the-loop (HIL) experiments and RCP units, which allow to conduct real-time simulation experiments while searching for the corresponding hyperparameters (HPs) instead of using sequential simulations on computer clusters to reduce the simulation-reality gap.
- Direct comparison of steady-state and transient behavior of the developed RL-based controller to another optimal control method – MPC – and to classic PI controllers and predictive dead-beat controllers in real-world experiments.
- Extension beyond a proof-of-concept application with constant load to an application with arbitrary loads, ultimately validating the feasibility of the developed RL-based approach as a subordinate voltage controller in an MG application.

2 Fundamentals

As motivated in Sec. 1, this thesis targets to investigate whether RL can be used for voltage control of an inverter at component level in order to establish a self-adaptive control approach with minimum human effort. For this purpose, the necessary basics are introduced in the following along with the consideration of superimposed control levels regarding inverter-based MGs. Thereby, the latter are required to show the applicability of the RL-based voltage controller in a decentralized MG.

2.1 Optimal Control for Voltage-Forming Inverters

Given a dynamical system with the system state $\mathbf{x}$, system input $\mathbf{u}$, disturbance $\mathbf{d}$ and system output $\mathbf{y}$, the goal of optimal control is to find a control input for the system which is optimal with regard to a certain cost function [69]. In the following, the general control task is defined as a reference tracking problem with disturbance rejection and the system input is to be calculated in such a way that the output $\mathbf{y}$ follows a certain reference $\mathbf{y}^*$ without violating the system boundaries. These boundaries define the state, action and output space $\mathbb{X}, \mathbb{U}, \mathbb{Y}$, respectively, while $\mathbb{D}$ defines the disturbance space. The optimal control problem for a dynamic, discrete-time system can be defined as follows:

$$\begin{aligned} \min_{\mathbf{u}_k} \quad & \sum_{i=0}^{\infty} \gamma^i \mathcal{V}_{k+i+1}(\mathbf{y}_k^*, \mathbf{y}_k, \mathbf{u}_k), \\ \mathbf{x}_0 = & \mathbf{x}_{k=0}, \\ \mathbf{x}_{k+1} = & \mathbf{f}_k(\mathbf{x}_k, \mathbf{u}_k, \mathbf{d}_k), \\ \mathbf{y}_k = & \mathbf{w}_k(\mathbf{x}_k, \mathbf{u}_k, \mathbf{d}_k), \\ \mathbf{x}_k \in & \mathbb{X} \subseteq \mathbb{R}^n, \mathbf{u}_k \in \mathbb{U} \subseteq \mathbb{R}^m, \mathbf{y}_k, \mathbf{y}_k^* \in \mathbb{Y} \subseteq \mathbb{R}^l, \mathbf{d}_k \in \mathbb{D} \subseteq \mathbb{R}^o. \end{aligned} \tag{2.1}$$

Here, the function $\mathbf{f} : \mathbb{X} \times \mathbb{U} \times \mathbb{D} \rightarrow \mathbb{X}$ defines the state transition over time and $\mathbf{w} : \mathbb{X} \times \mathbb{U} \times \mathbb{D} \rightarrow \mathbb{Y}$ the output function, composing the available measurement signals. $\mathcal{V} : \mathbb{Y} \times \mathbb{Y} \times \mathbb{U} \rightarrow \mathbb{R}$ defines the stage costs, the sum of which must be minimized. Thereby, the discount factor $\gamma \in [0, 1[$ can be used to set whether future costs should be weighted more heavily or less heavily. Moreover, k denotes the discrete-time index and $\mathbf{x}_0$ is the initial state of the system. To formulate an optimal tracking problem for

a voltage-forming inverter, the stage costs $\mathcal{V}(\|\mathbf{v}_k^* - \mathbf{v}_k\|_p)$ are defined depending on the deviation between the measured voltage $\mathbf{v}$ and the targeted reference voltage $\mathbf{v}^*$, while the corresponding dynamic system is introduced in the next section.

2.2 Voltage Source Inverter

The three-phase voltage source inverter (VSI) with LC -filter, presented in Fig. 2.1, is a common topology as an interface between a DC-link and an electrical power grid, especially in the case of decentralized MGs. The depicted three-level topology, which is used in the subsequent application, offers advantages compared to a standard two-level inverter, such as lower component voltage stress, decreased total harmonic distortion (THD) and reduced losses at industry-grade switching frequency, which leads to lower operational temperature and failure rates. Thus, it provides higher reliability and a possible reduction in filter size, weight and costs [70–72]. Thereby, in both three-level and two-level architectures, the phase voltage v_i at the filter input can be modulated via a pulse width modulation (PWM) to enable a continuous control set (CCS). It is assumed that the pulse frequency components in the signals resulting from the switching behavior of the power electronic device are smoothed by a suitable filter design with a cutoff frequency significantly below the switching frequency. The controller should not compensate for this switching behavior, but rather for the error in the dynamic averaging model [73]. Therefore, regular sampling is applied (cf. Sec. 3.1). Thus, a continuously selectable voltage for the following control task can be assumed via PWM. More details can be found in, e.g., [74].

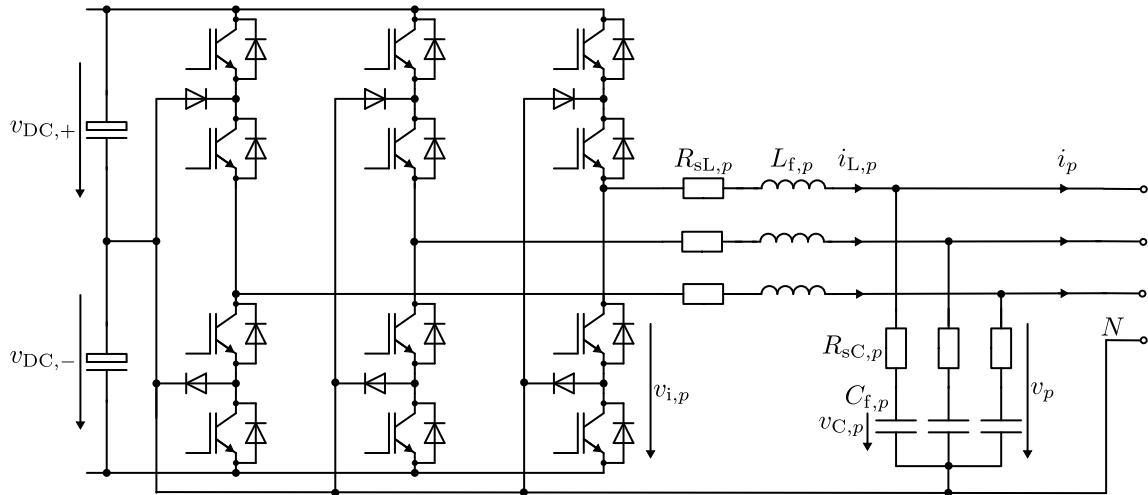


Fig. 2.1: Three-phase VSI with LC -filter, while one phase is labeled for the plant components with the index $p \in \{a, b, c\}$.

As depicted in Fig. 2.1, the considered topology is a three-level neutral-point-clamped (NPC) inverter, which consists of three equivalent phases with a common neutral point

N that is connected to the DC center potential. Therefore, the plant for the investigated voltage control task can be inferred from a single-phase representation. The current i_L through the filter inductor L_f and the voltage v_C across the filter capacitor C_f change depending on the filter input voltage v_i and the load current i , which acts as disturbance. One phase of the VSI in Fig. 2.1 can be modeled by a set of continuous-time ordinary differential equations (ODEs):

$$\begin{aligned}\frac{d}{dt}\mathbf{x}_s(t) &= \mathbf{A}_s\mathbf{x}_s(t) + \mathbf{b}_s u_s(t) + \mathbf{h}_s d_s(t), \\ \mathbf{y}_s(t) &= \mathbf{C}_s\mathbf{x}_s(t) + \mathbf{g}_s d_s(t),\end{aligned}\tag{2.2}$$

where

$$\begin{aligned}\mathbf{x}_s(t) &= \begin{bmatrix} i_L(t) \\ v_C(t) \end{bmatrix}, & \mathbf{y}_s(t) &= \begin{bmatrix} i_L(t) \\ v(t) \end{bmatrix}, & u_s(t) &= v_i(t), \\ d_s(t) &= i(t), & \mathbf{A}_s &= \begin{bmatrix} -\frac{R_{sL}+R_{sC}}{L_f} & -\frac{1}{L_f} \\ \frac{1}{C_f} & 0 \end{bmatrix}, & \mathbf{b}_s &= \begin{bmatrix} \frac{1}{L_f} \\ 0 \end{bmatrix}, \\ \mathbf{h}_s &= \begin{bmatrix} \frac{R_{sC}}{L_f} \\ -\frac{1}{C_f} \end{bmatrix}, & \mathbf{C}_s &= \begin{bmatrix} 1 & 0 \\ R_{sC} & 1 \end{bmatrix}, & \mathbf{g}_s &= \begin{bmatrix} 0 \\ -R_{sC} \end{bmatrix}.\end{aligned}$$

With the continuous-time state-space model (2.2) being linear, this ODE is approachable by means of exact discretization, which allows discrete-time analysis in the following without consideration of discretization errors (cf., e.g., [75]):

$$\mathbf{A} = e^{\mathbf{A}_s T_s}, \quad \mathbf{b} = (\mathbf{A} - \mathbf{I})\mathbf{A}_s^{-1}\mathbf{b}_s, \quad \mathbf{h} = (\mathbf{A} - \mathbf{I})\mathbf{A}_s^{-1}\mathbf{h}_s, \quad \mathbf{C} = \mathbf{C}_s, \quad \mathbf{g} = \mathbf{g}_s, \tag{2.3}$$

where T_s is the sampling time of the discrete-time system. The resulting discrete linear time-invariant (LTI) state-space model has the form:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{b}u_k + \mathbf{h}d_k \tag{2.4}$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{g}d_k. \tag{2.5}$$

However, the discrete-time system state $\mathbf{x}_k = [i_{L,k}, v_{C,k}]^T$ at time step k cannot be directly measured because of the internal series resistance R_{sC} of the filter capacitor (cf. Fig. 2.1), i.e., the internal capacitor voltage $v_{C,k}$ is not directly accessible. To depict the dynamics of the measured output $\mathbf{y}_k$, (2.5) is shifted by one time step $k + 1$ into the future and $\mathbf{x}_{k+1}$ from (2.4) is inserted:

$$\mathbf{y}_{k+1} = \mathbf{C}\mathbf{A}\mathbf{x}_k + \mathbf{C}\mathbf{b}u_k + \mathbf{C}\mathbf{h}d_k + \mathbf{g}d_{k+1}. \tag{2.6}$$

However, the state $\mathbf{x}_k$ is still included. Therefore, (2.5) is rearranged as

$$\mathbf{x}_k = \mathbf{C}^{-1}\mathbf{y}_k - \mathbf{C}^{-1}\mathbf{g}d_k, \tag{2.7}$$

and inserted into (2.6):

$$\mathbf{y}_{k+1} = \mathbf{C}\mathbf{A}\mathbf{C}^{-1}\mathbf{y}_k + \mathbf{C}\mathbf{b}u_k + (\mathbf{C}\mathbf{h} - \mathbf{C}\mathbf{A}\mathbf{C}^{-1}\mathbf{g})d_k + \mathbf{g}d_{k+1}. \quad (2.8)$$

This reformulation is possible for the considered system from Fig. 2.1, since the output matrix $\mathbf{C}$ is structurally invertible, i.e., the output and state variables can be transformed into each other without a loss of information. As a generalization of (2.8),

$$\mathbf{y}_{k+1} = \mathbf{f}'(\mathbf{y}_k, u_k, d_k, d_{k+1}) \quad (2.9)$$

directly describes the output dynamics with the transformed model $\mathbf{f}' : \mathbb{Y} \times \mathbb{U} \times \mathbb{D} \rightarrow \mathbb{Y}$. Assuming predictability of the disturbance variable, (2.8) can be condensed as follows:

$$\mathbf{y}_{k+1} = \underbrace{\mathbf{C}\mathbf{A}\mathbf{C}^{-1}}_{\mathbf{A}_y} \mathbf{y}_k + \underbrace{\mathbf{C}\mathbf{b}}_{\mathbf{b}_y} u_k + \underbrace{(\mathbf{C}\mathbf{h} - \mathbf{C}\mathbf{A}\mathbf{C}^{-1}\mathbf{g} + \mathbf{g}l_k)}_{\mathbf{g}_y} d_k. \quad (2.10)$$

Here, $l_k : \mathbb{D} \rightarrow \mathbb{D}$ defines the assumed disturbance dynamic, i.e., $\hat{d}_{k+1} = l_k d_k$. The necessity, validity and implementation of this assumption depend on the application and will be discussed accordingly at a later stage.

2.3 Reinforcement Learning Fundamentals

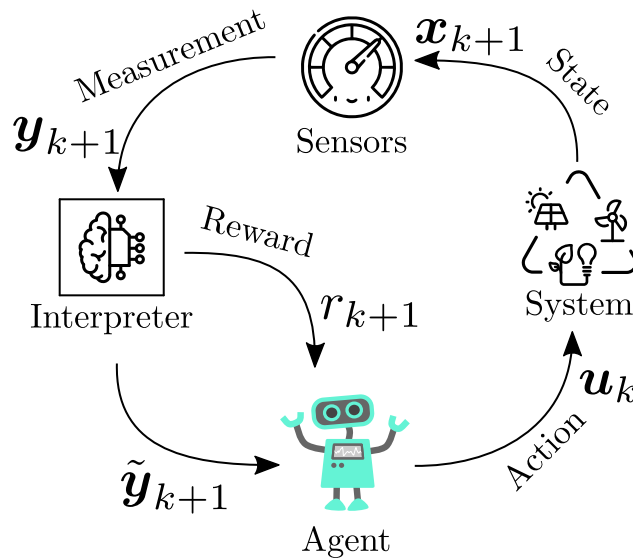


Fig. 2.2: Reinforcement learning agent in the interaction with the environment (derivative of [76]).

RL is a subdomain of machine learning where an artificial intelligence (AI) – the agent – learns a task by interacting with its surroundings – the environment. For this purpose, the RL agent applies an action $\mathbf{u}_k$ at time step k to the environment, which is determined on the basis of the momentary measurement $\mathbf{y}_k$. In the following, the environment to interact with is the VSI introduced in Sec. 2.2. As depicted in Fig. 2.2, the action is evolved within the environment, leading to the next system state $\mathbf{x}_{k+1}$, which can be measured by sensors delivering the measurement $\mathbf{y}_{k+1}$. To learn from this interaction, the action is rated by the reward r_{k+1} , which defines the extent to which the control goal has been reached.

In the following derivation of the theoretical principles, it is assumed that the system behaves deterministically. Consequently, the aim is to find a deterministic control policy for the technical system, since stochastic behavior is undesirable in this context. Nevertheless, the complete theory part is also valid for the stochastic case considering expected values, as it would be useful, for example, when learning a policy in the context of games such as poker, where predictable behavior – unlike in the control of technical systems – can be a disadvantage [77].

In the RL framework, the environment is abstracted by a Markov decision process (MDP), where the history of a state is fully condensed in the momentary sample. This is expressed in the Markov property:

$$\begin{aligned}\mathbf{x}_{k+1} &= \hat{\mathbf{f}}(\mathbf{x}_k, \mathbf{x}_{k-1}, \dots, \mathbf{u}_k, \mathbf{u}_{k-1}, \dots, \mathbf{d}_k, \mathbf{d}_{k-1}, \dots) \\ &= \hat{\mathbf{f}}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{d}_k),\end{aligned}\tag{2.11}$$

which clarifies that the next state is only dependent on the information from the previous time step [78]. As can be seen in (2.9), this does not apply to the system under consideration here, since the next state also depends on future values. Additionally, $\mathbf{x}_k$ cannot be measured directly, but as described in Sec. 2.2, the dynamics of the system can be reformulated to represent the dynamic behavior regarding the system output $\mathbf{y}_k$ without loss of information. Therefore, the Markov property can be approximated by corresponding feature engineering:

$$\tilde{\mathbf{y}}_k = \phi(\mathbf{y}_k, \mathbf{y}_{k-1}, \mathbf{y}_{k-2}, \dots, \mathbf{u}_{k-1}, \dots, \mathbf{d}_k, \mathbf{d}_{k-1}),\tag{2.12}$$

with $\phi : \mathbb{Y} \times \mathbb{U} \times \mathbb{D} \rightarrow \tilde{\mathbb{Y}} \subseteq \mathbb{R}^{\tilde{l}}$. Extending the measurement vector $\mathbf{y}_k$ by making past measurements available to the agent, an attempt is made to reproduce the Markov property by introducing time lag features for some observables, cf. [79]. This is not exclusive to measurements, but may also be applied to, e.g., past actions. Additionally, expert knowledge can be used here to add or extract additional information in order to improve the learning process. According to (2.12), a prediction of the disturbance variable is required for the system described in (2.9). This can be understood from the perspective of expert knowledge and will be discussed in detail at the relevant points of the specific applications. As a control task is to be learned in the following, the reference values, e.g., $\mathbf{v}^*$ for the voltage control task introduced in Sec. 2.1, can be made available to the agent via feature engineering, too.

In the context of closed-loop control, the goal of RL is to find an explicit policy $\mathbf{u}_k = \boldsymbol{\mu}(\tilde{\mathbf{y}}_k)$ (with $\boldsymbol{\mu} : \tilde{\mathbb{Y}} \rightarrow \mathbb{U}$) that maps each measurement to an action which is supposed to maximize the return g_k , which is defined by the cumulative, discounted reward:

$$g_k = r_{k+1} + \gamma r_{k+2} + \gamma^2 r_{k+3} + \dots = \sum_{i=0}^{\infty} \gamma^i r_{k+i+1} = \sum_{i=0}^{\infty} \gamma^i (1 - \mathcal{V}_{k+i+1}). \quad (2.13)$$

Based on the definition of optimal control in (2.1), the return to be maximized can be linked to the cost function as defined above. Herein, it can also be seen that the return maximization problem can be equivalently translated to a cost minimization problem as it is introduced in Sec. 2.1 and typically used within MPC [4, 80]. This highlights the close relation between RL and MPC, which both are disciplines of optimal control [7]. Consequently, the reward design defines the task to be learned and thus represents one of the most important degrees of freedom for RL-based controller design.

In an approximate MDP, the value of an action with regard to the expected return $q : \tilde{\mathbb{Y}} \times \mathbb{U} \rightarrow \mathbb{R}$ is determined by the Bellman equation:

$$q(\tilde{\mathbf{y}}_k, \mathbf{u}_k) = g_k = r_{k+1} + \gamma q(\tilde{\mathbf{y}}_{k+1}, \mathbf{u}_{k+1}). \quad (2.14)$$

The learning goal in RL-based control is to find an optimal policy $\boldsymbol{\mu}^*$ which maps the momentary measurement to the action that maximizes the return:

$$\mathbf{u}_k^* = \boldsymbol{\mu}^*(\tilde{\mathbf{y}}_k) = \arg \max_{\mathbf{u}_k} q(\tilde{\mathbf{y}}_k, \mathbf{u}_k). \quad (2.15)$$

2.3.1 Function Approximation in Reinforcement Learning

As introduced in Sec. 2.2, the state space according to (2.12) and the action chosen $\mathbf{u}_{\text{RL}}$ by the agent are continuous, so that the control task under investigation takes place in a continuous state-action space. In contrast to a finite state-action space, where tabular methods (cf. [78]) can be used to determine the value of an action in a state, the first challenge in the continuous state-action space application is to determine the q -value defined in (2.14). Covering that by a tabular-based approach would require infeasible to infinite memory, time and data demand [78, 81]. Therefore, the q -function is approximated by a parametrizable function approximator $\hat{q}_{\boldsymbol{\rho}}$ with parameters $\boldsymbol{\rho}$. The second challenge is to solve the maximization problem in (2.15). While it is possible in finite-control-set (FCS) applications to evaluate the limited number of possibilities according to (2.15) and select the best one, this becomes infeasible in CCS applications. For this purpose, a second function approximator $\hat{\boldsymbol{\mu}}_{\boldsymbol{\delta}}$, with parameters $\boldsymbol{\delta}$, defines the approximate optimal policy. Thereby, the number of parameters of the function approximators is much smaller than the number of possible states and actions. Consequently, changing a parameter affects not only one state, but many, which is called generalization. Suitable function approximators are required, which must be selected depending on the task to be solved. In the following applications, deep, state-less feedforward ANNs are utilized as function approximators. For their training, the deep deterministic policy gradient (DDPG) algorithm is utilized,

which yields a deterministic policy for the control problem defined in Sec. 2.1 [82]. The training of the parameters is based on the state transition experience:

$$\xi_k = \langle \tilde{\mathbf{y}}_k, \mathbf{u}_{\text{RL},k}, r_{k+1}, \tilde{\mathbf{y}}_{k+1}, t_{k+1} \rangle, \quad (2.16)$$

that is gathered from interaction with the environment. Here, t_{k+1} is set to 1 if $\mathbf{y}_{k+1}$ is terminal (e.g., in case of an emergency shutdown triggered by a limit violation), otherwise $t_{k+1} = 0$. While general information about the structure and training of ANNs can be found, e.g., in [78, 83, 84] the most important details about DDPG are described in Sec. A.1 for the sake of completeness and for the convenience of the reader.

2.3.2 Exploration vs. Exploitation

As introduced in (2.15), during training, the optimal action is chosen based on the momentary policy and the momentary policy is trained based on the experience gained in historic interactions. To achieve optimality of the policy, a complete coverage of the entire state-action space within that data is required. For this purpose, unknown state and action pairs have to be explored, which contradicts the control task and, hence, would not maximize the return (cf. (2.13) and (2.15)). Adequate exploration must be guaranteed during training. Since the trained policy is deterministic, exploration during the training is realized by an additional random signal $\boldsymbol{\nu}_k$. In the following a discrete-time Ornstein-Uhlenbeck (OU) noise with zero mean (cf. Sec. A.1 and [82]) is used, which is superimposed on the action determined by the policy and is therefore called action noise:

$$\mathbf{u}_k = \hat{\boldsymbol{\mu}}_\delta(\tilde{\mathbf{y}}_k) + \boldsymbol{\nu}_k. \quad (2.17)$$

This results in random voltages applied to the filter in order to yield previously unseen state transitions and state-action pairs. Over the course of the training, the action noise is linearly scaled down. Full action noise in the early learning phase allows a proper exploration, while to the end of the training the scaling puts the focuses more onto exploitation. After the training phase, the noise is completely turned off to enable the best possible control performance during the specific application, learned based on appropriate exploration during training. A complete exploration of the state-action space of a technical system may result in safety-critical situations if limit violations are provoked. This is critical when training directly on the real-world system, as will be discussed in this thesis. In order to ensure compliance with the system limits while simultaneously allowing sufficient exploration during training a safeguard algorithm will be introduced. It is set up to cancel and overwrite control actions from the RL agent if they are expected to violate systemic constraints.

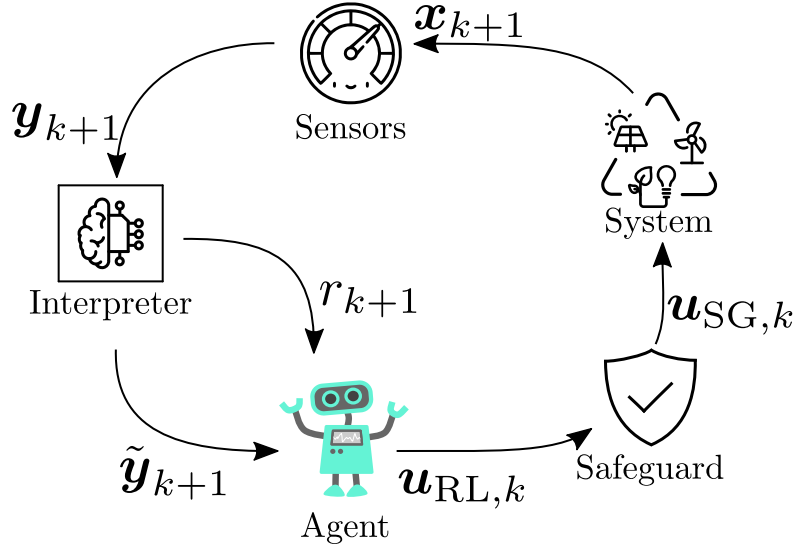


Fig. 2.3: Reinforcement learning framework extended by a safeguard component to ensure safe interaction with the environment (derivative of [76]).

2.4 Safeguarding in Continuous State and Action Space

In this chapter, the introduced RL framework is extended by a safety component – the safeguard (cf. Fig. 2.3) – to enable training in direct interaction with the plant hardware. The basics are derived from [85] and have been validated using a simple standard example in [86].

2.4.1 Derivation of the Relevant Constraints

During interaction, the RL controller learns the physical limitations of the environment, which must not be violated, via the reward feedback. These limitations are defined according to (2.1) by the state-action space $\mathbb{F}_0 = \mathbb{Y} \times \mathbb{U}$ of the continuous, physical quantities.¹ Therefore, especially in the early learning phase, the explorative behavior required for the training (cf. Sec. 2.3.2) can cause hardware damage or may provoke emergency shutdowns, if $\mathbf{y}_k$ exceeds the permissible system limits. To avoid this, the RL framework is extended by a safeguard, as depicted in Fig. 2.3. The safeguard’s task is to change the action $\mathbf{u}_{RL,k}$ selected by the agent if necessary, such that no constraints are violated in future steps. According to [85], this is the case if the action is chosen so that the point $[\mathbf{y}_k^T, \mathbf{u}_k^T]^T$ lies within the feasible set $\mathbb{F}_{\infty \rightarrow 0}$, whose computation is explained in

¹For better readability and to align with the term commonly used in the literature, the term state space is used, even though the calculations refer to the measured system output $\mathbf{y}$, which, apart from parasitic effects, corresponds to the states as already described in Sec. 2.2 (cf. (2.6)).

the following. This feasible set can be calculated starting from (2.10) and streamlining the one-step notation prediction and computation into a condensed form considering k_f prediction steps into the future from time step k . This yields:

$$\mathbf{y}_k = \underbrace{\begin{bmatrix} \mathbf{A}_y \\ \mathbf{A}_y^2 \\ \vdots \\ \mathbf{A}_y^{k_f} \end{bmatrix}}_{\mathcal{A}} \mathbf{y}_k + \underbrace{\begin{bmatrix} \mathbf{b}_y & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{A}_y \mathbf{b}_y & \mathbf{b}_y & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_y^{k_f-1} \mathbf{b}_y & \mathbf{A}_y^{k_f-2} \mathbf{b}_y & \dots & \mathbf{b}_y \end{bmatrix}}_{\mathcal{B}} \mathbf{u}_k + \underbrace{\begin{bmatrix} \mathbf{g}_y & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{A}_y \mathbf{g}_y & \mathbf{g}_y & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_y^{k_f-1} \mathbf{g}_y & \mathbf{A}_y^{k_f-2} \mathbf{g}_y & \dots & \mathbf{g}_y \end{bmatrix}}_{\mathcal{G}} \mathbf{D}_k, \quad (2.18)$$

with

$$\mathbf{y}_k = [\mathbf{y}_{k+1}^\top \mathbf{y}_{k+2}^\top \dots \mathbf{y}_{k_f}^\top]^\top, \quad \mathbf{u}_k = [\mathbf{u}_k^\top \mathbf{u}_{k+1}^\top \dots \mathbf{u}_{k_f-1}^\top]^\top, \quad \mathbf{D}_k = [\mathbf{d}_k^\top \mathbf{d}_{k+1}^\top \dots \mathbf{d}_{k_f-1}^\top]^\top.$$

For a single time step, the linear input and output constraints are defined as

$$\mathbf{W}_u \mathbf{u}_k \leq \boldsymbol{\omega}_u, \quad \mathbf{W}_y \mathbf{y}_k \leq \boldsymbol{\omega}_y, \quad (2.19)$$

which form convex polytopes. A polytope $\mathbb{F}_0$ is defined as a finite set of intersections of closed half-spaces, while $\mathbb{F}_0$ is convex if the connection line between any two points $\{\mathbf{a}, \mathbf{b}\} \in \mathbb{F}_0$ is contained in $\mathbb{F}_0$ [85]. An example of a three-dimensional, convex polytope is a cube. Another example for a three-dimensional polytope is depicted in green in Fig. 2.4, while a two-dimensional polytope is shown in yellow. Since the constraints defined by the physical limits in (2.19) must be adhered to not only in the momentary but also in future time steps, the condensed constraints can be defined as:

$$\mathcal{W}_u \mathbf{u}_k \leq \boldsymbol{\Omega}_u, \quad \mathcal{W}_y \mathbf{y}_k \leq \boldsymbol{\Omega}_y, \quad (2.20)$$

with

$$\begin{aligned} \mathcal{W}_u &= \text{diag} \underbrace{[\mathbf{W}_u \dots \mathbf{W}_u]}_{k_f \text{ times}}, & \boldsymbol{\Omega}_u &= \underbrace{[\boldsymbol{\omega}_u^\top \dots \boldsymbol{\omega}_u^\top]^\top}_{k_f \text{ times}}, \\ \mathcal{W}_y &= \text{diag} \underbrace{[\mathbf{W}_y \dots \mathbf{W}_y]}_{k_f \text{ times}}, & \boldsymbol{\Omega}_y &= \underbrace{[\boldsymbol{\omega}_y^\top \dots \boldsymbol{\omega}_y^\top]^\top}_{k_f \text{ times}}. \end{aligned}$$

Only the momentary measured value $\mathbf{y}_k$ is available. Therefore, (2.18) is inserted into (2.20), resulting in a formulation of the limitations of future measured values depending on the manipulated variables $\mathbf{u}_k$ to be selected:

$$\mathcal{W}_y \mathcal{B} \mathbf{u}_k \leq \boldsymbol{\Omega}_y - \mathcal{W}_y \mathcal{A} \mathbf{y}_k - \mathcal{W}_y \mathcal{G} \mathbf{D}_k. \quad (2.21)$$

From (2.21) and (2.20) the matrices

$$\mathbf{G} = \begin{bmatrix} \mathcal{W}_y \mathcal{B} \\ \mathcal{W}_u \end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix} \boldsymbol{\Omega}_y - \mathcal{W}_y \mathcal{G} \mathbf{D}_k \\ \boldsymbol{\Omega}_u \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} -\mathcal{W}_y \mathcal{A} \\ \mathbf{0} \end{bmatrix} \quad (2.22)$$

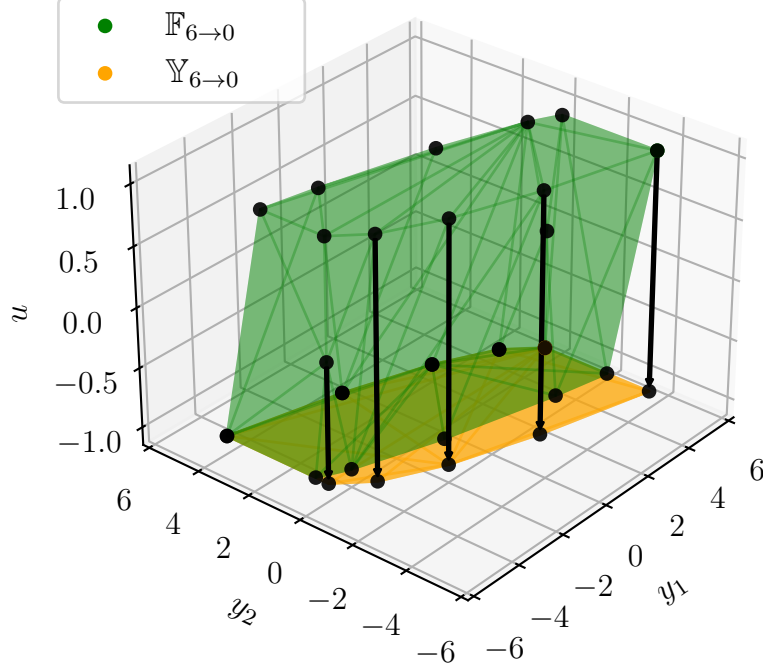


Fig. 2.4: Example of the projection of the three-dimensional feasible state-action space $\mathbb{F}_{6 \rightarrow 0}$, while considering $k_f = 6$ future time steps shown in green onto the corresponding two-dimensional feasible state space $\mathbb{Y}_{6 \rightarrow 0}$ shown in yellow. The corresponding application example is presented in Sec. 2.4.2.

define the inequality conditions which can be used to calculate the polytope $\mathbb{F}_{k_f}$, considering the next k_f actions [85]:

$$\mathbb{F}_{k_f} = \left\{ \begin{bmatrix} \mathbf{y}_k \\ \mathbf{u}_k \end{bmatrix} \in \mathbb{R}^{n+k_fm} \left| \begin{bmatrix} \mathbf{W}_y & \mathbf{0} \\ \mathbf{E} & \mathbf{G} \end{bmatrix} \begin{bmatrix} \mathbf{y}_k \\ \mathbf{u}_k \end{bmatrix} \leq \begin{bmatrix} \boldsymbol{\omega}_y \\ \mathbf{e} \end{bmatrix} \right. \right\}. \quad (2.23)$$

For the safeguard, only the immediately upcoming decision is to be affected by the momentary and future restrictions. According to [85], this is considered by projecting $\mathbb{F}_{k_f}$ onto the momentary time step:

$$\text{proj}_{\mathbb{F}_0}(\mathbb{F}_{k_f}) = \mathbb{F}_{k_f \rightarrow 0} = \left\{ \begin{bmatrix} \mathbf{y}_k \\ \mathbf{u}_k \end{bmatrix} \in \mathbb{R}^{n+m} \left| \mathbf{G}_{\text{feas}} \begin{bmatrix} \mathbf{y}_k \\ \mathbf{u} \end{bmatrix} \leq \mathbf{e}_{\text{feas}} \right. \right\}. \quad (2.24)$$

Here, the notation $\mathbb{F}_{k_f \rightarrow 0}$ highlights that the influence of k_f future state-action restrictions affect the momentary decision regarding the action $\mathbf{u}_k$ to be evaluated. An example of the

projection can be seen in Fig. 2.4. There, the three-dimensional feasible set shown in green is projected onto the two-dimensional set shown in orange. The corresponding application example is explained in detail in the next Sec. 2.4.2. As can be seen, this is a projection of the feasible state-action space considering six prediction horizon steps $\mathbb{F}_{6 \rightarrow 0}$ onto the corresponding feasible state space $\mathbb{Y}_{6 \rightarrow 0} = \{\mathbf{y}_k \in \mathbb{Y} \mid \exists \mathbf{u}_k \in \mathbb{R}^{6m} : \mathbf{G}\mathbf{u}_k \leq \mathbf{E}\mathbf{y}_k + \mathbf{e}\}$. The projection operation here is done utilizing computational geometry, cf. [85, 87], with toolboxes like *Multi-Parametric Toolbox 3.0* [88] or *Polytope* [89].

The calculation of the feasible set presented above is repeated with an increasing prediction horizon k_f until future restrictions no longer have an effect on the feasible state-action space of the momentary time step. This is indicated by no decreasing volume $V(\mathbb{F}_{k_f \rightarrow 0})$ of the projected feasible set for an increasing prediction horizon k_f . Then the projected feasible set no longer changes and thus represents the sought, final feasible set $\mathbb{F}_{\infty \rightarrow 0}$. The calculation of $\mathbb{F}_{\infty \rightarrow 0}$ is summarized in algorithm 1. The matrices $\mathbf{G}_{\text{feas}}$ and $\mathbf{e}_{\text{feas}}$ result

Algorithm 1 Calculation of the feasible set $\mathbb{F}_{\infty \rightarrow 0}$

Input: Input and output constraints $\mathbf{W}_u, \boldsymbol{\omega}_u, \mathbf{W}_y, \boldsymbol{\omega}_y$
Init: $V_{\text{old}} = 0$
for $k_f = 1, 2, \dots$ **do**
 Calculate $\mathcal{A}$ and $\mathcal{B}$ based on (2.18)
 $\mathbf{W}_u = \text{diag}[\mathbf{W}_u \dots \mathbf{W}_u]$
 $\mathbf{W}_y = \text{diag}[\mathbf{W}_y \dots \mathbf{W}_y]$
 $\boldsymbol{\Omega}_u = [\boldsymbol{\omega}_u \dots \boldsymbol{\omega}_u]^\top$
 $\boldsymbol{\Omega}_y = [\boldsymbol{\omega}_y^\top \dots \boldsymbol{\omega}_y^\top]^\top$
 Calculate $\mathbf{G}, \mathbf{e}, \mathbf{E}$ based on (2.22)
 Calculate $\mathbb{F}_{k_f}$ based on (2.23)
 Calculate $\mathbb{F}_{k_f \rightarrow 0} = \text{proj}_{\mathbb{F}_0}(\mathbb{F}_{k_f})$ according to [85]
 if $|V_{\text{old}} - V(\mathbb{F}_{k_f \rightarrow 0})| \leq \Delta V_{\text{min}}$ **then**
 $\mathbb{F}_{\infty \rightarrow 0} = \mathbb{F}_{k_f \rightarrow 0}$
 break
 end if
 $V_{\text{old}} = V(\mathbb{F}_{k_f \rightarrow 0})$
end for

from the projection and can then be used within the safeguard to change the action $\mathbf{u}_{\text{RL},k}$ – if necessary – as little as possible to $\mathbf{u}_{\text{SG},k}$ (cf. Fig. 2.3), such that it satisfies the system constraints in the momentary and future time steps by minimizing the convex cost function:

$$\begin{aligned} \mathbf{u}_{\text{SG},k} &= \arg \min_{\mathbf{u}_k} \|\mathbf{u}_k - \mathbf{u}_{\text{RL},k}\|^2, \\ \text{s.t.} \quad & \mathbf{G}_{\text{feas}} \begin{bmatrix} \mathbf{y}_k \\ \mathbf{u}_k \end{bmatrix} \leq \mathbf{e}_{\text{feas}}. \end{aligned} \tag{2.25}$$

2.4.2 Example: Double Integrator

The following example application illustrates the derivation of the relevant constraints described in Sec. 2.4.1 using the example of the fully measurable double integrator:

$$\mathbf{y}_{k+1} = \underbrace{\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}}_{\mathbf{A}_y} \mathbf{y}_k + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{\mathbf{b}_y} u_k, \quad (2.26)$$

subject to the input and output constraints

$$-1 \leq u_k \leq 1 \quad \text{and} \quad \begin{bmatrix} -5 \\ -5 \end{bmatrix} \leq \mathbf{y}_k \leq \begin{bmatrix} 5 \\ 5 \end{bmatrix} \quad \text{for all } k \geq 0. \quad (2.27)$$

The linear constraints can be rewritten according to (2.19) in polytope form (cf. [85]):

$$\underbrace{\begin{bmatrix} -1 \\ 1 \end{bmatrix}}_{\mathbf{w}_u} u_k \leq \underbrace{\begin{bmatrix} 1 \\ 1 \end{bmatrix}}_{\omega_u} \quad \text{and} \quad \underbrace{\begin{bmatrix} -1 & 0 \\ 1 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix}}_{\mathbf{w}_y} \mathbf{y}_k \leq \underbrace{\begin{bmatrix} 5 \\ 5 \\ 5 \\ 5 \end{bmatrix}}_{\omega_y}. \quad (2.28)$$

Representing the state-action space in a polytope according to (2.28) without considering the dynamics in (2.26) results in a cube as depicted in Fig. 2.5a). This cube represents all the state-action pairs which are technically possible regarding the defined system constraints. Considering the dynamics of the system may shrink the feasible set of state-action pairs. This can be seen by starting from (2.18) and calculating the feasible set (2.24) for the constraint double integrator described above for $k_f = 1$. The resulting feasible set $\mathbb{F}_{1 \rightarrow 0}$ is depicted in Fig. 2.5b) and Fig. 2.6. As can be seen, the cube is reduced and additional linear constraints cutting the edges of $\mathbb{F}_0$ (cf. Fig. 2.5a)) have been added to $\mathbf{G}_{1 \rightarrow 0}$ and $\mathbf{e}_{1 \rightarrow 0}$. For $k_f = 1$, (2.22) results in the matrices:

$$\mathbf{G} = \begin{bmatrix} \mathbf{w}_y \mathbf{B} \\ \mathbf{w}_u \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \\ -1 \\ 1 \end{bmatrix}, \quad \mathbf{e} = \begin{bmatrix} \Omega_y \\ \Omega_u \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \\ 5 \\ 5 \\ 1 \\ 1 \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} -\mathbf{w}_y \mathbf{A} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & -1 \\ 0 & 1 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \quad (2.29)$$

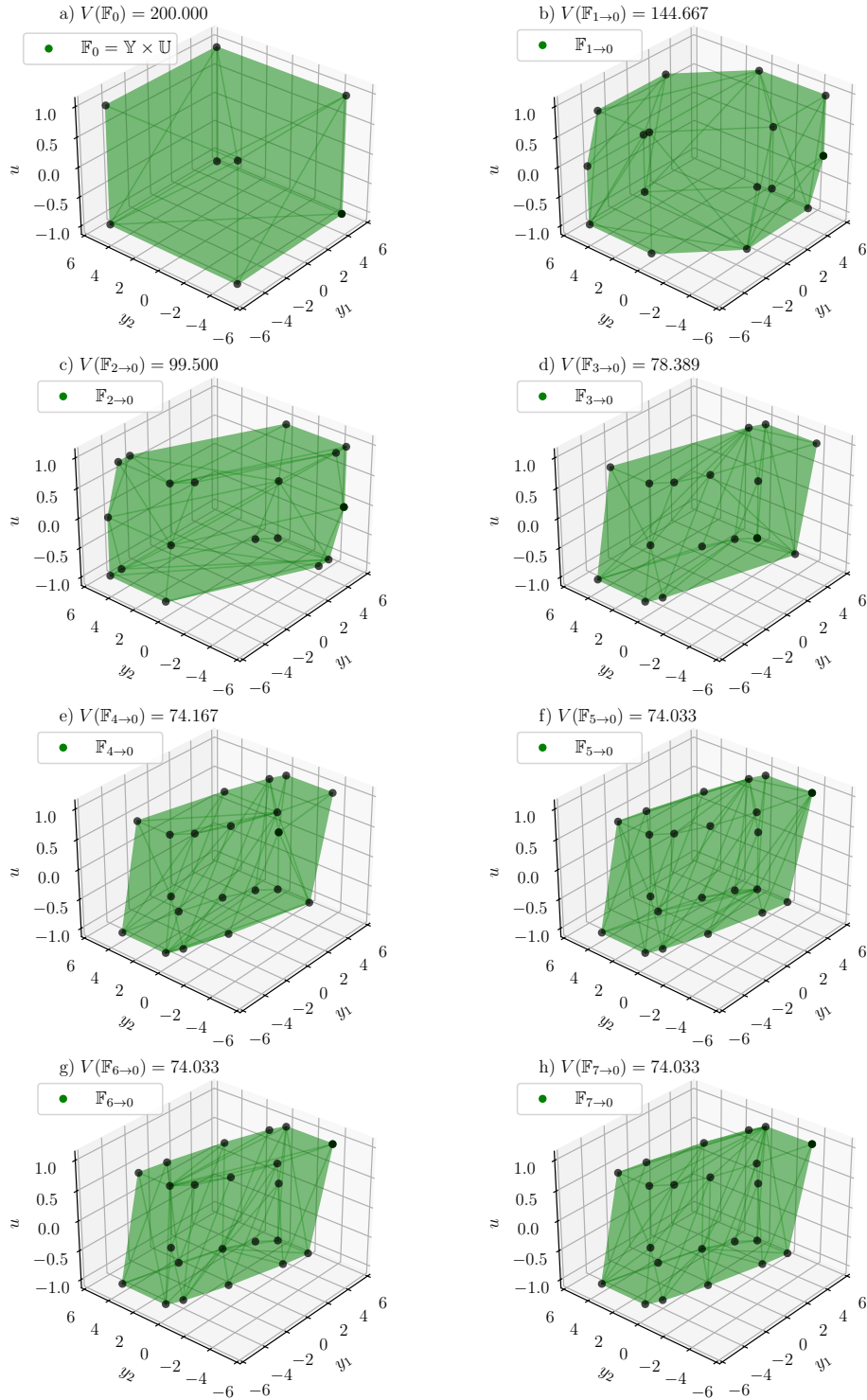


Fig. 2.5: State-action space $\mathbb{F}$ and feasible sets $\mathbb{F}_{k_f \rightarrow 0}$ for increasing k_f with corresponding volume for the double integrator described in (2.26) and (2.27).

Since for $k_f = 1$ the condensed matrices correspond to the non-condensed matrices, e.g.:

$$\mathcal{W}_y = \underbrace{[W_y \dots W_y]}_{k_f=1} = W_y,$$

the matrices $G_{1 \rightarrow 0}$ and $e_{1 \rightarrow 0}$ can be calculated without a projection operation, since no influence of restrictions of future time steps on the momentary one is taken into account. Accordingly, the feasible set is given by:

$$\mathbb{F}_{1 \rightarrow 0} = \left\{ \begin{array}{c} \left[\begin{array}{c} \mathbf{y}_k \\ u_k \end{array} \right] \in \mathbb{R}^{n+m=3} \mid \underbrace{\begin{bmatrix} -1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ -1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix}}_{G_{1 \rightarrow 0}} \left[\begin{array}{c} \mathbf{y}_k \\ u_k \end{array} \right] \leq \underbrace{\begin{bmatrix} 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 1 \\ 1 \end{bmatrix}}_{e_{1 \rightarrow 0}} \end{array} \right\}. \quad (2.30)$$

These additional restrictions can be explained with the dynamic equation (2.26) of the double integrator in combination with the constraints in (2.27). The dynamics (2.26) in outer areas of the cube can result in a movement of the system outputs ending outside of the defined system boundaries. This must be counteracted by the system input u_k . However, according to (2.27), the input variable is limited to ± 1 , which, e.g., at the point $\mathbf{y}_k^\top = [-5 \ -4]$, is not sufficient to compensate for a limit violation in the next time step. Hence, already the transition to $\mathbf{y}_k^\top = [-5 \ -4]$ must be prevented early on. An additional difficulty of the system under consideration is that the system input can only indirectly affect y_1 by first changing y_2 . Taking into account a single prediction horizon step is not sufficient to calculate the final feasible set, which comprises the state-action pairs that are feasible in order to not exceed the system boundaries on an infinite time horizon. This is demonstrated in Fig. 2.6, where the blue point $[\mathbf{y}_k^\top \ u_k]^\top = [0 \ -5 \ 1]^\top$ is located inside $\mathbb{F}_{1 \rightarrow 0}$. Substituting $[\mathbf{y}_k^\top \ u_k]^\top$ into (2.26) yields the red point $\mathbf{y}_{k+1}^\top = [-5 \ -4]$ ($u_{k+1} \in \mathbb{U}$ can be selected within the next time step and is chosen as -1 for graphical representation), which still lies within the defined system state boundaries. I.e., the feasible set $\mathbb{F}_{1 \rightarrow 0}$ marked in green defines the space in which state-action pairs can be selected so that the system limits are not exceeded in the next step $k_f = 1$. However, for this statement to hold true for all future time steps, every point in $\mathbb{F}_{1 \rightarrow 0}$ must map to the feasible state space $\mathbb{Y}_{1 \rightarrow 0}$ to represent the final feasible set $\mathbb{F}_{\infty \rightarrow 0}$:

$$\mathbf{A}_y \mathbf{y}_k + \mathbf{b}_y u_k \rightarrow \mathbb{Y}_{\infty \rightarrow 0} \quad \forall [\mathbf{y}_k^\top, u_k]^\top \in \mathbb{F}_{\infty \rightarrow 0}. \quad (2.31)$$

This does not hold for the $[\mathbf{y}_k^\top \ u_k]^\top = [0 \ -5 \ 1]^\top$ in Fig. 2.6. If the calculation described above is continued with $\mathbf{y}_{k+1}$, it is apparent that there is no $u_{k+1} \in \mathbb{U}$ that

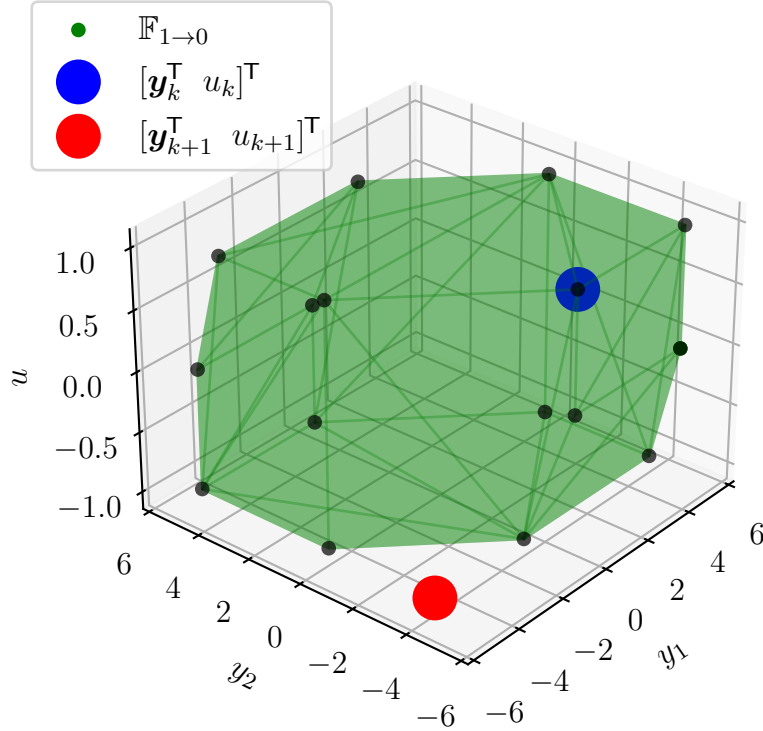


Fig. 2.6: Feasible set $\mathbb{F}_{1 \rightarrow 0}$ of the double integrator considering $k_f = 1$, with $[\mathbf{y}_k^T \ u_k]^T = [0 \ -5 \ 1]^T$ resulting in $[\mathbf{y}_{k+1}^T \ u_{k+1}]^T = [-5 \ -4 \ -1]^T$ ($u_{k+1} \in \mathbb{U}$ can be selected within the next time step and is chosen for graphical representation to be -1).

prevents $\mathbf{y}_{k+2}$ from exceeding the limits. Extending the calculations in (2.30) to take further prediction steps into account adds a row per prediction step to $\mathbf{G}$ (while (2.29) has to be extended accordingly). Since only the restrictions of the (future) limitations onto the momentary chosen action u_k to be applied are of interest, the projection operation described in Sec. 2.4.1 has to be applied. Following algorithm 1, (2.31) holds when the volume of the resulting feasible set $V(\mathbb{F}_{1 \rightarrow 0})$ does not decrease further for increasing k_f . However, as can be seen in Fig. 2.5, the volume changes until $k_f = 5$, at which point the algorithm converges. The sets depicted in Fig. 2.5 with $k_f = 5, 6, 7$ are identical feasible sets $\mathbb{F}_{\infty \rightarrow 0}$, which satisfy (2.31).

2.4.3 Differential Constraints

To limit the rate of change of a variable, differential constraints are introduced in the following. The matrices derived in Sec. 2.4.1 defining the constraints for the safeguard's optimizer have to be extended. This can be for example motivated if the capacitor current i_C of the system presented in Sec. 2.2 is to be limited to prevent thermal overloading of

the capacitor. Since i_C is not measured directly, this constraint can be indirectly realized by limiting the rate of change:

$$\Delta_k V_{\max} = \frac{I_{C,\max}}{C_f} \quad (2.32)$$

of the corresponding voltage v , resulting in the differential constraints:

$$\mathbf{W}_{\Delta_k \mathbf{y}} \Delta_k \mathbf{y}_k \leq \boldsymbol{\omega}_{\Delta_k \mathbf{y}}, \quad (2.33)$$

with

$$\Delta_k \mathbf{y}_k = \frac{\mathbf{y}_k - \mathbf{y}_{k-1}}{T_s}, \quad \mathbf{W}_{\Delta_k \mathbf{y}} = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix}, \quad \boldsymbol{\omega}_{\Delta_k \mathbf{y}} = \Delta_k V_{\max} \mathbf{1}_{2 \times 1}. \quad (2.34)$$

Here, $\mathbf{1}$ denotes a matrix of ones with the specified size. To consider the change-rate constraint for k_f prediction steps, according to [Own1] the condensed constraints can be defined as:

$$\mathcal{W}_{\Delta_k \mathbf{y}} \Delta_k \mathcal{Y}_k \leq \boldsymbol{\Omega}_{\Delta_k \mathbf{y}}, \quad (2.35)$$

with

$$\Delta_k \mathcal{Y}_k = \frac{1}{T_s} (\mathcal{Y}_k - \mathcal{K} \mathcal{Y}_k - \mathcal{L} \mathbf{y}_k), \quad (2.36)$$

and

$$\begin{aligned} \boldsymbol{\Omega}_{\Delta_k \mathbf{y}} &= [\boldsymbol{\omega}_{\Delta_k \mathbf{y}}^\top \ \dots \ \boldsymbol{\omega}_{\Delta_k \mathbf{y}}^\top]^\top, \quad \mathcal{W}_{\Delta_k \mathbf{y}} = \text{diag} [\mathbf{W}_{\Delta_k \mathbf{y}} \ \dots \ \mathbf{W}_{\Delta_k \mathbf{y}}], \\ \mathcal{K} &= \begin{bmatrix} \mathbf{0}_{n \times n(k_f-1)} & \mathbf{0}_{n \times n} \\ \mathbf{I}_{n(k_f-1) \times n(k_f-1)} & \mathbf{0}_{n(k_f-1) \times n} \end{bmatrix}, \quad \mathcal{L} = [\mathbf{I}_{n \times n} \ \mathbf{0}_{n \times n(k_f-1)}]^\top. \end{aligned}$$

Inserting (2.36) into (2.35) leads to²:

$$\mathcal{W}_{\Delta_k \mathbf{y}} ((\mathbf{I} - \mathcal{K}) \mathcal{Y}_k - \mathcal{L} \mathbf{y}_k) \leq \boldsymbol{\Omega}_{\Delta_k \mathbf{y}} T_s, \quad (2.37)$$

where inserting the condensed state-space model (2.18) yields

$$\mathcal{W}_{\Delta_k \mathbf{y}} (\mathbf{I} - \mathcal{K}) \mathcal{B} \mathbf{u}_k \leq \boldsymbol{\Omega}_{\Delta_k \mathbf{y}} T_s - \mathcal{W}_{\Delta_k \mathbf{y}} (\mathbf{I} - \mathcal{K}) \mathcal{G} \mathcal{D}_k - \mathcal{W}_{\Delta_k \mathbf{y}} ((\mathbf{I} - \mathcal{K}) \mathcal{A} - \mathcal{L}) \mathbf{y}_k. \quad (2.38)$$

The introduction of additional constraints also adds additional lines to the matrices $\mathbf{G}$, $\mathbf{e}$ and $\mathbf{E}$ (cf. (2.22)):

²The index denoting the size of the identity matrix $\mathbf{I}$ has been omitted below, as it has the same dimension as $\mathcal{K}$.

$$\begin{aligned}
 G &= \begin{bmatrix} \mathcal{W}_y \mathcal{B} \\ \mathcal{W}_{\Delta_k y} (I - \mathcal{K}) \mathcal{B} \\ \mathcal{W}_u \end{bmatrix} \quad e = \begin{bmatrix} \Omega_y - \mathcal{W}_y \mathcal{G} \mathcal{D}_k \\ \Omega_{\Delta_k y} T_s - \mathcal{W}_{\Delta_k y} (I - \mathcal{K}) \mathcal{G} \mathcal{D}_k \\ \Omega_u \end{bmatrix} \\
 E &= \begin{bmatrix} -\mathcal{W}_y \mathcal{A} \\ -\mathcal{W}_{\Delta_k y} ((I - \mathcal{K}) \mathcal{A} - \mathcal{L}) \\ 0 \end{bmatrix}.
 \end{aligned} \tag{2.39}$$

These matrices can then be used in (2.23) to calculate $\mathbf{G}_{\text{feas}}$ and $\mathbf{e}_{\text{feas}}$, defining the inequality conditions for a quadratic program (QP) solver within the safeguard to account for the additional capacitor constraints.

2.4.4 System Identification

As introduced in Sec. 1.1, a main advantage of RL is its model-free and self-adaptive behavior. To utilize the safeguarding strategy as described above, it is necessary to equip it with a system model. To sustain the advantage of inherent adaptivity and still allow operation without a priori parameter knowledge, these model parameters must be identified. Prior parameter knowledge thereby refers to the system's parameterization, i.e., the physical plant parameters as well as the abstract mathematical prediction model parameters. These are typically used in expert-driven controller designs (cf. [90–92]) but are assumed to be unavailable in the given scenario. Since these parameters principally vary from the data sheet values, there is an additional motivation for operating the plant without prior knowledge of parameterization. A certain amount of system knowledge is required, namely the type of topology and its physical limits must be clear in advance, which is common for control tasks in technical systems.

For an exemplary, fully measurable model with a single input:

$$\underbrace{\begin{bmatrix} y_{1,k+1} \\ y_{2,k+1} \end{bmatrix}}_{\mathbf{y}_{k+1}} = \underbrace{\begin{bmatrix} \hat{a}_{11,k} & \hat{a}_{12,k} \\ \hat{a}_{21,k} & \hat{a}_{22,k} \end{bmatrix}}_{\hat{\mathbf{A}}_{y,k}} \underbrace{\begin{bmatrix} y_{1,k} \\ y_{2,k} \end{bmatrix}}_{\mathbf{y}_k} + \underbrace{\begin{bmatrix} \hat{b}_{1,k} \\ \hat{b}_{2,k} \end{bmatrix}}_{\hat{\mathbf{b}}_{y,k}} u_k, \tag{2.40}$$

the recursive least squares (RLS) algorithm can be utilized as follows. Defining the regressor vector:

$$\boldsymbol{\epsilon}_{k-1} = \begin{bmatrix} y_{1,k-1} & y_{2,k-1} & u_{k-1} \end{bmatrix}^\top, \tag{2.41}$$

and the output measurements:

$$\zeta_{y_1,k} = y_{1,k}, \quad \zeta_{y_2,k} = y_{2,k}, \tag{2.42}$$

the model parameters searched for:

$$\hat{\mathbf{x}}_{y_1,k} = \begin{bmatrix} \hat{a}_{11,k} & \hat{a}_{12,k} & \hat{b}_{1,k} \end{bmatrix}^\top, \quad \hat{\mathbf{x}}_{y_2,k} = \begin{bmatrix} \hat{a}_{21,k} & \hat{a}_{22,k} & \hat{b}_{2,k} \end{bmatrix}^\top, \tag{2.43}$$

can be determined according to [93] via:

$$\begin{aligned}\boldsymbol{\kappa}_{k-1} &= \frac{\boldsymbol{\Gamma}_{k-1} \boldsymbol{\epsilon}_{k-1}}{\lambda_k + \boldsymbol{\epsilon}_{k-1}^\top \boldsymbol{\Gamma}_{k-1} \boldsymbol{\epsilon}_{k-1}}, \\ \hat{\boldsymbol{\chi}}_{y_1/y_2,k} &= \hat{\boldsymbol{\chi}}_{y_1/y_2,k-1} + \boldsymbol{\kappa}_{k-1} \left(\zeta_{y_1/y_2,k} - \boldsymbol{\epsilon}_{k-1}^\top \hat{\boldsymbol{\chi}}_{y_1/y_2,k-1} \right), \\ \boldsymbol{\Gamma}_k &= \frac{1}{\iota_k} \left(\boldsymbol{I} - \boldsymbol{\kappa}_{k-1} \boldsymbol{\epsilon}_{k-1}^\top \right) \boldsymbol{\Gamma}_{k-1}.\end{aligned}\tag{2.44}$$

Here, the parameter $\iota_k \in]0, 1]$ defines the forgetting factor. For values of $\iota_k < 1$, the estimation errors of the further past regressors are weighted lower than those of the more recent ones. $\boldsymbol{\Gamma} \sim \text{Cov}(\hat{\boldsymbol{\chi}}, \hat{\boldsymbol{\chi}})$ defines the scaled covariance matrix of the momentary parameter estimation. A priori knowledge can be included during the initialization of the covariance matrix $\boldsymbol{\Gamma}_0$ and the parameter vectors $\hat{\boldsymbol{\chi}}_0$.

2.5 Power Sharing

The goal of this thesis is the evaluation of the applicability of a data-driven controller for a voltage control task at component level in MGs. For an MG application, superimposed controllers are necessary for the construction of a decentralized MG, which will be briefly introduced in this section. The development of the latter is not the focus of this work and was adapted from preliminary work [94].

Droop control concepts are used to share the power demanded by the loads between the sources in an MG. The basic idea is to mimic the behavior of a synchronous generator, in order to share an increasing active power by decreasing frequency and increasing reactive power by decreasing voltage amplitude based on their droop characteristic defined via the control parameters K_P and K_Q [95]. As shown in [96], droop control with additional filtering of the power measurement is an approach to virtual inertia and can behave as a virtual synchronous generator (VSG). The droop control concept, shown in Fig. 2.7, is used in the following to calculate the voltage reference $\mathbf{v}_{abc,k}^*$ for the underlying voltage control. According to [46, 95, 97], the filter time constant is selected so that the cut-

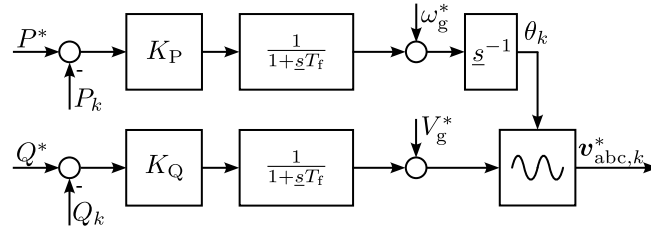


Fig. 2.7: Droop controller used to calculate the reference voltage per inverter (derived from [94]).

off frequency is one-tenth of the grid frequency ($T_f = 0.2\text{ s}$), while the droop gains are calculated as follows:

$$K_P = \frac{\Delta\omega_g}{P_{\max}}, \quad K_Q = \frac{\Delta V}{Q_{\max}}. \quad (2.45)$$

Here, $\Delta\omega_g$ and ΔV define the selected frequency and voltage deviation at maximum active or reactive power, respectively. Accordingly, no inverter or grid model is required for the design of the droop controller.

Independent control of active and reactive power is not possible if the reactance-to-resistance ratio (X/R) of the grid is low, as is typically the case in low-voltage MGs [98]. To counteract that, an inductive coupling between the inverters is realized by an inductor L_g located behind the LC filter in Fig. 2.1 (L_g not depicted here), connecting the inverter to the grid [94]. To form a stand-alone grid, a phase-locked loop (PLL)-based synchronization is used in the experiments, which evaluates whether an existing grid voltage is available for synchronization with the inverter output voltage $\mathbf{v}$ before connecting the inverter to the grid. If no grid voltage is available, the grid-forming inverter closes the switch and serves as a reference bus.

3 RL-Based Control of Voltage-Forming Grid Inverters

The fundamentals presented in Sec. 2 are combined into an RL-based controller for voltage-forming grid inverters, as shown in Fig. 3.1. As mentioned before, the RL agent should be trained through direct interaction with the hardware without simulated pre-training. Purely simulation-based investigations were conducted in [Own5, Own6, Co8, Co9] and are not the focus of this work.

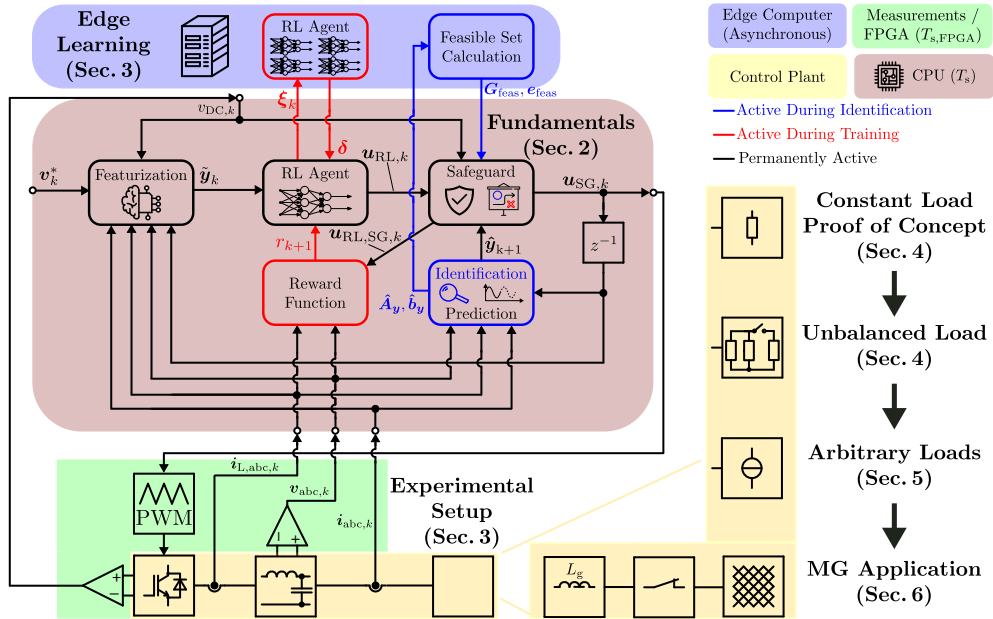


Fig. 3.1: Implemented reinforcement learning framework and structural overview of various application scenarios.

In this section, an edge learning toolchain is introduced transferring the fundamentals to allow a real-time capable application. The experimental setup is described beforehand, which is used as the plant to transfer the data-driven controller from a proof of concept with ohmic, passive loads in Sec. 4 to arbitrary loads in Sec. 5 in order to enable the appli-

cation of the developed RL-based controller in a decentralized MG application described in Sec. 6.

3.1 Experimental Setup

The experiments in this thesis are executed in the microgrid laboratory (MG-lab) at Paderborn University, which consists of 16 flexible inverter nodes [Co7]. The MG-lab can be used to emulate local energy grids, such as decentralized inverter-based MGs. The utilized hardware infrastructure is depicted in Fig. 3.2, showing two grid nodes of the MG-lab. Each node consists of a three-level, NPC inverter with a nominal power of 250 kVA per inverter with a configurable output filter taking either L -, LC - or LCL -type form. For the investigated control task, one inverter – source (cf. Fig. 3.2) – is used to implement a voltage-forming grid inverter.

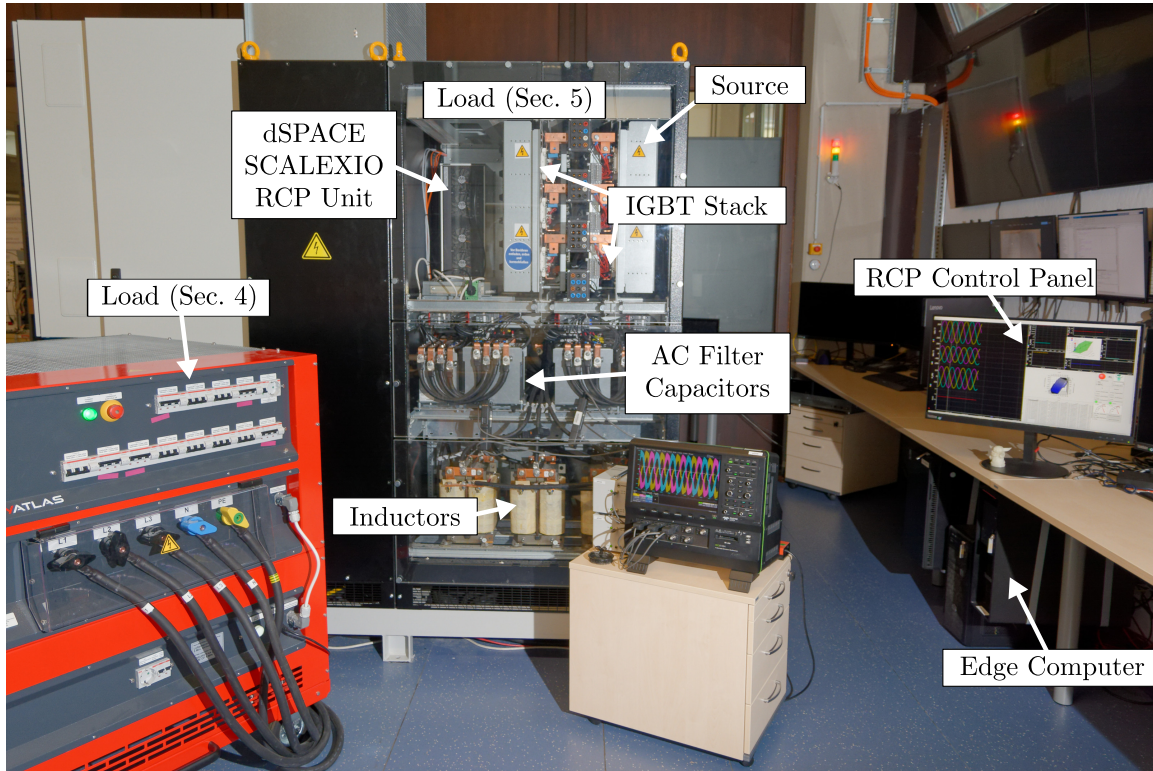


Fig. 3.2: Experimental setup in the MG-lab, showing two 250 kVA industrial-grade inverters connected back-to-back and a resistor, which can be used as a passive load.

In the following, an RL-based controller is to be trained on this hardware to supply the shown passive, ohmic load, described in Sec. 4. The control concept is then extended to arbitrary loads, where the other inverter shown in Fig. 3.2 serves as the active load, starting in Sec. 5.

The controllers under investigation are implemented on the DS6001 CPU-card of a dSPACE SCALEXIO RCP unit and executed once per switching period with a switching frequency of $f_s = 10$ kHz, resulting in a sampling time of $T_s = 100$ μ s. The controller output corresponds to the phase voltages at the filter input, which is realized via a three-level PWM, implemented in the DS6601 field programmable gate array (FPGA) card of the RCP unit. The PWM generates the switching commands for the insulated gate bipolar transistors (IGBTs). The IGBT-stacks consist of three Infineon F3L300R07PE4 modules per phase, connected in parallel. In order to avoid short circuits between the three voltage potentials, an interlock deadtime of 2 μ s between the individual switching states is ensured by the driver hardware. Further, due to the parallel connection of the IGBT modules, their parasitic capacitances lead to significant charge and discharge durations up to 1 μ s at low absolute current. These non-ideal switching effects are mostly compensated by lookup-table-based repositioning of the edges of the switching signals inside the PWM implementation in the FPGA, which was not part of this thesis and has been implemented in preliminary work. Nevertheless, due to the decreased reliability of current sensors when operated far from nominal operation, precise compensation of inverter non-linearity at low current is hardly possible with analytic methods.

The measurements of the required currents and voltages are recorded by the RCP unit's analog-to-digital converters (ADCs) at a sampling rate of 5 MHz. In the following applications, the FPGA code triggers the controller implementation on the CPU synchronized with the PWM once per switching period, handing over the measurement values to the CPU. A regular sampling based on the inductor current i_L is used. The mathematical fundamentals of the used inverter are described in Sec. 2.2, where the corresponding circuit diagram is shown in Fig. 2.1. The controllers used in the following are implemented in MATLAB/Simulink and afterwards build on the RCP units. The ANNs for the RL-based controller are implemented on the RCP unit's CPU by means of standard blocks such as sums and product blocks with the ANN's weights, as well as the transfer function by means of comparator blocks. Xilinx was used for the FPGA-based implementation of, e.g., the PWM. All hardware specifications are listed in Tab. 3.1. Here, the maximum values (e.g., $V_{\max}$) are used within the calculations of the feasible set since they define the permanent operation range of the system. The limit values (e.g., $V_{\lim}$) refer to the physical limitations, while exceeding them can cause hardware damage or may provoke emergency shutdowns. Temporary operation between these two limits is generally possible. However, to prevent this from leading to thermal overload of the system, for example, the proposed safeguard concept would need to be expanded, as discussed in more detail in Sec. 7, while the overload range will not be considered further in this thesis.

Tab. 3.1: Specifications of test hardware, physical limitations and experiment setting

IGBT modules		F3L300R07PE4
Number modules per phase		3
Topology		3-level NPC
Control hardware		dSPACE SCALEXIO (DS6001, DS6601)
Edge computer		AMD Ryzen Threadripper PRO 3955WX
Inductor current safety limit	I_{lim}	700 A
Maximum inductor current	$I_{\text{L,max}}$	600 A
Capacitor voltage safety limit	V_{lim}	400 V
Maximum capacitor voltage	V_{max}	365 V
Maximum capacitor current	$I_{\text{C,max}}$	90 A
V_{DC} scaling maximum	$V_{\text{DC,max}}$	410 V
V_{DC} scaling minimum	$V_{\text{DC,min}}$	330 V
Droop frequency deviation	$\Delta\omega_{\text{g}}$	$2\pi\text{s}^{-1}$
Droop voltage deviation	ΔV	1 V
Maximum active power	P_{max}	250 kW
Maximum reactive power	Q_{max}	250 kVA
Switching frequency	f_{s}	10 kHz
Grid frequency	f_{g}	50 Hz
DC supply voltage	v_{DC}	740 V
Upper/lower DC supply voltage	$v_{\text{DC+/-}}$	370 V
Filter inductance	L_{f}	70 μH
Inductor series resistance	R_{sL}	1.1 m Ω
Filter capacitance	C_{f}	300 μF
Capacitor series resistance	R_{sC}	7 m Ω
Grid-side inductance	L_{g}	70 μH

3.2 Digital Control Delay Compensation

The evaluation of a digital controller needs some amount of time when implemented on a real-world control system. That means it is not possible to apply the controller output u_k (cf. Fig. 3.1) to the plant within the same time step k . The execution of the controller output towards the plant will happen with a delay of one sampling period, as depicted in Fig. 3.3. Therefore, a control delay compensation is required during the safeguarding. For this, a prediction of the state $\hat{y}_{k+1}$ is needed to evaluate the chosen action with regard to safety. The identified system matrices $\hat{A}_y$ and $\hat{b}_y$ are utilized for that prediction, like depicted in Fig. 3.1.

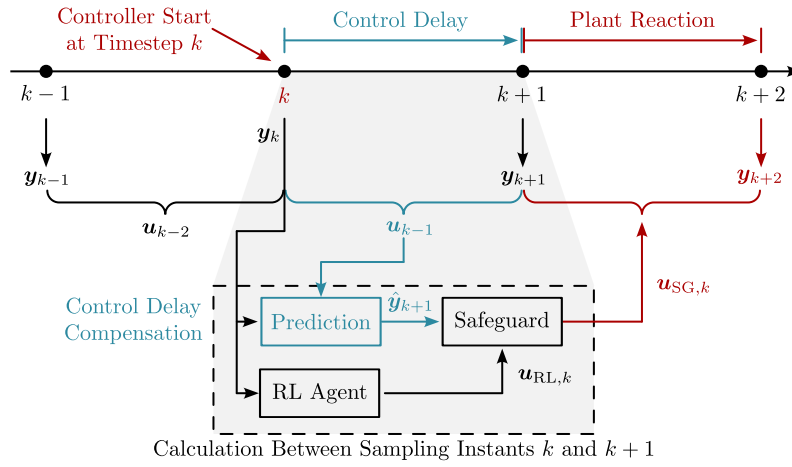


Fig. 3.3: Digital control delay compensation during the safeguarding (derived from [Own1]).

3.3 Reward Design

As introduced in Sec. 2.3, the control task is encoded within the reward function, rating each action chosen by the agent and the resulting state. To learn the reference tracking task in the presence of disturbance, the proposed reward (cf. Fig. 3.1) is maximized if the deviation of the measured voltage from the reference is zero:

$$r = \begin{cases} \max\left(\frac{1-\gamma}{3} \left(\sum_{p \in \{a,b,c\}} \left(1 - \sqrt{\frac{|v_p^* - v_p|}{2V_{\max}}} - \frac{|u_{RL,p} - u_{SG,p}|}{2} \right) \right), 0\right), & \text{if } \textcircled{a} \\ -1, & \text{otherwise,} \end{cases} \quad (3.1)$$

with

$$\textcircled{a} = |v_{\{a,b,c\}}| \leq V_{\lim} \wedge |i_{L,\{a,b,c\}}| \leq I_{\lim}. \quad (3.2)$$

If there is no setpoint deviation, the agent receives the maximum, undiscounted reward of $r = 1$ as feedback. In case of a setpoint deviation, the reward is reduced down to

$r = 0$ if operated at the maximum allowed deviation (cf. Tab. 3.1). The mean root error (MRE) is used, because this function is steeper near the targeted reference than, for example, mean squared error (MSE). The system limits are taken into account within the reward function via a penalty term in the reward design. If the current or voltage limit is exceeded (cf. Tab. 3.1), $r_{k+1} = -1$ is returned. The undiscounted reward depicted in Fig. 3.4 is therefore defined in a value range of $r \in [-1, 1]$ and defines the optimal control task (cf. (2.1), (2.13) and (2.14)).

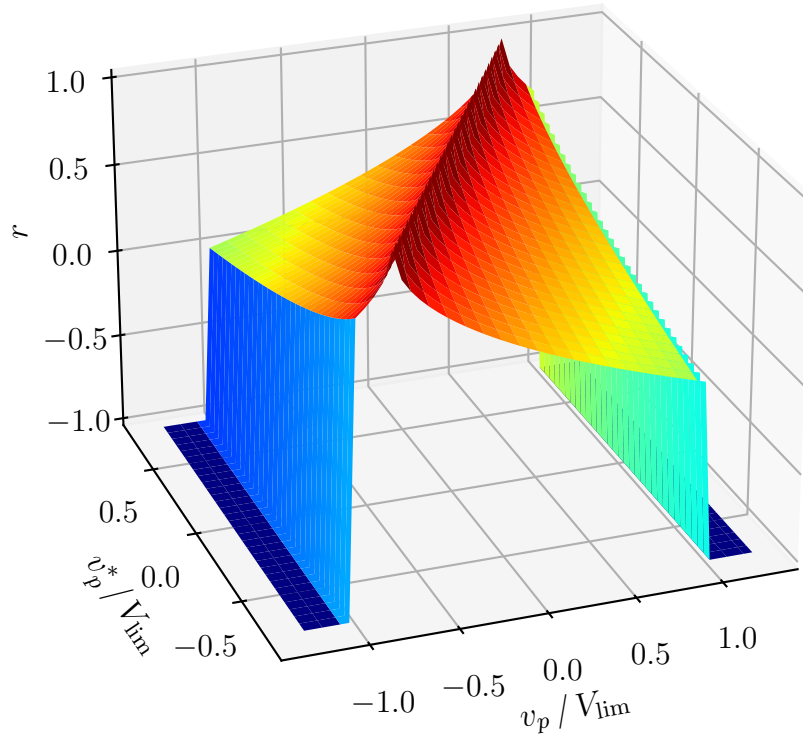


Fig. 3.4: Graphical representation of the undiscounted, single-phase reward function defined in (3.1) without taking the penalty of the safeguard intervention into account (with $p \in \{a, b, c\}$).

As described in Sec. A.1, one ANN (called the critic) is used during training to approximate the state-action value defined by the Bellman equation (2.14). This estimation is realized using ANNs (cf. Sec. 2.3.1), where it is advisable to normalize the inputs and outputs such that all values have comparable magnitudes [99]. In order to normalize the return to a range of $g \in [-1, 1]$, the reward design takes the discount factor gamma into consideration, such that the maximum return over an infinite time horizon equals one [22]. To enable a learned control policy without further safeguard interventions after the training phase, a penalty term is defined for these safeguard interventions during training,

as shown in (3.1), which increases linearly with the difference between the action chosen by the agent u_{RL} and the safe action determined via the safeguard u_{SG} .

3.4 Featurization

As depicted in Fig. 3.1 and defined in (3.3), the measurement $\mathbf{y}_k$ is augmented by additional feature engineering. This can improve the learning process and can be required to approximate the Markov property, as described in Sec. 2.3. Here, expert knowledge can be used to add or extract required or additional information, such as the reference values, past measurement values or the tracking error. The featurized measurement vector $\tilde{\mathbf{y}}_k$ is then provided to the agent, on whose basis the next control output is determined [78]. In the following application, the featurized measurement vector is defined as follows:

$$\tilde{\mathbf{y}}_k = \begin{bmatrix} \frac{\mathbf{i}_{\text{L,dq0/abc},k}^T}{I_{\text{lim}}} & \frac{\mathbf{v}_{\text{dq0/abc},k}^T}{V_{\text{lim}}} & \frac{(\mathbf{v}_{\text{dq0/abc},k}^*)^T}{V_{\text{lim}}} & \frac{\frac{1}{2}\Delta\mathbf{v}_{\text{dq0/abc},k}^T}{V_{\text{lim}}} & \frac{\mathbf{u}_{\text{SG},k-1}^T}{V_{\text{DC+/-},k}} \\ \frac{\mathbf{v}_{\text{dq0/abc},k-1}^T}{V_{\text{lim}}} & \frac{\mathbf{i}_{\text{dq0/abc},k}^T}{I_{\text{lim}}} & \frac{2(v_{\text{DC+/-},k} - V_{\text{DC,min}})}{V_{\text{DC,max}} - V_{\text{DC,min}}} - 1 \end{bmatrix}^T. \quad (3.3)$$

Here, the error between reference and measured voltages is defined by $\Delta\mathbf{v} = (\mathbf{v}^* - \mathbf{v})$, and all measurements are normalized to a range of $[-1, 1]$ by their limit values, which is recommended when using ANNs [99]. For the same reason, a min-max scaling is applied to normalize the measured voltage $v_{\text{DC+}}$, of the upper and $v_{\text{DC-}}$, of the lower DC-capacitor for a voltage deviation of ± 40 V (cf. Tab. 3.1). The DC-link voltage limits (e.g., $V_{\text{DC,max}}$) are not to be understood as physical limits here, but serve for normalization purposes. Their selection is based on the maximum observed DC-link voltage deviation during training. It should be noted that the specific form of the observation vector in (3.3) has been found via heuristic feature engineering during pre-investigations. Although this configuration leads to favorable control performance results, as will be shown in the upcoming applications, other observation vector compositions might result in an even better outcome. The selection of a holistically optimized feature set for the considered and related converter control applications is, however, not within the scope of this investigation.

Depending on the experiment, the measured variables are provided either in the fixed abc reference frame or in the rotating dq0 reference frame [91]. Therefore, the magnitude invariant transformation and the corresponding back-transformation with the phase angle of the grid voltage θ is considered:

$$\mathbf{T}_{\text{dq0}}(\theta) = \frac{2}{3} \begin{bmatrix} \cos(\theta) & \cos\left(\theta - \frac{2\pi}{3}\right) & \cos\left(\theta - \frac{4\pi}{3}\right) \\ -\sin(\theta) & -\sin\left(\theta - \frac{2\pi}{3}\right) & -\sin\left(\theta - \frac{4\pi}{3}\right) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix}, \quad (3.4)$$

which defines the relation between physical abc quantities in a fixed reference frame, and virtual dq0 quantities in a rotating reference frame:

$$\mathbf{x}_{dq0} = \mathbf{T}_{dq0}(\theta)\mathbf{x}_{abc}, \quad \mathbf{x}_{abc} = \mathbf{T}_{dq0}^{-1}(\theta)\mathbf{x}_{dq0}. \quad (3.5)$$

Here, $\mathbf{x}$ can be, e.g., the measured currents or voltages.

3.5 Training via Edge Computing

The training of the RL agent is not executed on the RCP unit's CPU, where only the actor's ANN $\hat{\boldsymbol{\mu}}_{\boldsymbol{\delta}}$ and the action noise need to be implemented for the control application. The tuning of the controller parameters defined by the actor's weights $\boldsymbol{\delta}$ is outsourced to an external workstation via edge computing. Therefore, a toolchain that is oriented at [100] is set up, where the experience $\boldsymbol{\xi}_k$ defined in (2.16) is sent to the edge device via Ethernet at each control cycle step (cf. Fig. 3.1). The communication is realized via User Datagram Protocol (UDP) to minimize latencies. The DDPG training algorithm described in Sec. A.1 runs on the workstation based on the received experiences. After each training step, the updated actor parameters $\boldsymbol{\delta}$ are sent back to the RCP unit. Hence, only the forward path of $\hat{\boldsymbol{\mu}}_{\boldsymbol{\delta}}$ has to be real-time capable (cf. Sec. 3.1), while the parameter updates during training can be conducted asynchronously on the edge device. To decouple the tasks of communicating parameters and measurements from the training of the ANNs on the edge device, they are separated into different threads. This enables asynchronous execution, which is desirable because a training cycle usually lasts longer than a controller cycle of the RCP unit. In this thesis, the DDPG implementation of the Stable Baselines3 Python library [101] is utilized in the training processes.

3.6 Hyperparameter Optimization in Real-Time Application

In order to train the RL agent for the task defined in Sec. 2.1, it must first be configured. Since, as described in Sec. 2.3.1, ANNs are used as function approximators. The number of layers that make up the respective ANN and the number of neurons per layer must first be selected. As described in [78, 83, 84], each neuron is defined not only by the network weights of the inputs but also by an activation function for the summed inputs, which represents another important degree of freedom. The neural network used for the real-time control task should be trained based on experiences (2.16) so that it delivers the best action for the current measurement according to (2.15). During the training process, a numerical optimizer is used to change the network weights in such a way that the cost function (A.3) is maximized. As introduced in Sec. 2.3.1, this requires another ANN-based function approximator for the q -values to approximate (2.14). This ANN is also trained using an optimizer, which aims to minimize the cost function (A.1) based on the experiences (2.16). The step sizes – the learning rates – of these optimizers are further important degrees of freedom that must be parameterized before training. In addition to the most important parameters already described, there are many others that must be configured before training. A list of all parameters is provided in Tab. A.1. These

parameters are referred to as hyperparameters (HPs), because they must be selected before optimizing the ANNs for the task encoded by the reward. Since selecting these parameters is a task-dependent, non-linear task, a superordinate optimization process called hyperparameter optimization (HPO) is used to determine these parameters.

Within the HPO, the experiment – training the RL agent for voltage control – is carried out several times with different HP settings. The experiments during the HPO are real-time simulations on the RCP units that replicate the targeted real-world experiment for selecting the best HPs. After completing the training process in one of the experiments, the HP set (chosen number of neurons, learning rate, ...) is validated using a central performance metric. To do so, the trained controller is evaluated using a standardized test scenario. The metric used for this is the averaged, undiscounted return g_{test} according to (2.13). The return depends on the reward design according to Sec. 3.3. The reward can take on a maximum value of 1, which means that for the experiment considered here, there is no deviation between the controlled voltage and the reference value. Thus, a value of $g_{\text{test}} = 1$ means that the trained controller parameterized with the chosen HP set always reached the required voltage in the test case. Since this is not always possible for physical reasons in the event of load or reference steps, $g_{\text{test}} = 1$ often cannot be achieved depending on the test case. This rating via g_{test} is then used in the top level hyperparameter optimizer to select the next set of HPs.

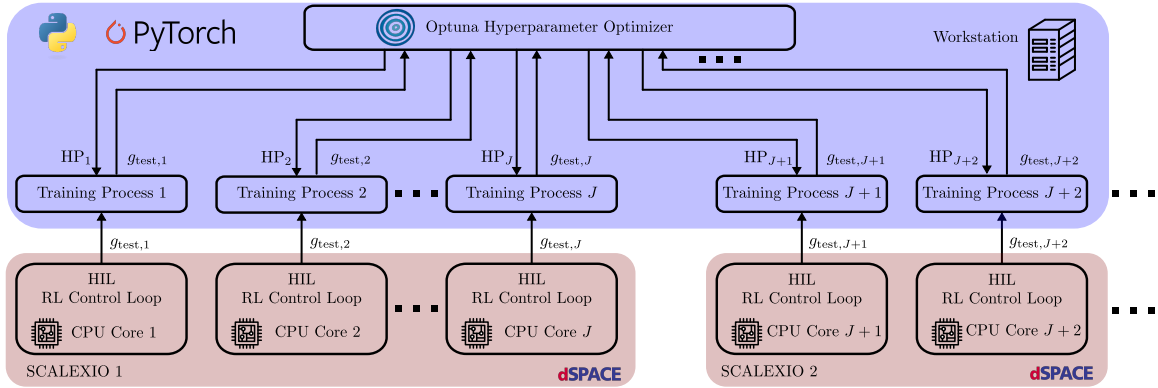


Fig. 3.5: Distributed HPO framework: utilizing parallelized HIL environments for real-time hyperparameter tuning on ten RCP units with more than fifty cores in total. Only the communication needed for the HPO is shown, while the training communication (cf. Fig. 3.1) is omitted for clarity.

In a sequential simulation framework, which is the established approach to HPO (cf. [Own5]), a training step is performed after each control cycle. In such a case, both the simulation of the environment and the training of the RL-based controller’s ANNs take place on the same computing hardware. This often involves high-performance computing (HPC) clusters, where the calculations benefit from both the performance and high potential for parallelization of the hardware. This assumption does not hold in the targeted real-time application, since the learning process runs asynchronously from the control

cycle executed on the RCP unit (cf. Sec. 3.5). Therefore, a new HPO framework based on a distributed, real-time HIL environment is presented in the following. This approach takes the asynchronous fashion of edge learning into account. The basic framework has originally been published in [Own1] and is depicted in Fig. 3.5.

Here, three different levels are visible. The top level contains the HP optimizer, which is executed on a workstation – in this thesis, this will be the edge computer depicted in Fig. 3.2 (cf. Tab. 3.1). Here, the HPO framework Optuna [102] is utilized to suggest the next HPs. The determined HPs are used to configure the individual training processes. The training, i.e., the execution of the learning algorithm, takes place on the edge computer, while the actual control routine runs on the real-time capable embedded hardware. The training process introduced in Sec. 3.5 communicates with a HIL real-time control loop implemented on a single core of one SCALEXIO RCP unit. Each experiment is identical to a real-time HIL environment utilizing the model introduced in Sec. 2.2. Thereby, the details of the training and the communication required for the individual training are omitted in Fig. 3.5 in favor of readability, but are similar to the stand-alone deployment in Fig. 3.1. The only difference is that the control plant is simulated on the RCP units. After the first training process with the first chosen set of HPs (HP_1 , cf. Fig. 3.5) is started, the next training process starts with the next set of HPs (HP_2) determined by the Optuna optimizer. The experiment for the second training process is executed on the second CPU core of the same SCALEXIO RCP unit (CPU core 2, cf. Fig. 3.5). If there are no further free CPU cores on the RCP unit available, the next SCALEXIO system of the MG-lab is utilized. This can be continued until either no more free RCP unit with free cores is available or until a capacity limit of the edge computer (e.g., CPU or RAM) is reached.¹ When a training is completed, the above mentioned, deterministic test case is rolled out, and the undiscounted, average return g_{test} is sent back to the Optuna optimizer to rate the corresponding HP sample. Based on this performance metric, the optimizer then chooses the next HPs to start the next training process and experiment on the freed-up CPU core.

In order to run the training on the RCP unit with different HPs, additional attention must be paid to the implementation of the ANN on the RCP unit used for control, which was realized to allow a variable architecture (i.e., parameterizable number of hidden layers and number of neurons per layer) without recompilation of the real-time software. The communication depicted in Fig. 3.5 is realized via Ethernet utilizing UDP, comparable to the edge learning toolchain described in Sec. 3.5. To allow parallelization of the different training processes, each training is executed in a new Python process utilizing multiprocessing. Since further details such as test case and search space for the respective HPO are application-dependent, they are described in the respective application chapters.

¹The data from the Optuna optimizer is stored in an SQL-based database. In such cases, it is possible to use another edge computer that has access to the MG-lab's Ethernet and the database to train additional RL agents for the HPO. However, only one edge computer was used in the following.

4 Application: Constant, Resistive Load

In order to prove the concept of the methods presented in Sec. 2, the RL-based controller as shown in Fig. 4.1 is utilized to learn a voltage control task for a grid-forming inverter using the reward defined in (3.1). For this purpose, the passive, resistive load shown in Fig. 3.2 is used. The given setup is to be utilized for both the investigation of the safeguarding concept and the performance comparison of the developed RL-based controller with a related optimal control method – MPC. The passive load ensures a fair head-to-head comparison, excluding the influence of active loads and their control on the grid.

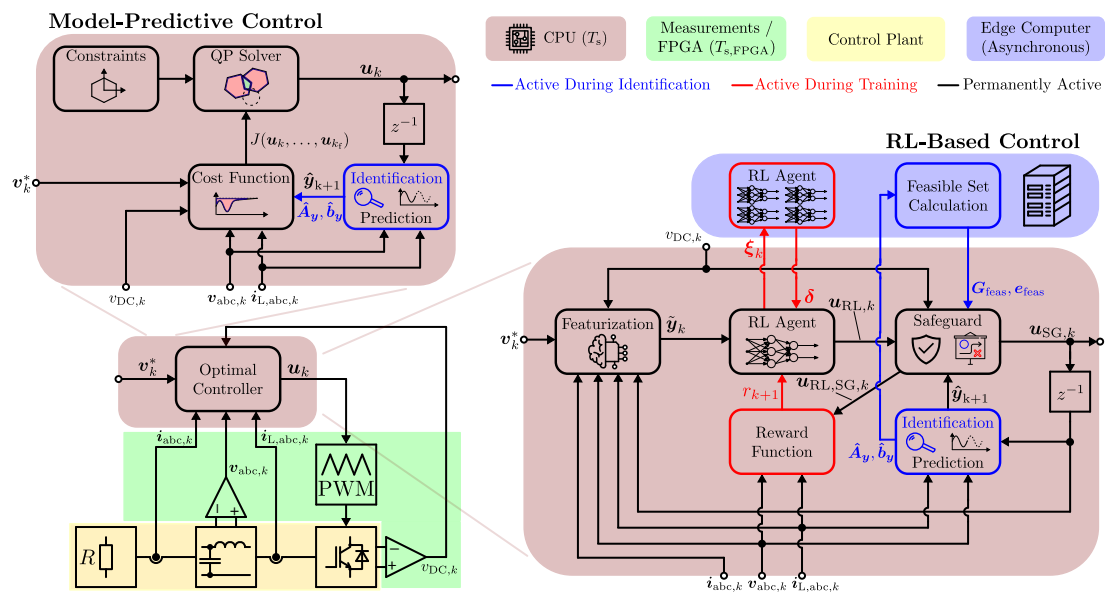


Fig. 4.1: Control plant and control structure of the investigated RL-based controller and the MPC, derived from [Own1].

The results were originally published in [Own1], where the MPC was developed and implemented by Dominik Schmies and the MPC experiments were performed on the same hardware as the RL experiments. The theory and design of MPC are not within the scope

of this work and the results from [Own1] are herein merely considered as a comparison benchmark.

4.1 State-Space Representation of the Plant

The experiment examined in this chapter is shown in Fig. 4.1. The hardware specifications according to Tab. 3.1 apply, except for the limitation regarding the inductor current. The current limits were reduced to $I_{\text{lim}} = 300 \text{ A}$ and $I_{\text{L,max}} = 250 \text{ A}$ for this proof of concept. The load resistor was configured to $R_p = 2 \Omega$ phase to neutral, with $p \in \{a, b, c\}$. Since the load resistance is constant in the following, the resulting load current, which would normally be considered as a disturbance variable, can be taken into account by considering the load resistance as part of the state space description. Thus, the disturbance variable depends directly on the measured output voltage, resulting in a simplified, single-phase model according to (2.2):

$$\frac{d}{dt} \mathbf{x}_s(t) = \mathbf{A}_s \mathbf{x}_s(t) + \mathbf{b}_s u_s(t), \quad \mathbf{y}_s(t) = \mathbf{C}_s \mathbf{x}_s(t), \quad (4.1)$$

where

$$\mathbf{x}_s = \begin{bmatrix} i_L \\ v_c \end{bmatrix}, \quad \mathbf{y}_s = \begin{bmatrix} i_L \\ v \end{bmatrix}, \quad \mathbf{A}_s = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \quad \mathbf{b}_s = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \quad \mathbf{C}_s = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix},$$

with

$$\begin{aligned} a_{11} &= -\frac{R_{sL}}{L_f} - \frac{R_{sC}R}{L_f(R_{sC} + R)}, & a_{12} &= -\frac{1}{L_f} + \frac{R_{sC}}{L(R_{sC} + R)}, \\ a_{21} &= \frac{R}{C_f(R_{sC} + R)}, & a_{22} &= -\frac{1}{C_f(R_{sC} + R)}, \\ b_1 &= \frac{1}{L_f}, & b_2 &= 0, \\ c_{11} &= 1, & c_{12} &= 0, \\ c_{21} &= \frac{R_{sC}R}{R_{sC} + R}, & c_{22} &= \frac{R}{R_{sC} + R}. \end{aligned}$$

Accordingly, the measurable state-action space is three-dimensional, which allows a clear graphical representation of the safeguard for validation purposes. The required time-discrete model can be calculated according to (2.3) – without a disturbance variable – using:

$$\mathbf{A} = e^{\mathbf{A}_s T_s}, \quad \mathbf{b} = (\mathbf{A} - \mathbf{I})\mathbf{A}_s^{-1}\mathbf{b}_s, \quad \mathbf{C} = \mathbf{C}_s. \quad (4.2)$$

The resulting discrete LTI state-space model has the form:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{b}u_k \quad (4.3)$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k. \quad (4.4)$$

When deriving the dynamic equation with respect to the measured variable, no prediction regarding the disturbance variable is necessary:

$$\mathbf{y}_{k+1} = \underbrace{\mathbf{C}\mathbf{A}\mathbf{C}^{-1}}_{\mathbf{A}_y} \mathbf{y}_k + \underbrace{\mathbf{C}\mathbf{b}}_{\mathbf{b}_y} u_k. \quad (4.5)$$

4.2 Safeguard in Real Time

To derive the constraints necessary for the safeguard in the application depicted in Fig. 4.1, the model established in (4.5) is used to calculate the matrices $\mathbf{G}_{\text{feas}}$ and $\mathbf{e}_{\text{feas}}$. For this, the theory presented in Sec. 2.4.1 starting from (2.18) is used. It should be noted that the term for the disturbance variable is not used at this point, as explained above. To calculate the feasible set, the extension regarding the capacitor current limit in (2.39) introduced in Sec. 2.4.3 is used. Utilizing the data sheet values defined in Tab. 3.1 for the state space model (4.1), algorithm 1 converges after two steps. The resulting feasible state-action space for a single phase of the plant system is depicted in Fig. 4.2.

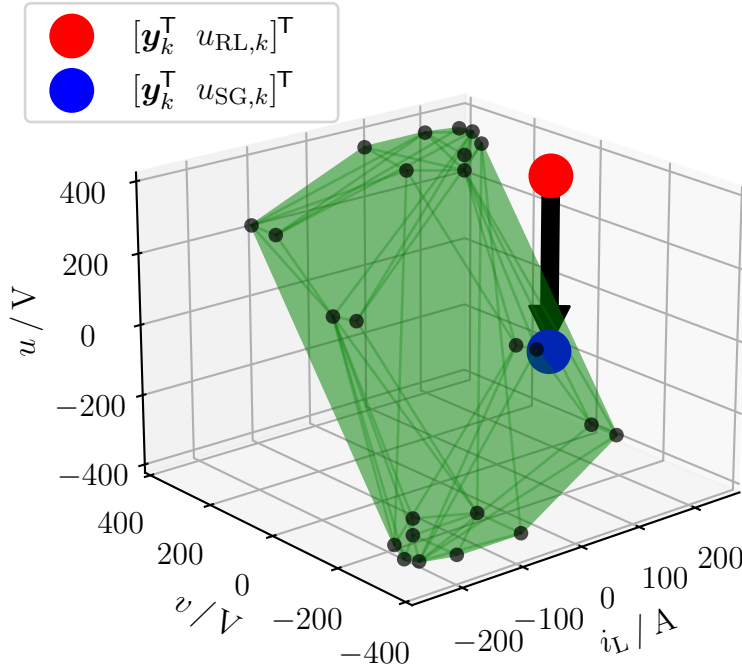


Fig. 4.2: Feasible set $\mathbb{F}_{\infty \rightarrow 0}$ for the voltage-forming grid inverter with LC filter and resistive load defined in (4.1) in green. A safeguard intervention is showcased for the case of $\mathbf{y}_k^T = [180 \text{ A} \quad -20 \text{ V}]$.

As the chosen action $u_{\text{RL},k}$ in Fig. 4.2 for the specific phase would lead to harmful system states in future steps (red point), the action is altered to $u_{\text{SG},k}$ (blue point) such that the constraints are satisfied. Here, the red point represents the state-action tuple:

$$[\mathbf{y}_k^\top \ u_k]^\top = [i_{L,k} \ v_k \ u_k]^\top = [180 \text{ A} \quad -20 \text{ V} \quad 370 \text{ V}]^\top. \quad (4.6)$$

Utilizing the model defined in the previous section, the inductor current is predicted to be $i_{L,k+1} = 606 \text{ A}$ in the next time step, which is more than twice the configured current limit $I_{\text{lim}} = 300 \text{ A}$. As introduced in Sec. 2.4.1, all state-action tuples lying inside the green shaded feasible set avoid future limit violations. Since the momentary measurement in Fig. 4.2 is fixed the only degree of freedom is the voltage at the filter input, which is the control variable. The voltage $u_{\text{RL},k}$ selected by the agent is too high in the exemplary scenario. In order to change the RL agent's choice as little as feasibly necessary, the following convex cost function is to be minimized (cf. (2.25)):

$$\begin{aligned} u_{\text{SG},k} &= \arg \min_{u_k, \vartheta} (u_k - u_{\text{RL},k})^2 + \psi \vartheta, \\ \text{s.t.} \quad & \begin{bmatrix} \mathbf{G}_{\text{feas}} & -\mathbf{1}_{l \times 1} \\ \mathbf{0}_{1 \times (n+m)} & -1 \end{bmatrix} \begin{bmatrix} \mathbf{y}_k \\ u \\ \vartheta \end{bmatrix} \leq \begin{bmatrix} \mathbf{e}_{\text{feas}} \\ 0 \end{bmatrix}. \end{aligned} \quad (4.7)$$

If, e.g., due to measurement noise, sensor or model deviations, the measured state $\mathbf{y}_k$ leaves the feasible state space $\mathbb{Y}_{\infty \rightarrow 0}$ (cf. Sec. 2.4.1), there is no feasible solution for u_k to (4.7). To find the best possible trade-off in such a case, a slack variable ϑ that may soften the constraints, is introduced for each phase [103]. Here, $\psi > 0$ is a punishment factor that corresponds to softening the feasibility conditions (4.7) by means of the slack variable ϑ . $\mathbf{1}$ denotes a vector of ones with the specified size and l defines the number of lines in $\mathbf{G}_{\text{feas}}$. Additionally, the matrices $\mathbf{G}_{\text{feas}}$ and $\mathbf{e}_{\text{feas}}$ have to be extended to ensure that the slack variable stays positive as shown in (4.7) (cf. [103]). In this way, a solution for $u_{\text{SG},k}$ is defined even if the feasible state space is exited, e.g., due to measurement noise. If the punishment factor ψ is chosen to be higher than the maximum possible feasible cost, safety can be ensured during the whole training process of the RL agent as presented in the upcoming sections. Applying (4.7) to (4.6) results in the blue point $[\mathbf{y}_k^\top \ u_{\text{SG},k}]^\top$, which lies within the feasible set. By reducing the voltage applied to the filter down to $u_{\text{SG},k} = -128 \text{ V}$, the overcurrent can be avoided and (2.31) is satisfied, which allows a feasible solution in the following time steps. The optimization can be calculated by means of a standard QP solver [4, 104]. If the feasible set does not change during the experiment, it can be approximated by a look-up table (LUT) calculating the intersection points with the surface of the feasible set for a specific state. Both variants were successfully validated in the application examined later. For the application example described in this chapter, while implicitly taking the disturbance variable into account, using a LUT instead of a solver can result in computing time advantages. This depends on the complexity of the problem defined in (4.7). In the task considered in the next chapter, the variable load current adds another dimension to the required LUT, which in turn makes its use unattractive for reasons of computing time.

4.3 Data-Driven Safeguard

As introduced in Sec. 2.4.4, system identification is used to sustain the RL's advantage of inherent adaptivity and independence of any a priori system parameter knowledge. The recursive least squares (RLS) (cf. (2.44)) is used in the following to identify the model described in Sec. 4.1 which is required for the safeguard. As depicted in Fig. 4.3, an excitation signal is fed into the plant in a real-world experiment before the training to identify the required linear state-space model. Since no a priori parameter knowledge

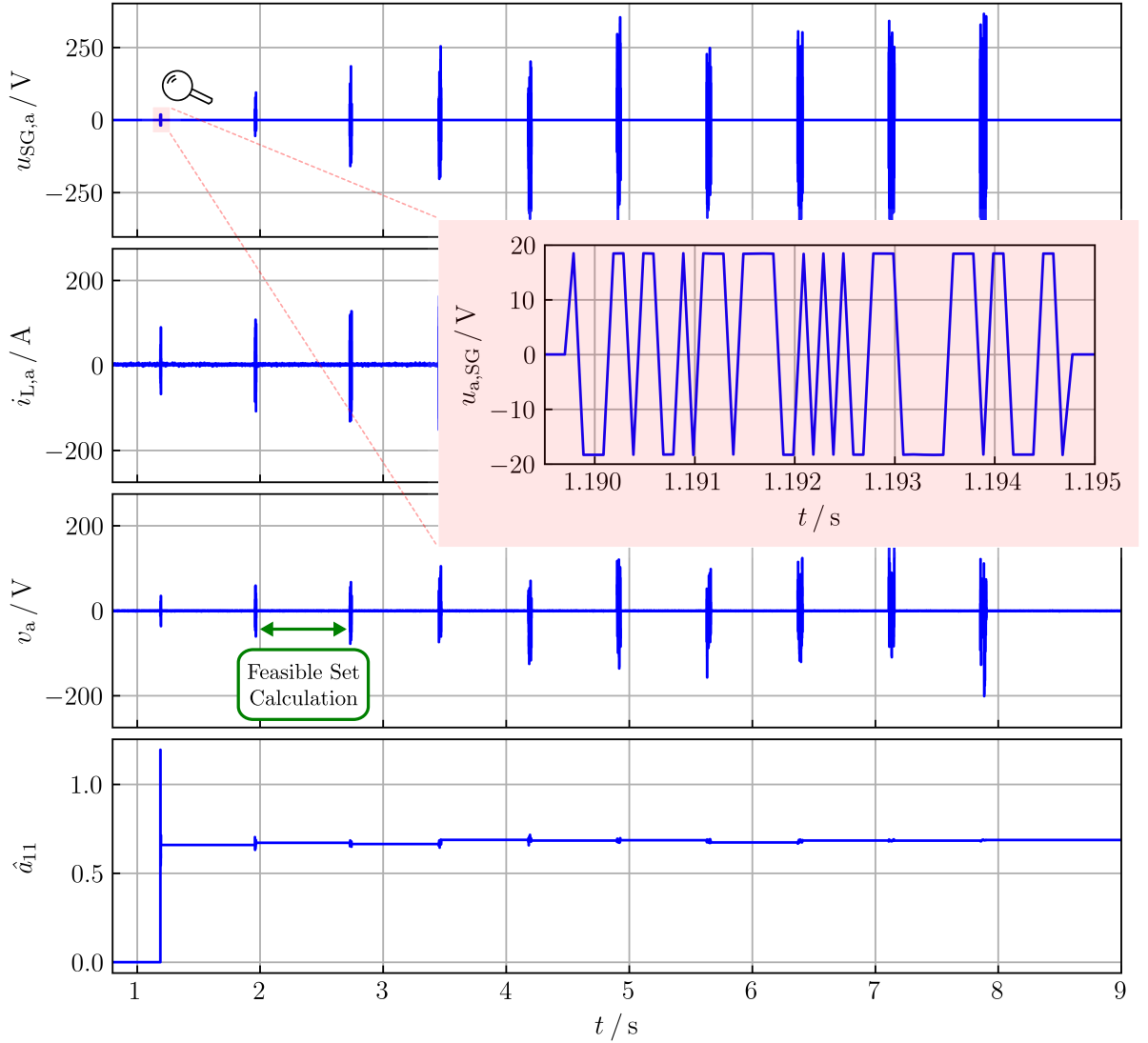


Fig. 4.3: Measured system excitation process in a real-world experiment with pulsating noise inputs for phase a.

is assumed, no safeguard is available for the first excitation signal. It is assumed that a modulation of 5 % of the control voltage can be applied for 50 steps with almost alternating

sign based on an amplitude-modified pseudorandom binary sequence (PRBS) (cf. zoom in Fig. 4.3) without damaging the system. This assumption can be scaled down for systems with higher safety requirements. The amplitude (first pulse: 5%) and length (first pulse: 50 steps) of the pulse is scaled up linearly over the course of the identification process. Similarly, the forgetting factor (cf. Sec. 2.4.4) is linearly scaled from 0.9 to 1 during the identification. The RLS to identify the LTI model is implemented on the CPU of the RCP unit. As shown in Fig. 4.3, the calculation of the feasible set takes more than half a second. Therefore, the identification and calculation of the safeguard is performed prior to the start of training because otherwise, the exploratory behavior of the RL-based controller could already cause an unstable system state within this time. Since the model is not expected to change over time, this is sufficient. If that is not the case, a continuous identification can be done in parallel to the learning process. To not

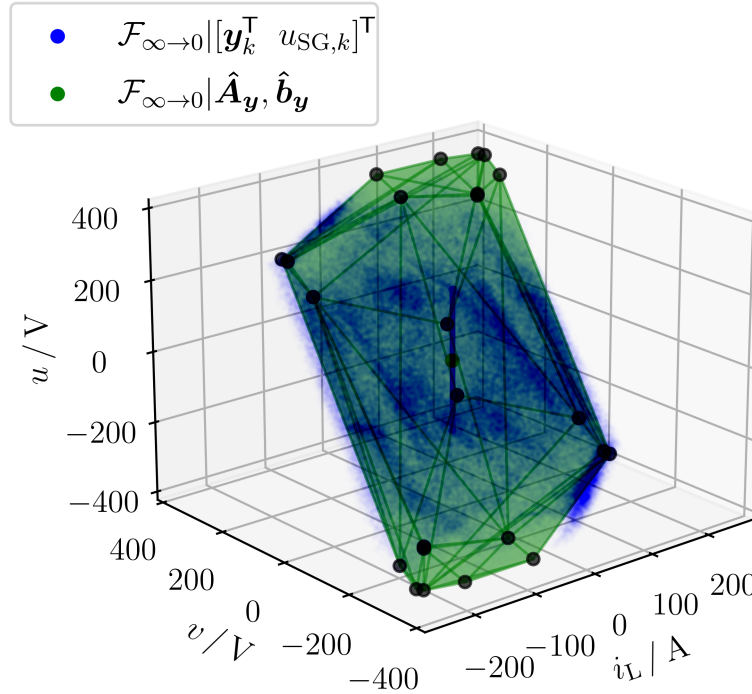


Fig. 4.4: Feasible set $\mathbb{F}_{\infty \rightarrow 0}$ based on the identified matrices in green and measurement points in blue resulting from random control outputs to evaluate the functionality of the safeguard.

take a priori system parameter knowledge into account, the initial model parameters are set to zero as depicted for $\hat{a}_{11}$ (cf. (4.5)) in Fig. 4.3, where the parameter evaluation during the identification process is visible. As shown in Fig. 4.1, the identified model parameter estimates $\hat{\mathbf{A}}_y$ and $\hat{\mathbf{b}}_y$ are sent to the edge computer via Ethernet (UDP) after the identification pulse has ended. On the edge device, they are then used to calculate $\mathbf{G}_{\text{feas}}$ and $\mathbf{e}_{\text{feas}}$, which define the feasible set (cf. Sec. 4.2). After the first excitation pulse sequence, the safeguard is activated using the identified feasible set. It is updated

with the newly acquired measurement data after each excitation pulse. In addition to computing the constraint matrices, the identified system matrices $\hat{\mathbf{A}}_y$ and $\hat{\mathbf{b}}_y$ are utilized to compensate for the digital control delay (cf. Sec. 3.2).

To evaluate the data-driven safeguard without a priori parameter knowledge, Fig. 4.4 depicts the feasible set calculated with the identified system matrices and constraints according to Tab. 3.1 (besides the constraints regarding i_L as described above), in green. Measurement points recorded in an experiment to validate the safeguard are shown in blue. The modulation index is drawn randomly to cover the entire permissible operation range of the depicted phase. As can be seen, the resulting measurements overlap with the green shaded feasible set. Only slight deviations in the boundary areas can be seen, which are within the expected measurement uncertainty. This demonstrates the functionality of the proposed safeguard without the use of prior knowledge and motivates utilizing this method in the following experiments to safely train the data-driven RL-based controller.

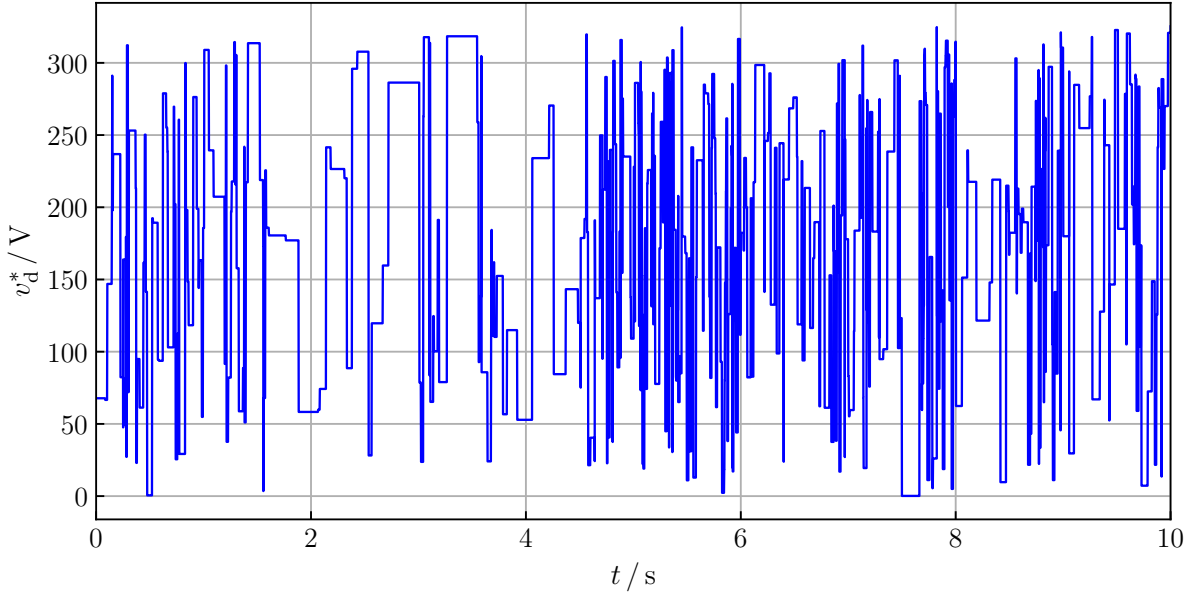


Fig. 4.5: Test case, showing the reference voltage v_d^* , which is used to evaluate the different HPs during the HPO after training ($v_{q,0}^* = 0$ V over the entire course).

4.4 Hyperparameter Configuration

Before starting the learning process, the RL algorithm must be configured. As described in Sec. 3.6, the size of the artificial neural networks (ANNs) (number of layers, number of neurons, activation function,...) and the learning algorithm parameters (learning rate, discount factor,...) must be selected. Since this is a task-dependent, nonlinear optimization problem, the hyperparameter optimization (HPO) introduced in Sec. 3.6 is used. In the following, the deep deterministic policy gradient (DDPG) algorithm (cf. Sec. A.1)

is utilized to learn the voltage control task according to Sec. 2.1 to supply a constant, resistive load within the experiment shown in Fig. 4.1. The edge learning toolchain presented in Sec. 3.5 is utilized for this purpose. As introduced in Sec. 3.6, this experiment is executed during the HPO as real-time simulations on the RCP units.

In the application investigated here, the training takes place in the rotating dq0 reference frame (cf. Sec. 3.4), while the corresponding transformations in Fig. 4.1 have been omitted for clarity. To provide sufficient exploration of the state space, the reference $\mathbf{v}^*$ is changed on the basis of a multivariate random walk during training. The load resistance remains constant, so that different voltage setpoints result in different load currents. The probability of a reference change is scaled over the course of the training. The magnitude of $v_{q,0}^*$ is scaled down while the magnitude of v_d^* is scaled up accordingly. This is motivated by the fact that, in the later application, the goal is to track a certain reference value in $v_d^* = \text{const.}$ while, normally, the other two reference values are $v_{q,0}^* = 0 \text{ V}$. This should provide sufficient exploration of the state space during the training. After each training run, the agent trained with the drawn HPs is evaluated using the deterministic test case shown in Fig. 4.5, which is excluded from any training run. The average, undiscounted return g_{test} achieved by the trained agent on the test case serves as the performance metric for the HPO. As described in Sec. 3.6, a trained RL agent can achieve a maximum of $g_{\text{test}} = 1$, which is not always possible due to physical reasons, e.g., during reference steps.

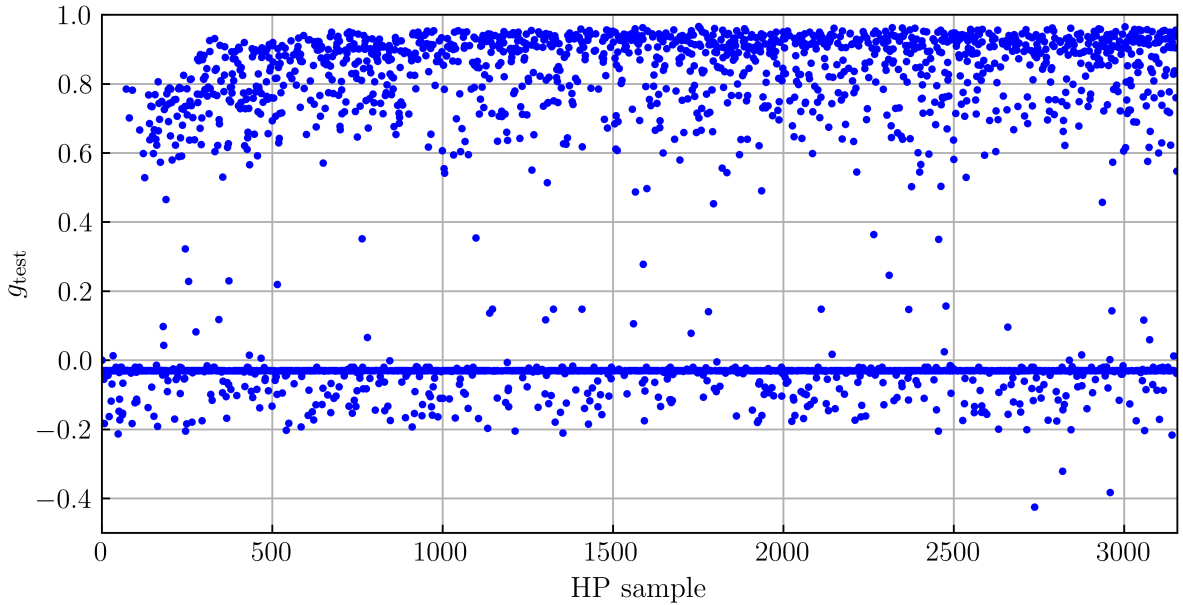


Fig. 4.6: HPO learning curve showing the undiscounted, average return of a deterministic test case performed by a trained agent for each HP sample.

Within the presented HPO, more than 3 300 samples of different HP sets were evaluated. The corresponding learning curve of the HP optimizer, represented by the achieved g_{test}

on the test case in Fig. 4.5, is depicted in Fig 4.6.¹ Here, one HP sample represents a trained RL-based controller with one set of HP configuration chosen by the HPO. It can be seen that a high return is already achieved after about 1000 samples, and does not improve significantly with an increasing number of HP sets.

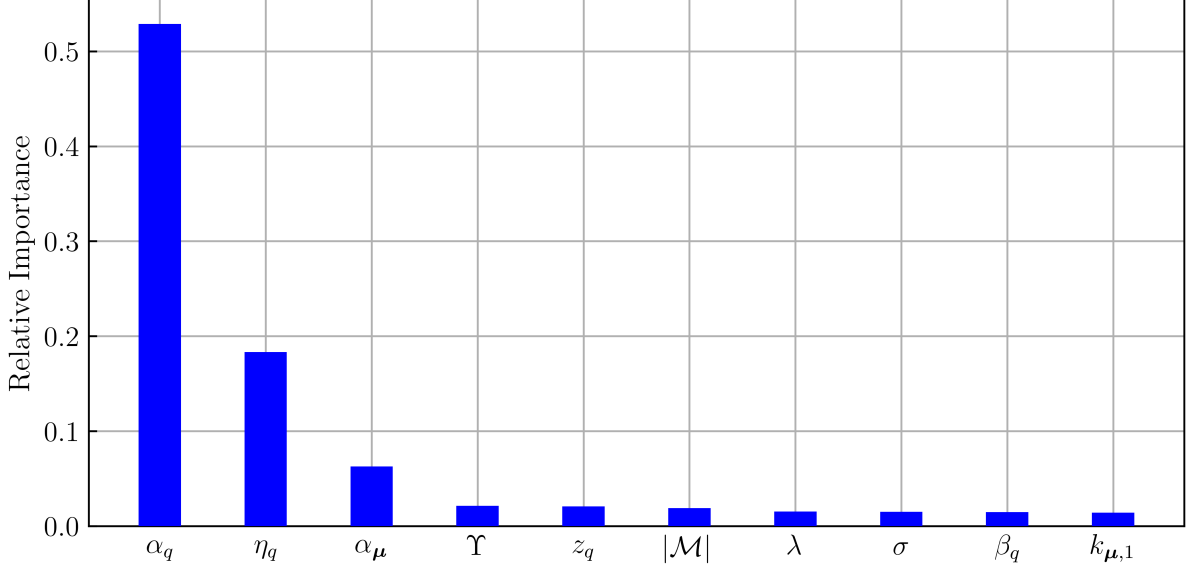


Fig. 4.7: Relative importance of the ten most important HPs within the HPO.

The ten most important HPs during the HPO are depicted with their corresponding relative importance in Fig. 4.7. Here, the importance is a non-negative number, where a higher value indicates that the parameter is more important to the HPO than a parameter with a lower value [102]. The importance, calculated according to [105], of all parameters together adds up to 100 %, with the ten most important parameters shown already accounting for 90 %. The ten most important HPs depicted in Fig. 4.7 are:

1. Step size (learning rate) of the optimizer used during training of the ANN representing the critic α_q (cf. Sec. A.1),
2. number of neurons per hidden ANN layer η_q of the critic,
3. learning rate of the optimizer used during training of the ANN which is utilized within the control loop to calculate the duty cycle – the actor α_μ ,
4. dropout Υ , a regularization parameter to stabilize the training,

¹The return shown here was calculated using the function defined in (3.1) without the $\max(\cdot)$ operator. This was added after the HPO was finished and motivated by testing with different weighting factors for the penalty term. The following experiments were nevertheless carried out with the parameter set found here, as it delivers sufficiently good results. Furthermore, it can be expected that the return achieved with (3.1) will yield a similar result to that achieved with the HP set used in Tab. A.1, since the $\max$ operator only affects negative rewards, which mainly comes into play in the case of poor performance.

5. number of layers of the critic ANN z_q ,
6. replay buffer size $|\mathcal{M}|$, in which the experiences are stored on the edge computer,
7. action noise parameter stiffness λ (cf. (A.4)),
8. action noise parameter diffusion σ (cf. (A.4)),
9. parameter of the LeakyReLU activation function of the critic β_q (cf. Sec. A.1).
10. time step at which the reduction of the learning rate of the actor ends $k_{\mu,1}$

The most important HPs are the step sizes of the optimizers for training the ANNs – the learning rates – and the architectures of the ANNs (here, number of layers and number of neurons per layer). However, the discount factor γ does not appear among the most important parameters, which could be explained by the dynamics of the control plant. Furthermore, the dropout factor Υ is among the ten most important parameters. This represents a regularization parameter to stabilize the training (cf. [106]). With regard to the listed parameters, it should be noted that 75 % of the relative importance is attributable to the first three parameters – the learning rates of the ANNs and number of neurons per hidden layer of an ANN.

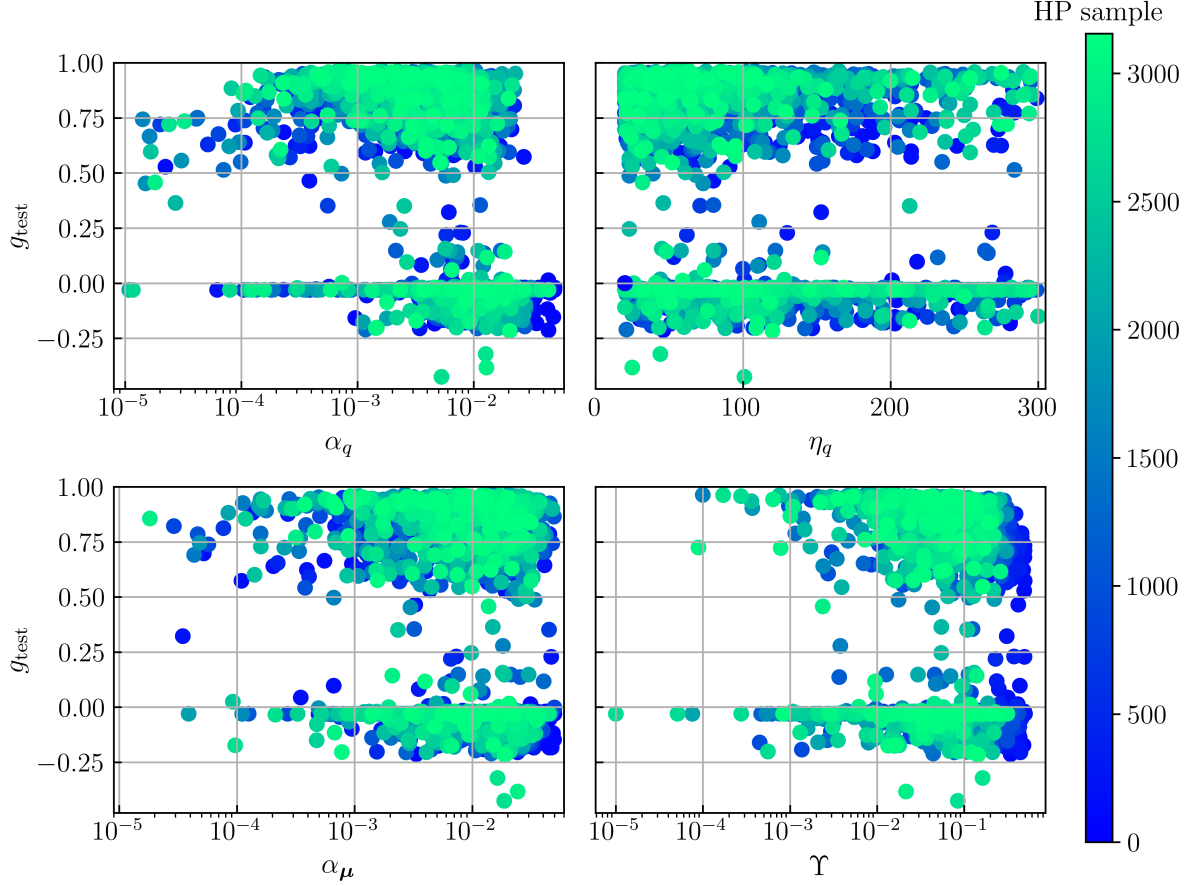


Fig. 4.8: Performance g_{test} achieved with the different values of the corresponding HP during the HPO. Here, blue samples indicate early samples within the HPO, while green samples indicate more recent ones.

Fig. 4.8 presents the four most important HPs with their respective drawn value of the HP and the performance g_{test} achieved. The performance does not depend solely on the parameter considered in Fig. 4.8, but depends on the whole parameter set. The respective samples are highlighted in color, with dark blue representing the oldest HP samples and light green representing the newest. This allows the development of the HPO over time to be read in color. For the depicted parameters, a clear trend can be seen as to how the parameter should be selected in order to achieve good performance. Further samples within the scope of the HPO should improve this trend even further, but the HP set determined by the present HPO can achieve sufficient controller performance in comparison to benchmark controllers, as shown below. The HP set listed in Tab. A.1 was used, which delivers sufficiently good results compared to other benchmark controllers, as shown in the following.

4.5 Training Behavior

Since the HPs were determined directly in the RCP framework, they are used without further adaptation to train the RL agent by means of a hardware experiment. To investigate the learning behavior, five randomly initialized agents are trained for 20 000 steps using the edge-learning framework from Sec. 3.5. Before each training run, the system identification (ID) process according to Sec. 4.3 is executed to parameterize the safeguard. Each training run takes roughly seven minutes and the corresponding learning curves are depicted in Fig. 4.9. The fact that the dispersion of the blue curve decreases towards

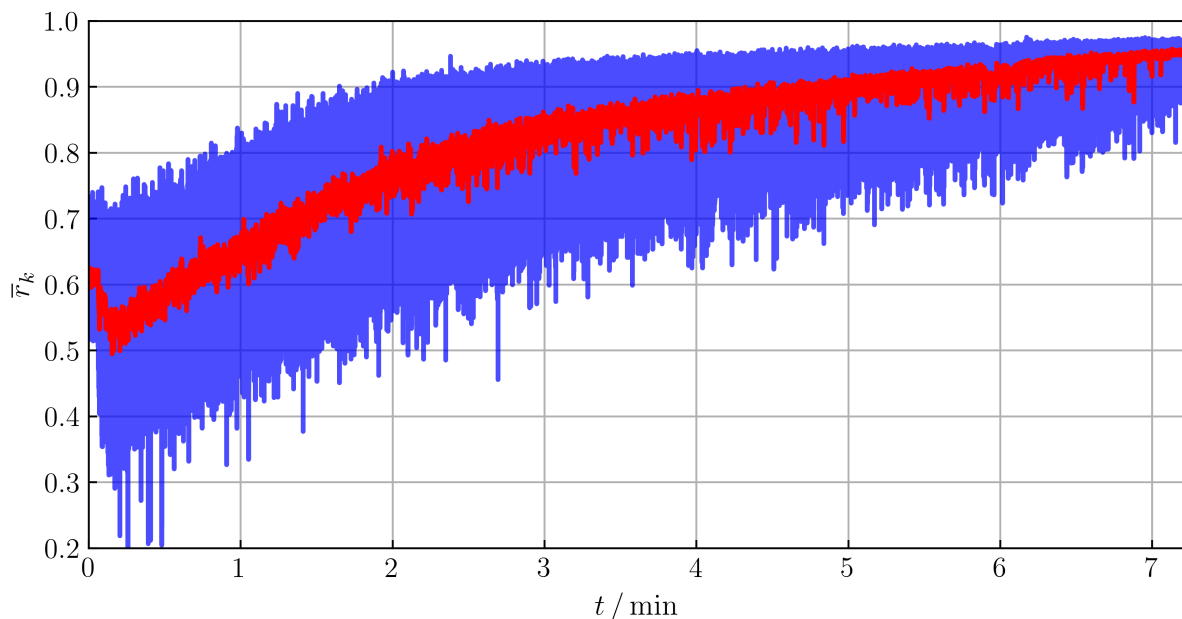


Fig. 4.9: Undiscounted, averaged mean reward per learning step of the five trained agents per approach in blue. A moving average with window size 1000 is shown in red.

the end of the training indicates a good convergence and, therefore, a robust training toolchain with low variance. It can be observed that, after seven minutes of training, the RL controller reaches a high reward close to the theoretical unity maximum (which is practically not achievable due to random reference changes during training). Longer training could lead to even better tracking performance, but the following performance analysis proves that the discussed training setup is sufficient to achieve results comparable to those of MPC. In the following, the best of the five trained agents is used.

4.6 Steady-State Behavior

In order to investigate the trained RL-based controller with regard to the steady-state behavior, the voltage tracking is evaluated for 50 equidistant voltage references in the

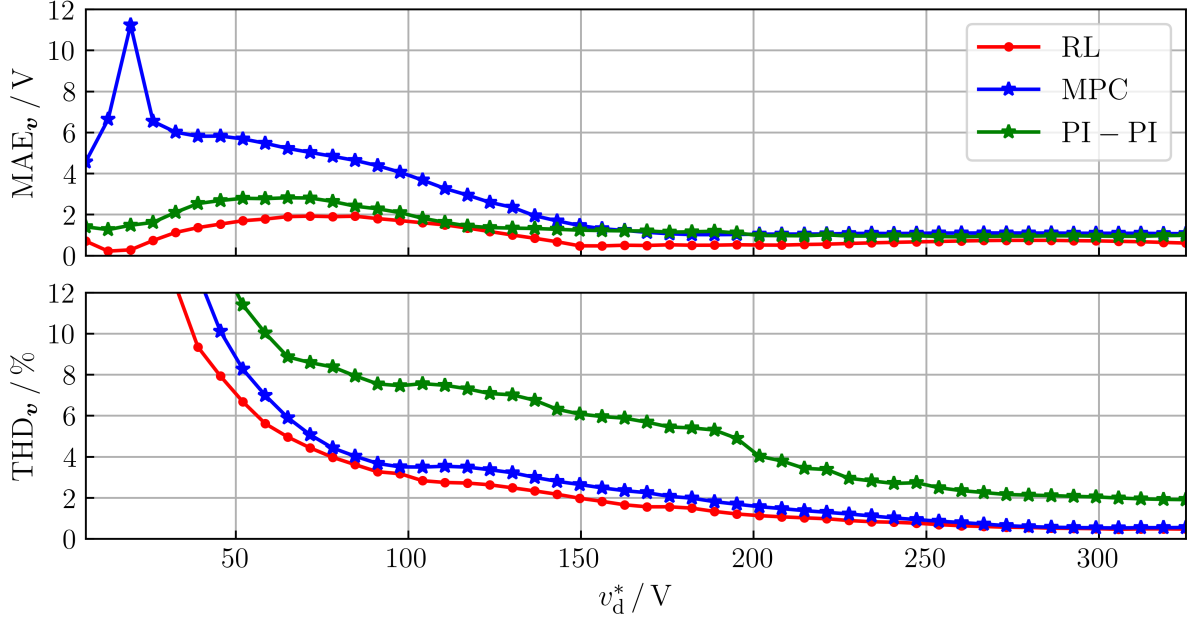


Fig. 4.10: Comparison of RL's, MPC's and PI-PI steady-state behavior for different setpoints.

range $v_d^* \in [0 \text{ V}, 325 \text{ V}]$ (with $v_{q,0}^* = 0 \text{ V}$). The mean absolute error (MAE) of the measured output voltage $\mathbf{v}_{abc}$ and the respective THD, measured over a period of $T_{\text{meas}} = 1 \text{ s}$, can be seen in Fig. 4.10. The voltage is measured with a HIOKI PW6001 power analyzer, while the MAE and THD are calculated according to:

$$\begin{aligned} \text{MAE}_v &= \frac{1}{T_{\text{meas}} f_s} \sum_{k=1}^{T_{\text{meas}} f_s} \frac{1}{3} \sum_{p \in \{a,b,c\}} |v_{p,k}^* - v_{p,k}|, \\ \text{THD}_v &= \frac{1}{T_{\text{meas}} f_s} \sum_{k=1}^{T_{\text{meas}} f_s} \frac{1}{3} \sum_{p \in \{a,b,c\}} \frac{\sqrt{\sum_{h=2}^H v_{p,k,h}^2}}{v_{p,k,h=1}}. \end{aligned} \quad (4.8)$$

Here, h defines the harmonic order and H the maximum analysis order. For this analysis, $H = 100$ is chosen. In order to compare the RL-based controller to another optimal control method, the MPC developed in [Own1] is used. Additionally, the experiment is carried out with a PI controller with an outer voltage control loop and an inner current control loop, implemented in a rotating reference frame. The PI controllers were designed according to [90], to achieve a good trade-off between steady-state accuracy and transient performance. The focus was on the transient behaviour, resulting in the following parameterization:

$$\begin{aligned} K_{p,c} &= 0.499 \text{ mV/A} & \text{and} & \quad K_{i,c} = 1.139 \text{ V/(As)}, \\ K_{p,v} &= 0.894 \text{ A/V} & \text{and} & \quad K_{i,v} = 1497.206 \text{ A/(Vs)}. \end{aligned} \quad (4.9)$$

Tab. 4.1: Steady-state performance – MAE and THD – of different controllers for the steady-state from the experiment shown in Fig. 4.10.

v_d^* / V	MAE _v / V			THD _v / %		
	MPC	RL	PI-PI	MPC	RL	PI-PI
65	5.22	1.90	2.81	5.90	4.96	8.86
150	1.46	0.47	1.24	2.64	1.97	6.08
325	1.06	0.61	0.98	0.55	0.47	1.91

In Fig. 4.10, it can be seen that the RL controller achieves an MAE below 2 V in the whole operating range, which is slightly smaller than the MAE that is achieved by the PI-PI and smaller than the error of the MPC, especially for small reference values. The tracking errors of the MPC and the PI-PI controller can be explained by non-linearities caused, e.g., by effects of the interlock time and non-linearities of the sensors, which could not be fully compensated for (cf. Sec. 3.1). This effect is investigated in more detail in the next section. For the THD, a similar trend can be seen. While for small reference values, the RL controller achieves smaller THD values than the MPC, both optimal controllers achieve a THD < 0.6% at a reference value of 325 V. The PI-PI controller shows a higher THD over the entire voltage range. While the steady-state performance of the PI-PI controller could be enhanced by another parameterization (cf. (4.9)), its transient behavior would deteriorate drastically. Since the goal, as motivated in Sec. 1, was to have a controller that could respond as quickly as possible and therefore to achieve better comparability concerning the transient behavior, a higher steady-state error was accepted. For a quantitative comparison, Tab. 4.1 documents the performance (MAE and THD) of the different controllers in steady state for partial and full load range.

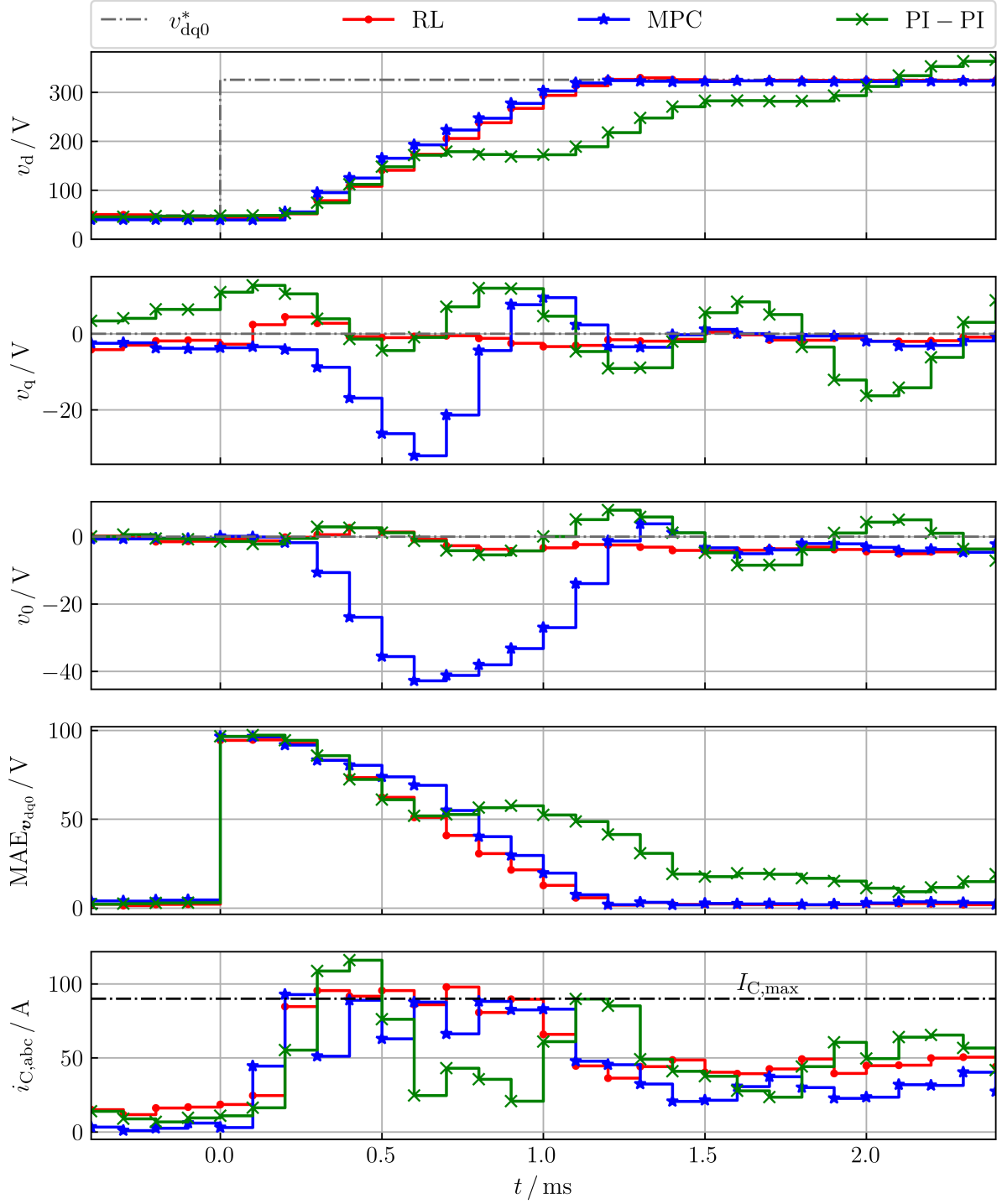


Fig. 4.11: Comparison of the step responses of the different control approaches.

4.7 Transient Behavior

As shown in the previous section, the RL controller shows lower MAE in the steady state than both MPC and PI-PI controller for low reference voltages. To investigate this in addition to the transient behavior, a step starting from $v_d^* = 50$ V to $v_d^* = 325$ V is investigated. The other setpoints are set to $v_{q,0}^* = 0$ V. The results are depicted in Fig. 4.11, with the reference change at $t = 0$ s. The measurements were recorded with dSPACE ControlDesk. It can be seen that the RL agent and the MPC reach the reference in $t_{\text{rise}} = 1.2$ ms without overshoot, while the PI-PI controller reaches the setpoint the first time after $t_{\text{rise}} = 2$ ms, followed by an overshoot, lasting 8 ms (not depicted in the figure). The rise times t_{rise} achieved by the different controllers are provided in Tab. 4.2, while t_{rise} is calculated according to [104] for the period of time after a reference step when the new reference value is reached. The classic PI controller requires a settling time $t_{\text{settle},5\%}$ more than seven times that of the optimal controllers to reach the voltage setpoint, which can be seen in Tab. 4.2. Here, the settling time $t_{\text{settle},5\%}$ is calculated according to [104] with a 5% tolerance band. The rise time of RL and MPC is greater than the settling time here because $t_{\text{settle},5\%}$ is defined in a 5% tolerance band and neither controller overshoots. The number of decimal places is limited by T_s . Additionally, Tab. 4.2 lists the error-time integral $\int \text{MAE}_v dt$ of the $\text{MAE}_{v_{dq0}}$ shown in Fig. 4.11 calculated over 10 ms after the step at $t = 0$ s.

Tab. 4.2: Transient performance of different controllers for the step response shown in Fig. 4.11. Listed are the $\int \text{MAE}_v dt$ / Vs measured over 10 ms after the step, the rise time t_{rise} and the settling time t_{settle} in a 5% tolerance band.

	MPC	RL	PI-PI
$\int \text{MAE}_v dt$ / Vs	0.092	0.083	0.193
t_{rise} / ms	1.2	1.2	2.0
$t_{\text{settle},5\%}$ / ms	1.1	1.1	8.4

The rise time can be reduced by lowering the constraints regarding the capacitor current defined in (2.33) (cf. [Own1]). However, the differential constraints are necessary to avoid capacitor damage in the early learning phase of the RL agent, which can be caused by erratic voltage changes due to exploration resulting in high currents which can cause thermal overloading (cf. Sec. 2.4.3). Although acceptable for a short time period, safe operation cannot be ensured during the training and must therefore be considered in the safeguard.² A solution to this problem could be to introduce an $\int i^2 dt$ constraint instead of the hard

²Since the MPC does not need to be trained through exploration, this problem does not occur, allowing to react more quickly during transients by reducing these constraints under the assumption that transients only occur seldomly as otherwise the elevated capacitor current would also become problematic in the MPC case.

voltage change limit [107]. This would require sufficient changes both in the constraint calculation for the MPC and in the implementation of the safeguard and would also affect the finite computation time examined in Sec. 4.10, which is why this investigation will be continued with the capacitor current limitations already introduced.

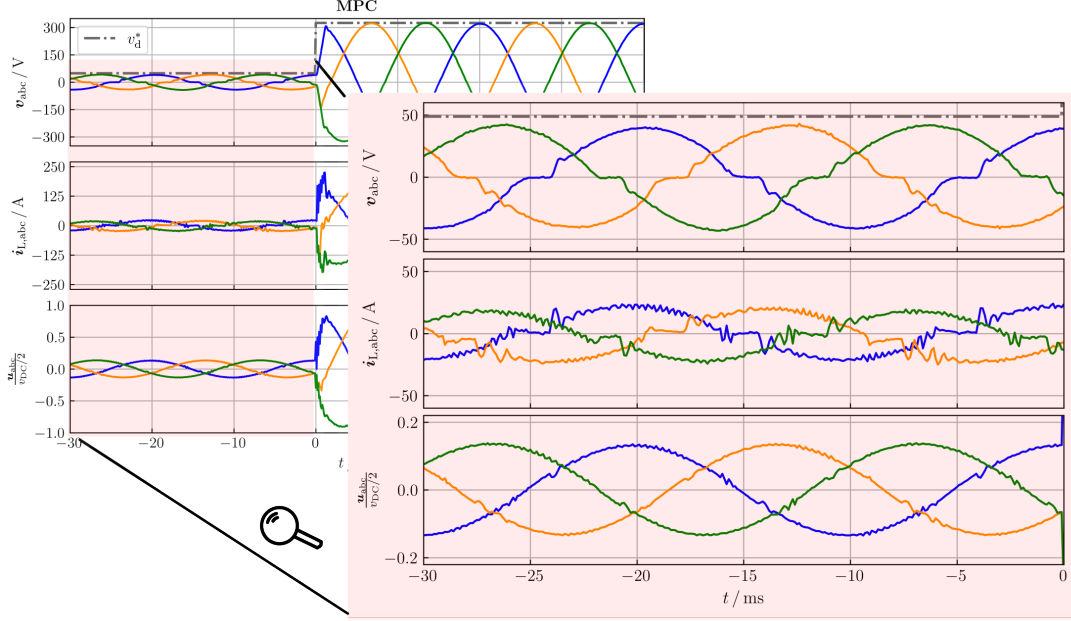


Fig. 4.12: Step response, from Fig. 4.11 in abc coordinates, performed by the MPC.

The reason for the significant deviation from the setpoint in case of MPC can be seen in Fig. 4.12 for $t < 0$ s. As illustrated in the zoom window, the current remains at zero for very low modulation indices. These non-linearities do not align with the (idealized) model assumptions and could be attributed to the interlock time combined with current sensor non-linearities which are not taken into account within the prediction model (cf. [Own1]). As can be seen in the zoom window in Fig. 4.13, the RL controller has learned to counteract that non-linear effect by alternately selecting a larger and a smaller action such that the desired voltage is reached on average. Therefore, as already seen in Sec. 4.6, when comparing the optimal controllers' performance at the lower voltage reference ($t < 0$ s) in Fig. 4.11, the RL controller achieves a lower MAE in the voltage reference tracking task than the MPC controller.

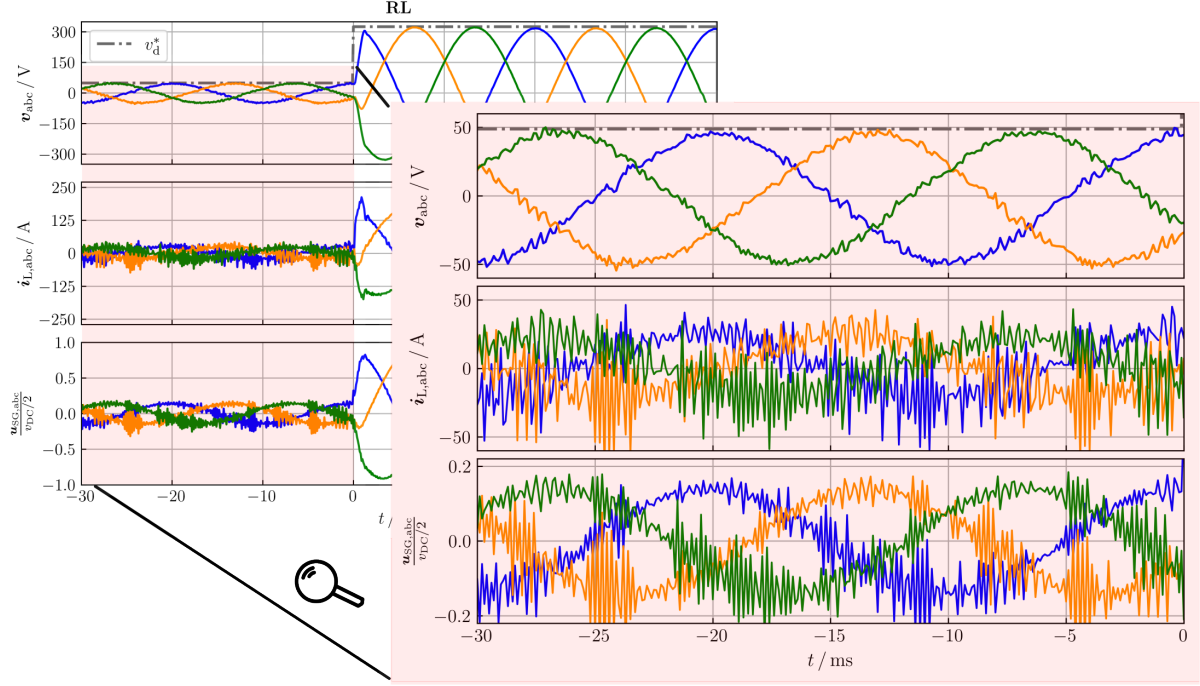


Fig. 4.13: Step response, as shown in Fig. 4.11 in abc coordinates, performed by the RL controller.

4.8 Black-Start Capability

Fig. 4.14a) shows the ability of the RL-based controller to black-start a (micro-)grid under load and perform an emergency shutdown. For comparability, the same experiment utilizing MPC is depicted in Fig. 4.14b). Similar to the step response in Sec. 4.7, both controllers perform the black-start at $t = 0$ s in about 1.2 ms without overshoot and in compliance with the system constraints. The same can be observed for the emergency shutdown at $t = 40$ ms. This demonstrates the ability of the developed RL-based controller to cover the entire operating range of the grid-forming inverter with fast transients and highly accurate stationary tracking performance, similar to the MPC. Thereby, the RL-based controller does not hold the inductor current constantly at $i_L = 0$ A when the voltage reference is set to $v_d^* = 0$ V. This could be explained by the design of the reward function, which is only dependent on the voltage deviation and does not incorporate any preferences for the inductor current. This could be improved by considering the current ripple within the reward function. Another approach would be to extend the actor by an according integral part as it has been studied in a simulative investigation in [Own5]. However, since the performance of the RL-based controller is already comparable to or better than that of the reference optimal controller (MPC) in both transient and steady state, these variants were not implemented. The effect could also be explained by how the RL agent is compensating the inverter non-linearities.

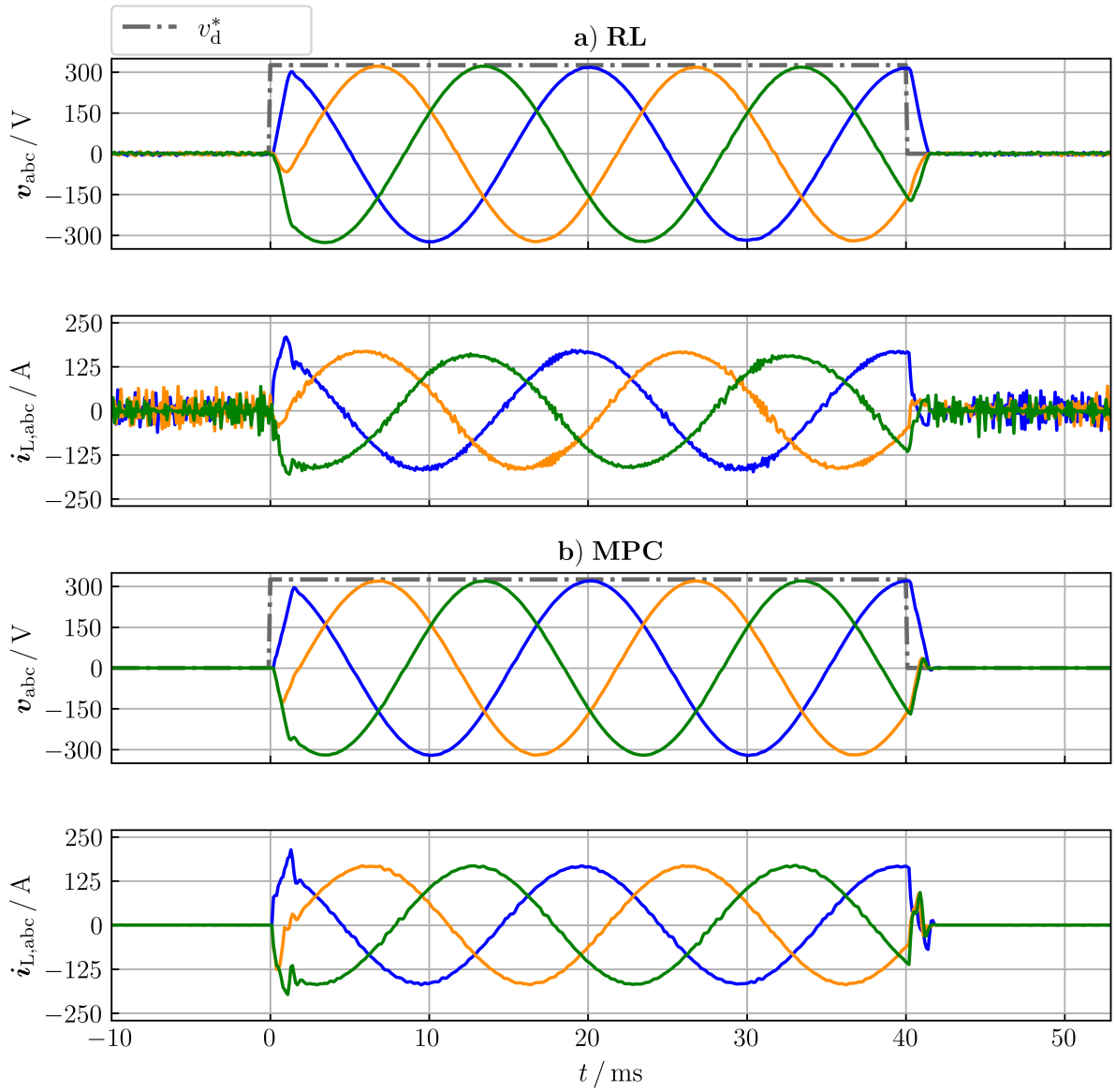


Fig. 4.14: Proof of the black-start and emergency stop capability with corresponding comparison to MPC.

4.9 Operation with Active Output Constraints

In order to show that the safeguard fulfills the requirements regarding the constraints, an experiment is shown in Fig. 4.15a), where the RL-based controller is operated at the current limit. For comparison, the same experiment is executed with the MPC in Fig. 4.15b). For this purpose, the maximum current used within the calculation for the feasible set and the limits of the MPC is temporarily set (from $I_{L,\max} = 250$ A in the previous experiments) down to $I_{L,\max} = 100$ A for this experiment. This results in the current amplitude i_L to exceed its boundary in the voltage control task defined in Sec. 4.1. Nevertheless, the RL agent was trained for a safeguard limit of $I_{L,\max} = 250$ A. In order to keep the quantities i_L and v inside the defined constraints, MPC and the safeguard follow strongly different approaches. The MPC is capable of predicting the impact of an active constraint and accounts for this case foresightedly. It is able to prevent a constraint violation by setting an appropriate plant input $\mathbf{u}$. In contrast, in

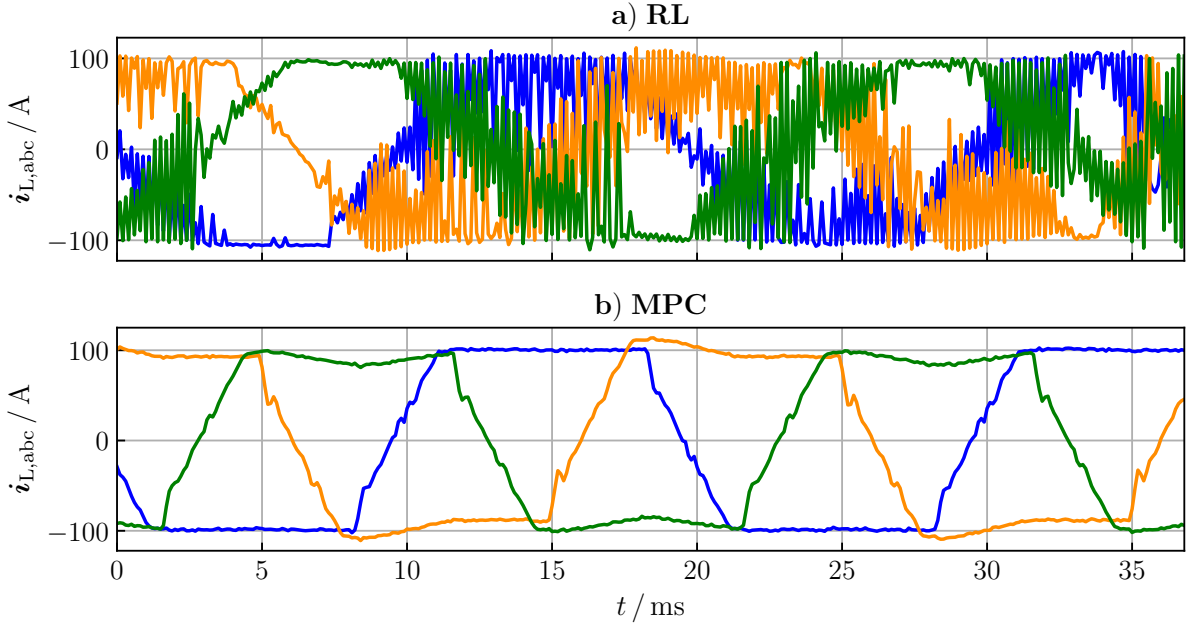


Fig. 4.15: Comparison of the current waveforms at active constraints with $R = 2\ \Omega$ and the maximum current set to $I_{L,\max} = 100$ A for RL and MPC for $v_d^* = 325$ V.

the presented RL framework, the safeguard investigates the control command for each phase individually and shifts the voltage $\mathbf{u}$ if necessary, such that both phase current and voltage adhere to the constraints. Here, the RL agent only receives indirect feedback via the reward about the post-processing of the input value $\mathbf{u}$, which results in a cutoff in case of an output value exceeding the limit, visible in Fig. 4.15 a). The RL controller targets to compensate the manipulation caused by the safeguard, which results in some oscillations. This effect is further amplified because the utilized policy was trained for a safeguard limit of $I_{L,\max} = 250$ A. Nevertheless, the safeguard allows compliance with

the newly configured current limit, independently of the RL controller's training. This concludes the investigation and confirms the functionality of the safeguard already shown in Fig. 4.4.

4.10 Computing Time Evaluation

Since the previous chapters have already demonstrated the advantages of the RL-based controller compared to the PI-based controller and MPC, the question of computational effort remains. This chapter examines the required computing time in form of the turnaround time T_{TA} of the different control approaches. For RL, two time measurements are provided. One after the training is completed and the training process is stopped and a second, RL_{train} , where additional time-consuming processes are active, required during training, such as receiving and processing the weights of the ANNs. The resulting computing times are listed in Tab. 4.3. The Intel Core i7-6820EQ @ 2.8 GHz real-time processor of the DS6001 card of the dSPACE SCALEXIO is utilized for control (cf. Tab. 3.1). The test case shown in Fig. 4.5 was used to determine the calculation time. In addition, the MAE of the different controllers performing the test case is documented (cf. (4.8), with $T_{meas} = 10$ s according to Fig. 4.5). Fig. 4.16 shows the range of the computing time required in comparison to the available task turnaround time T_s marked in red. The median, minimum and maximum are depicted, with the first and third quartiles omitted for reasons of clarity, as the spread from the median is very small. As can be seen, the computing efforts of RL and MPC are comparable, with MPC having a smaller median task turnaround time. In the case of RL, this depends heavily on the size of the actor's ANNs implemented on the CPU, and in the case of MPC, it depends on the selected prediction horizon (here, $k_{f,MPC} = 2$, cf. [Own1]). The PI-based controller requires significantly less computing time. It should be noted that approximately 6 μ s is already required for the I/O operations to communicate, e.g., the measurement values. The MAE is higher for the PI-based controller than for the optimal control approaches (cf. Tab. 4.3). This depends on the form of the test case, which consists of many reference steps. As already shown in Sec. 4.7, the PI-based controller performs worse during transients than the RL-based controller and the MPC.

Tab. 4.3: Performance and required calculation time of different controllers on the test case shown in Fig. 4.5.

	MAE_v / V	$T_{TA,med} / \mu s$	$T_{TA,max} / \mu s$
MPC	3.02	17.48	39.22
RL	2.34	24.15	27.46
RL_{train}	2.34	30.80	34.25
PI-PI	15.74	9.18	11.04

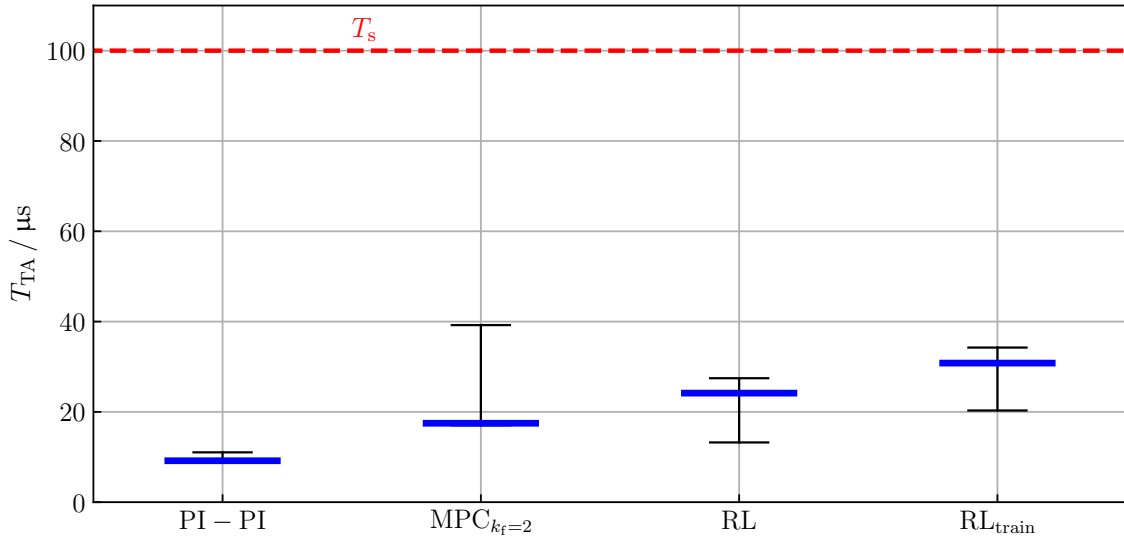


Fig. 4.16: Ranges of the turnaround time T_{TA} for the different control algorithms determined using the test scenario depicted in Fig. 4.5. The median turnaround time $T_{TA,med}$ is highlighted in blue.

4.11 Unbalanced Loads

The integration of renewable energy sources (RES) as well as single-phase connected loads can create unbalances in the electrical energy grid and have a negative impact on grid quality and stability in terms of frequency and voltage [108]. Classic PI controllers usually operate in a rotating dq0 reference frame. With an imbalance, oscillations occur in that reference frame which significantly impair the control performance of the PI controller and have a negative impact on the grid quality. To control these oscillations, symmetrical sequence decomposition (SSD) can be utilized, whereby an unbalanced system is represented as a linear combination of several balanced components [109]. As described in [110], this technique can be used to overcome the challenges of PI controllers in combination with unbalanced loads. However, it introduces a further delay into the control loop, which may lead to poor control performance, especially during transients [110]. Proportional resonant (PR) controllers can instead be used directly in the fixed reference frame and do not require SSD or other transformations. This overcomes the disadvantage described above [3, 108]. However, PR controllers are challenged when dealing with harmonics, often requiring a control design for each harmonic [3]. There are many approaches to combine PI and PR controllers with each other in order to combine their advantages, but this requires a complex, iterative control design [111, 112]. Since the RL-based controller under investigation has demonstrated good performance with the symmetrical load already examined, it will be investigated whether it is possible to learn to supply an unbalanced load without extensions of the control structure or feature engineering. The RL-based controller was implemented so far in the rotating reference frame as described

above and in addition it will be investigated in the following, whether it can also perform in the fixed abc reference frame. For this purpose, the experiment is extended by a second three-phase resistor in parallel to the one used above as shown in Fig. 4.17. The results presented below were originally published in [Own2].

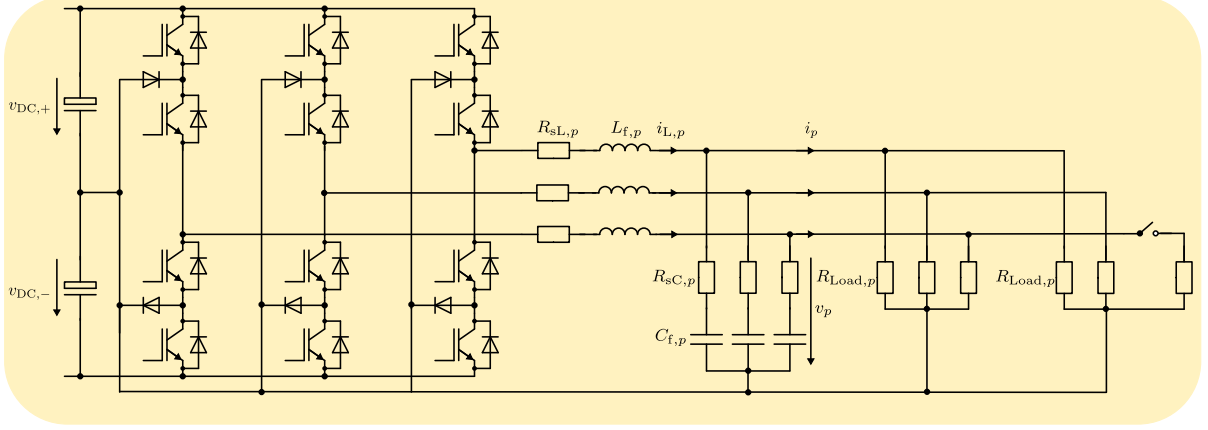


Fig. 4.17: Extension to an unbalanced, resistive load of the plant highlighted in yellow shown in Fig. 4.1 by connecting a second resistor in parallel. The switch changes the experiment between balanced and unbalanced load.

Four different experiments are investigated in the following:

1. dq0_B : Training an RL agent in the rotating dq0 reference frame to supply a balanced load ($R_{\text{Load,abc}} = 2\Omega$)
2. abc_B : Training an RL agent in the fixed abc reference frame to supply a balanced load ($R_{\text{Load,abc}} = 2\Omega$)
3. dq0_{UB} : Training an RL agent in the rotating dq0 reference frame to supply an unbalanced load ($R_{\text{Load,a}} = 4\Omega$, $R_{\text{Load,bc}} = 2\Omega$)
4. abc_{UB} : Training an RL agent in the fixed abc reference frame to supply an unbalanced load ($R_{\text{Load,a}} = 4\Omega$, $R_{\text{Load,bc}} = 2\Omega$).

The four cases listed here describe the different training scenarios. After the training, the differently trained agents are tested in a balanced and unbalanced load case, both in abc and dq0. Besides the load resistor, the experiment is parameterized similarly to the one in Sec. 4.1. Especially, the maximum current is again configured to $I_{L,\text{max}} = 250\text{ A}$.

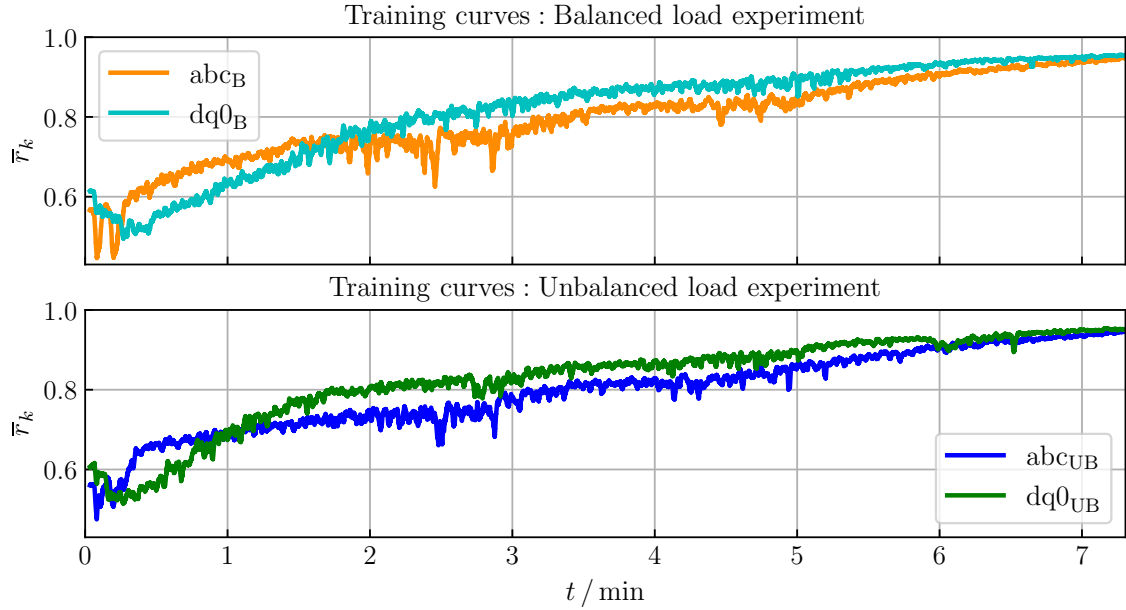


Fig. 4.18: Undiscounted, average reward of 3 trained agents per approach with a moving average with window size of 1 s.

4.11.1 Learning Behavior

Three agents per setting have been trained using the framework introduced in Sec. 3.5. Thereby, the same HPs, determined for the balanced load case in Sec. 4.5, are utilized (cf. Tab. A.1) and no structural changes have been made to the RL-based controller. Fig. 4.18 shows the resulting learning curves. Depicted is the undiscounted, average reward of the three trained agents per approach with a moving average with window size of 1 s for better comparability. The learning curve dq0_B represents the same experiment as in Sec. 4.5 and is comparable to the red learning curve in Fig. 4.9. This highlights the reproducibility of the experiment. The upper subplot shows the training curves of the balanced load experiment while in the lower subplot, the training curves of the unbalanced load experiment are depicted. Indicated by the increasing reward, it can be concluded that the RL agent can be trained to supply a balanced and an unbalanced load, since the upper and lower subplots are similar in the respective learning curves and final reward values. Showing different behavior between the abc and dq0 learning curves during the training can be explained on the one hand by the increased excitation by the AC signals and on the other hand by the fact that it is easier to track a DC signal. Toward the end of training, all settings perform similarly well, with the trainings in dq0 having a marginal lead. Accordingly, it is possible to control both a balanced and an unbalanced load by an RL agent in both a fixed abc and a rotating dq0 reference frame with similar performance. In order to compare the control performance between the different approaches in more detail and to evaluate whether it is possible to control an unbalanced load with an agent

trained in a balanced load setting (and vice versa), the transient and steady-state behavior are further investigated below.

4.11.2 Transient Behavior

To investigate the transient behavior of the different approaches, the best agent per setting is used to perform a reference step in v_d^* from 50 V to 325 V while $v_{q,0}^* = 0$ V, as already investigated in Sec. 4.7. Fig. 4.19 shows the resulting performance of the four different agents supplying a balanced load. Although two agents operate in the fixed abc frame, all measurements are shown here in the rotating dq0 reference frame for better comparability. All four agents are able to complete the reference step in about 1 ms, with the agents

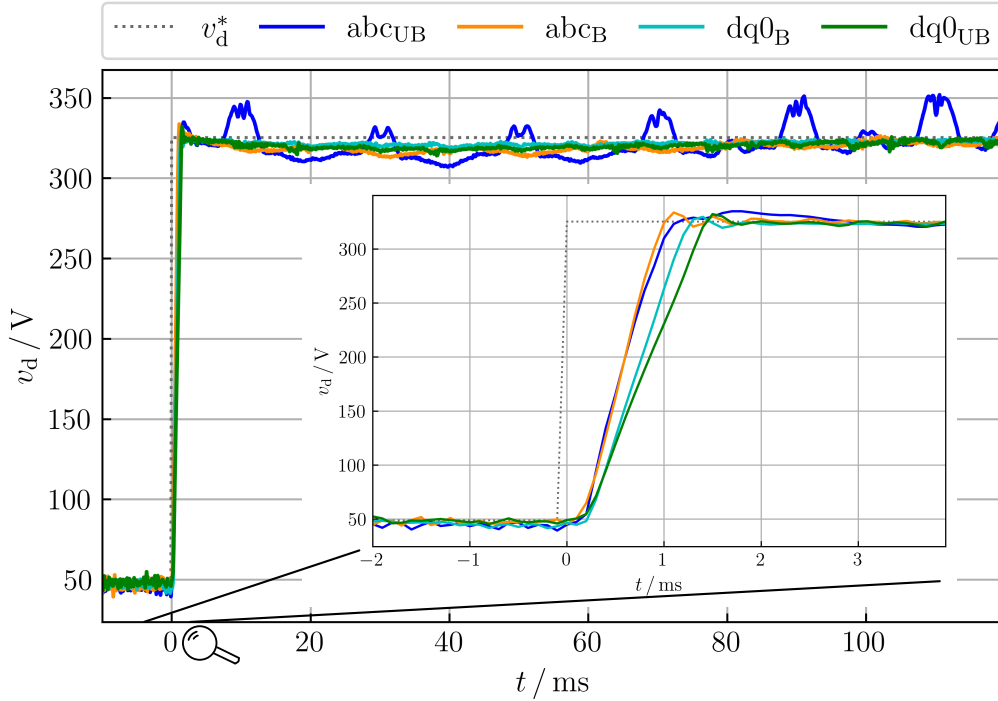


Fig. 4.19: Best RL agent per setting performing a reference step in v_d in the balanced load setup (cf. Fig. 4.17) plotted in the dq0 reference frame.

trained in abc being faster than those trained in dq0. This can again be explained by the nature of the explorative behavior of the reference signals provided in the non-rotating abc frame, as compared to those in the dq0 stationary frame. On the other hand, the agents trained in dq0 are slightly better in steady-state performance, as will be shown in the next section. In addition, it can be observed that the agent trained in dq0 in an unbalanced load setting ($dq0_{UB}$) is also able to supply a balanced load with the demanded voltage and performs similarly well to the agent trained in dq0 on the balanced load, while the agent abc_{UB} performs worst compared to all other agents. This can be explained because the abc_{UB} agent has to compensate for a learned offset in one phase caused by the unbalance

during its training, while for the $dq0_{UB}$ agent this effect is presented as disturbances. Fig. 4.20 shows the same experiment for the unbalanced load setting. Both controllers

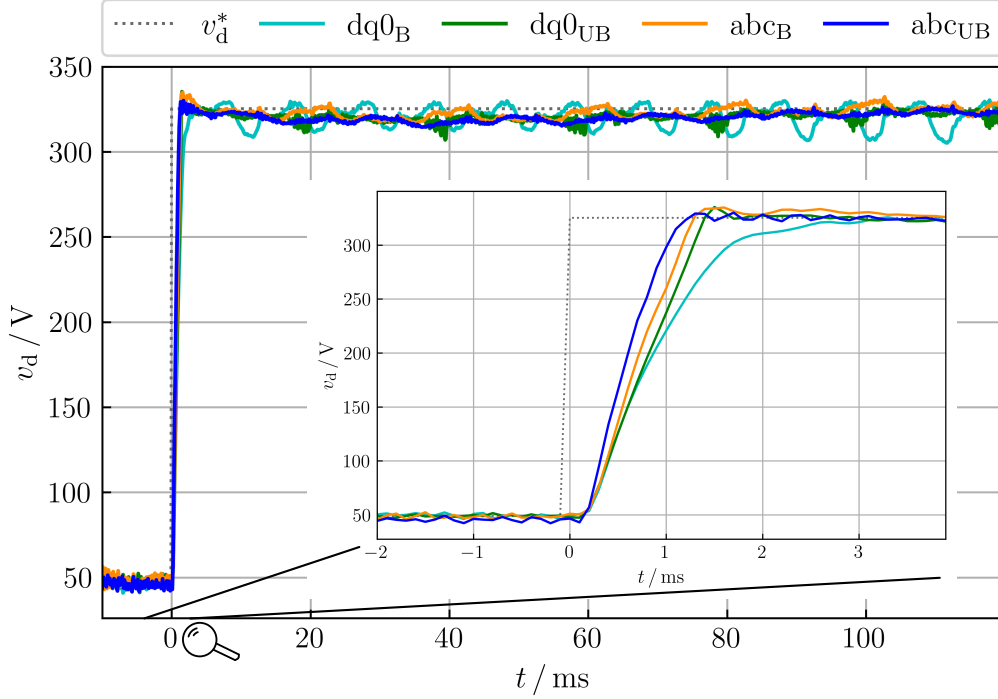


Fig. 4.20: Best RL agent per setting performing a reference step in v_d in the unbalanced load setup (cf. Fig. 4.17) plotted in the $dq0$ reference frame.

trained in the unbalanced experiment setting perform well, while abc_{UB} performing best. With the $dq0_{UB}$ agent, slight, recurring oscillations can be recognized, which are due to the oscillations in the rotating reference frame caused by unbalance, like described above. This can be observed even more strongly with $dq0_B$, which was not specifically trained for an unbalanced load. This leads to the conclusion that it is possible to supply an unbalanced load in the $dq0$ reference frame using an RL agent. Fig. 4.21 shows the measurements of the same reference step in abc for the unbalanced load, while the abc_{UB} agent is used in the shown experiment. The unbalance in phase a (blue) is clearly visible in the currents $i_{L,abc}$, while it is not visible in the voltage curves as demanded. The imbalance can also be seen in the zoomed subplot of $v_{L,abc}$ in Fig. 4.21. $v_{L,abc} = u_{SG,abc} - v_{abc}$ is thereby calculated from the difference of the measured filter output and input voltage and low-pass filtered.

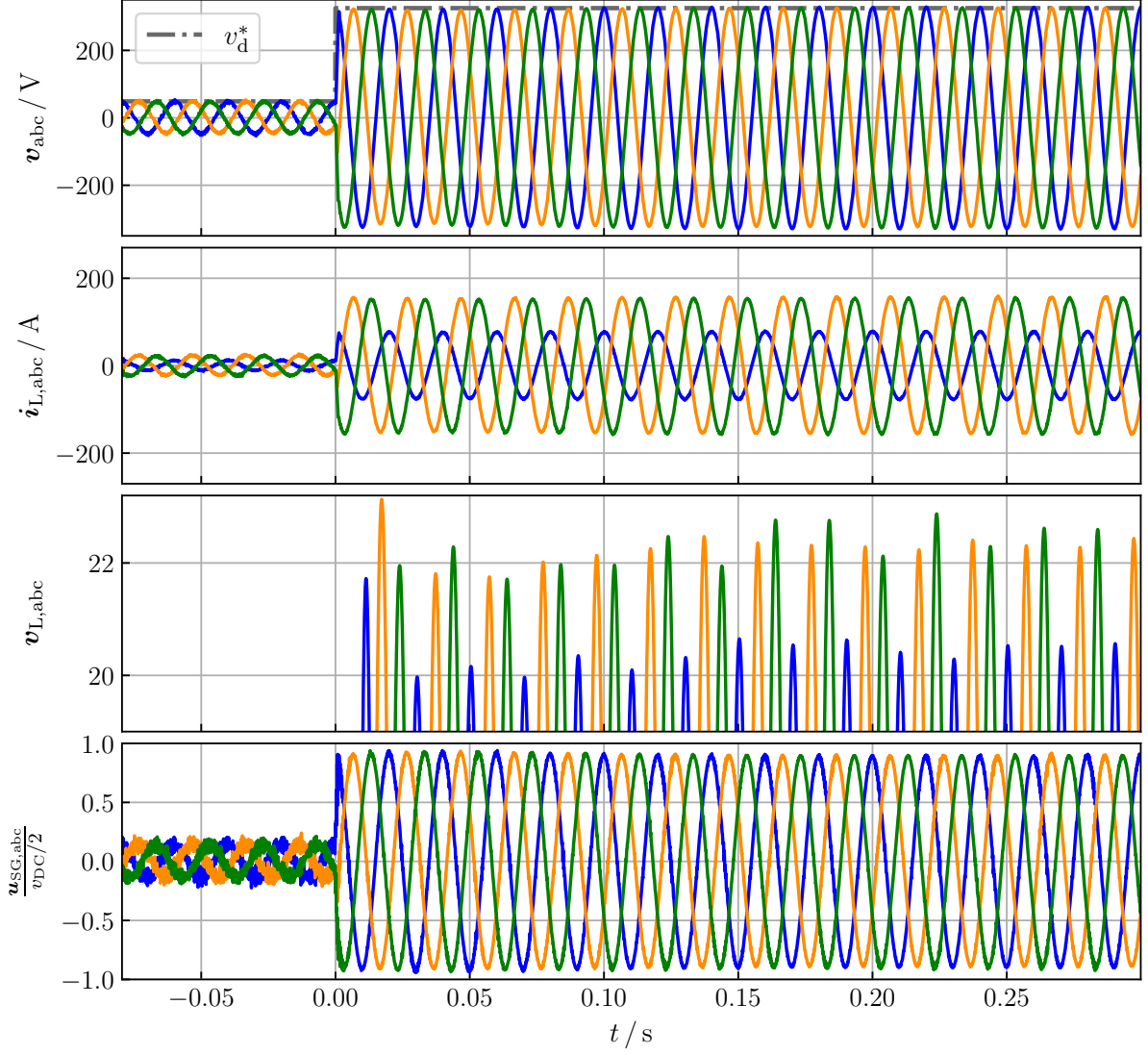


Fig. 4.21: Measured results of the best RL agent trained on an unbalanced load setting (abc_{UB}) performing the same reference step in the unbalanced load setting shown in Fig. 4.20 plotted in the abc reference frame.

4.11.3 Steady-State Behavior

To compare the four trained settings with regard to their steady-state behavior, similar to Sec. 4.6, different voltage setpoints v_d^* (while $v_{q,0}^* = 0$ V) are held in an experiment for two seconds. The MAE between the setpoint v_{dq0}^* and the measured voltage transformed into dq0 during steady state is shown in the upper subplot of Fig. 4.22 for the balanced load experiment. The lower subplot shows the THD of the output voltage v_{abc} measured with a HIOKI PW6001 power analyzer. It can be seen that all agents perform well besides abc_{UB} , as already seen in the transient behavior experiment for a balanced load. The THD results confirm the previous investigation results.

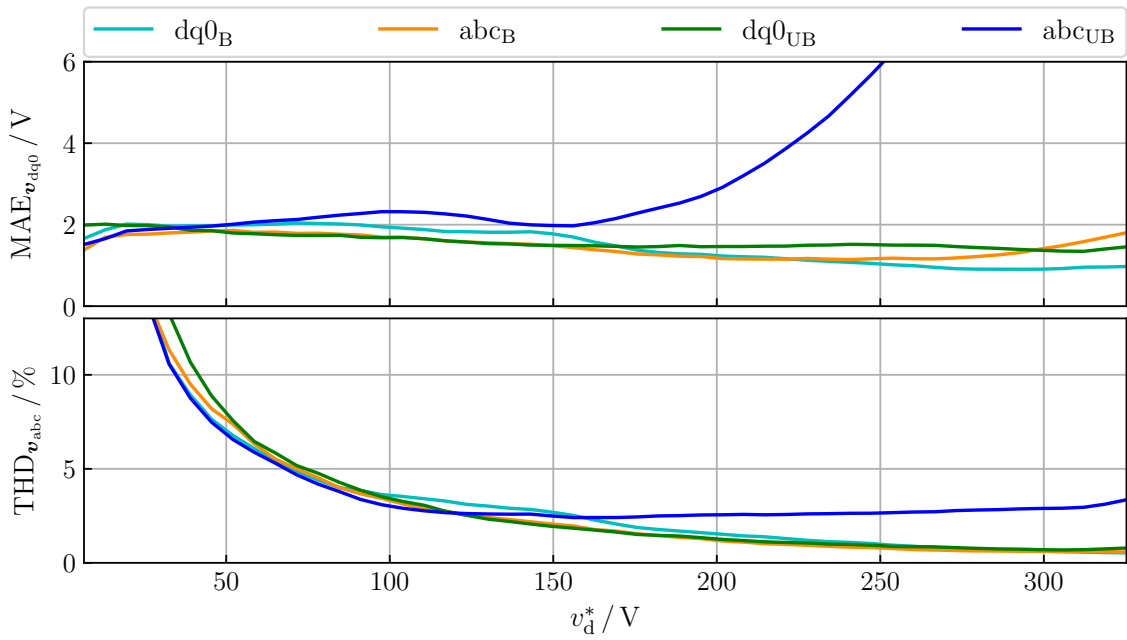


Fig. 4.22: Best RL agent per approach performing in steady state for different setpoints v_d^* in the balanced load setting.

Fig. 4.23 shows the corresponding results for the steady-state behavior in the unbalanced load experiment. It can be observed that the agents trained in the unbalanced setting achieve better results in the MAE with regard to steady-state behavior with no recognizable difference concerning whether the training took place in dq0 or abc. The agents trained in the balanced setting show an increasing MAE as the reference increases. In the THD measurement in the lower subplot, it can be observed that all agents perform similarly well except for the agent trained in dq0 in the balanced load setting for high reference values. This corresponds to the result presented in the transient behavior and is caused by the oscillations in the rotating reference frame caused by the unbalance.

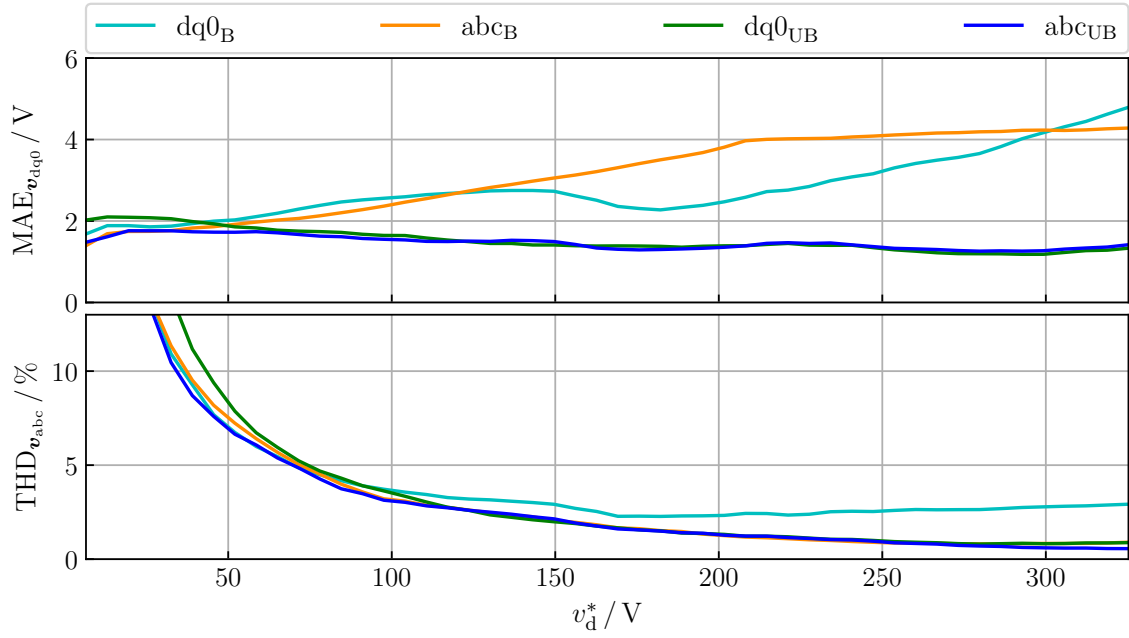


Fig. 4.23: Best RL agent per approach performing in steady state for different setpoints v_d^* in the unbalanced load setting.

knowledge concerning filter hardware and load, this section is an important step towards establishing an autonomous learning framework for heterogeneous inverter control applications with minimal human intervention and design effort. The results presented in the following were originally published in [Own3].

The RL-based voltage controller according to Fig. 3.1 is used as a source (cf. Fig. 5.1). The experimental investigation is performed on the two industrial-grade inverters connected back-to-back, as per Fig. 3.2, while the hardware specifications per inverter are listed in Tab. 3.1. Especially, the maximum current is now configured to $I_{L,\max} = 600$ A. During the following experiments, one inverter acts as a voltage-forming inverter to train the RL agent, while the other inverter performs as arbitrary and variable load. This load is implemented with an FCS current control MPC similar to [113]. The voltage-forming inverter forms a decentralized stand-alone grid depending on an oscillator within the control circuit, while the currents of the load inverter are selected depending on this reference so that the desired active and reactive power can be realized.

5.1 Safeguard Extension

As can be seen in (2.21) and (2.38) the calculation of the feasible set is dependent on the future disturbance. This requires a prediction of the disturbance d and a recalculation of the feasible set for a changing disturbance behavior. Hence, possibly available information regarding the future load behavior could be incorporated here (e.g., derived from a load model). If such information is not available, the disturbance can be considered in simplified form as an input with constant amplitude (cf. Sec. 2.2):

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{A}\mathbf{x}_k + \tilde{\mathbf{B}}\tilde{\mathbf{u}}_k, \\ \mathbf{y}_k &= \mathbf{C}\mathbf{x}_k + \tilde{\mathbf{E}}\tilde{\mathbf{u}}_k, \end{aligned} \quad (5.1)$$

with $\tilde{\mathbf{u}}_k = [u_k \ i_k]^\top \in \tilde{\mathcal{U}} \subseteq \mathbb{R}^{\tilde{m}}$, $\tilde{\mathbf{B}} = [\mathbf{b} \ \mathbf{h}]$ and $\tilde{\mathbf{E}} = [\mathbf{0} \ \mathbf{g}]$. According to (2.10), (5.1) can be reformulated to depict the dynamics of the measured output:

$$\mathbf{y}_{k+1} = \underbrace{\mathbf{C}\mathbf{A}\mathbf{C}^{-1}}_{\mathbf{A}_y} \mathbf{y}_k + \underbrace{\left(\mathbf{C}\tilde{\mathbf{B}} + \mathbf{C}\mathbf{A}\mathbf{C}^{-1}\tilde{\mathbf{E}} + \sin\left(\theta_{k+1} + p\frac{2\pi}{3}\right)\tilde{\mathbf{E}} \right)}_{\tilde{\mathbf{B}}_y} \tilde{\mathbf{u}}_k. \quad (5.2)$$

Here, $p \in \{0, 1, 2\}$ enumerates the specific phase. If further expert knowledge about the future load profile is available, this can be incorporated into the calculations via l_k . Using (5.2) starting from (2.18) with the extension described in the Sec. 2.4.3 and extending the input constraints according to:

$$\tilde{\mathbf{W}}_u = \begin{bmatrix} \mathbf{I}_{\tilde{m} \times \tilde{m}} \\ -\mathbf{I}_{\tilde{m} \times \tilde{m}} \end{bmatrix}, \tilde{\boldsymbol{\omega}}_u = \begin{bmatrix} \frac{v_{\text{DC}}}{2} & i_{L,\max} & \frac{v_{\text{DC}}}{2} & i_{L,\max} \end{bmatrix}^\top, \quad (5.3)$$

results in a projected feasible set formulation as follows:

$$\mathbb{F}_{\infty \rightarrow 0} = \left\{ \begin{bmatrix} \mathbf{y}_k \\ \tilde{\mathbf{u}}_k \end{bmatrix} \in \mathbb{R}^{n+\tilde{m}} \left| \tilde{\mathbf{G}}_{\text{feas},d} \begin{bmatrix} \mathbf{y}_k \\ \tilde{\mathbf{u}}_k \end{bmatrix} \leq \tilde{\mathbf{e}}_{\text{feas},d} \right. \right\}. \quad (5.4)$$

Since the load current is considered as a system input (cf. (5.1)), $\mathbb{F}_{\infty \rightarrow 0}$ results from the projection of $\mathbb{F}_{k_f \rightarrow 0}$ onto the state-action space, represented by four dimensions, including the load current. This has the advantage that the measured load current can be considered, while the influence of the future load current range is taken into account by the projection. A one-step prediction of the load current is still necessary in order to compensate for the control delay, as described in Sec. 3.2. The matrices $\tilde{\mathbf{G}}_{\text{feas},d}$ and $\tilde{\mathbf{e}}_{\text{feas},d}$ can then be used in the safeguard to evaluate the action $u_{\text{RL},k}$ chosen by the RL agent with regard to safety and alter it into the safe action $u_{\text{SG},k}$ if necessary, such that no future system constraints are violated:

$$\begin{aligned} u_{\text{SG},k} &= \arg \min_{u_k, \vartheta} (u_k - u_{\text{RL},k})^2 + \psi \vartheta, \\ \text{s.t.} \quad & \begin{bmatrix} \tilde{\mathbf{G}}_{\text{feas},d} & -\mathbf{1}_{l \times 1} \\ \mathbf{0}_{1 \times (n+m+o)} & -1 \end{bmatrix} \begin{bmatrix} \mathbf{y}_k \\ \tilde{\mathbf{u}}_k \\ \vartheta \end{bmatrix} \leq \begin{bmatrix} \tilde{\mathbf{e}}_{\text{feas},d} \\ 0 \end{bmatrix}. \end{aligned} \quad (5.5)$$

Note that the constraints are defined based on $\tilde{\mathbf{u}}_k$, while the minimized metric only regards the system input u_k . Therefore, the number of variables to be optimized here is the same as in (4.7), but taking the load current into account leads to additional inequality conditions in (5.5). This is insignificant in terms of the computational effort required for the optimization while extending the usability to the entire application range required for a voltage-forming inverter.

To sustain the advantage of inherent adaptivity and no requirement of any a priori parameter knowledge, the model parameters required are identified using the toolchain introduced in Sec. 4.3. The identified matrices $\hat{\mathbf{A}}_{\mathbf{y}}$ and $\hat{\mathbf{B}}_{\mathbf{y}}$ are then used to calculate $\tilde{\mathbf{G}}_{\text{feas},d}$ and $\tilde{\mathbf{e}}_{\text{feas},d}$, defining the constraints required for the optimization within the safeguard. Unlike in Sec. 4.2, the load is not part of the $\mathbf{A}_{\mathbf{y}}$ matrix here. In order to identify the corresponding components in $\tilde{\mathbf{B}}_{\mathbf{y}}$, sufficient excitation by the load current is required. This is ensured in the following by a non-zero load current during the identification process.

5.2 Training Behavior

The learning task for the RL-based controller is encoded via the reward function (3.1), which is universally applicable to voltage-forming inverters, regardless of filter order and dimensioning, and has already been used in Sec. 4. Similar to Sec. 4.4, the HPs are determined based on about 3300 samples using the HPO framework described in Sec. 3.6 with the experiment shown in Fig. 5.1. Again, a real-time simulation of that experiment here is used for HPO. The determined HPs are listed in Tab. A.2, which are used for

training in the following. Additionally, the learning curve of the HPO and further details are shown in Sec. A.2.2.

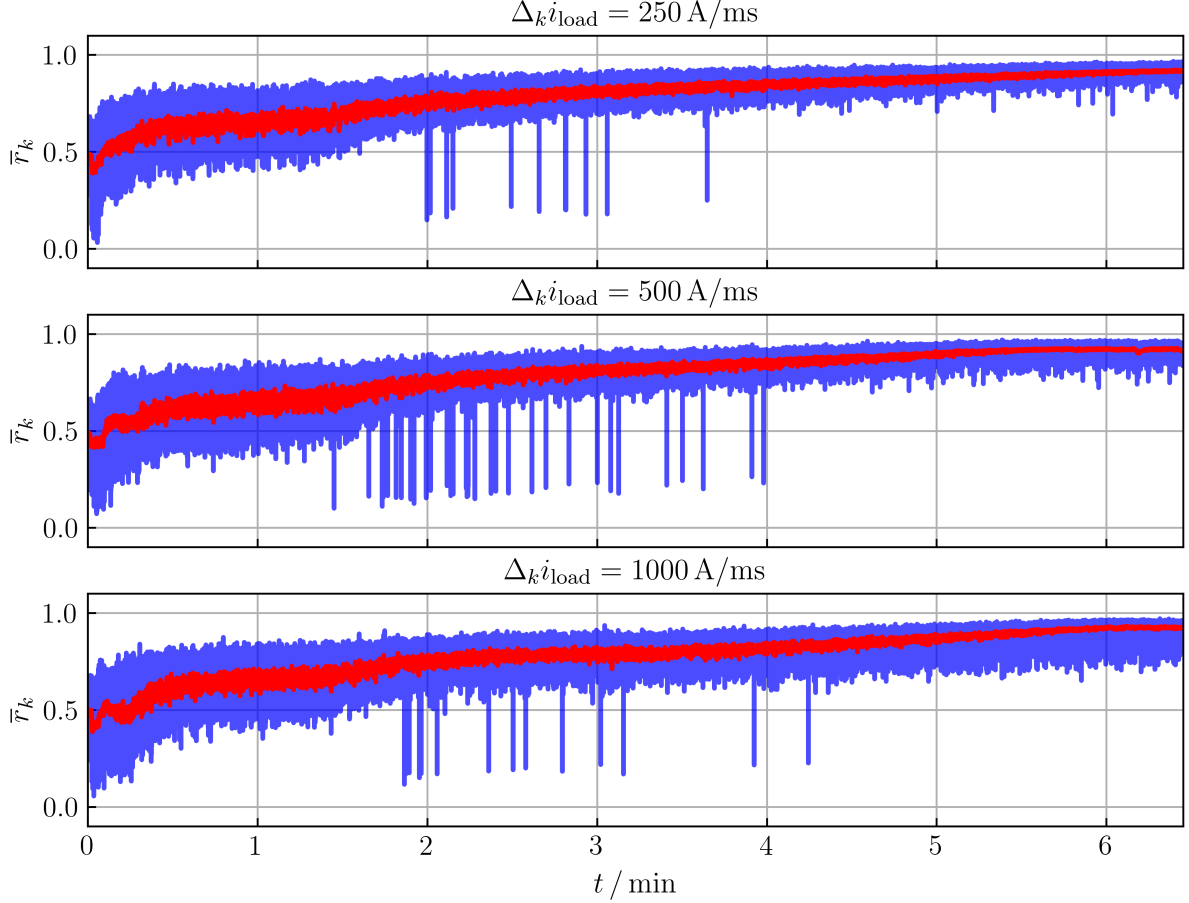


Fig. 5.2: Learning curves for different load settings, represented by the undiscounted, mean reward per learning step of the 3 agents trained per setting in blue. A moving average with window size 1000 is shown in red.

Before each training run, the data-driven safeguard according to Sec. 5.1 is calculated, with algorithm 1 converging after two steps. During the training, the active and reactive power of the load is changed based on a random walk within the range $S_{\text{Load}} \in [0, 250 \text{ kVA}]$, with $\varphi_{\text{Load}} \in [0, 2\pi]$, defining the angle of the grid voltage relative to the load current. The resulting load current amplitudes are scaled up stepwise every 1000 learning steps, from 100 A at the beginning of the training to 500 A after 5000 learning steps. At the same time, the probability of a reference change is scaled down over the course of the training, as already introduced in Sec. 4.5. In the FCS-MPC used to implement the load current, a changing reference in the internal control structure is limited by a rate limiter. To investigate the effects of this limitation on learning, several experiments are conducted with different limits $\Delta_k i \in \{250, 500, 1000\} \frac{\text{A}}{\text{ms}}$ for the rate limiter. Stable and successful training was possible with all three settings, although with the last setting the possible

$\Delta_k i$ is greater than the permitted capacitor current at the LC filter. This can be seen in the behavior of the learning curves of the RL controllers for the different $\Delta_k i$ settings in Fig. 5.2. The fact that the dispersion of all blue curves decreases towards the end of training indicates good convergence and thus a robust training toolchain with low variance for all $\Delta_k i$ settings. This is also underlined by the fact that after 6 minutes of training,

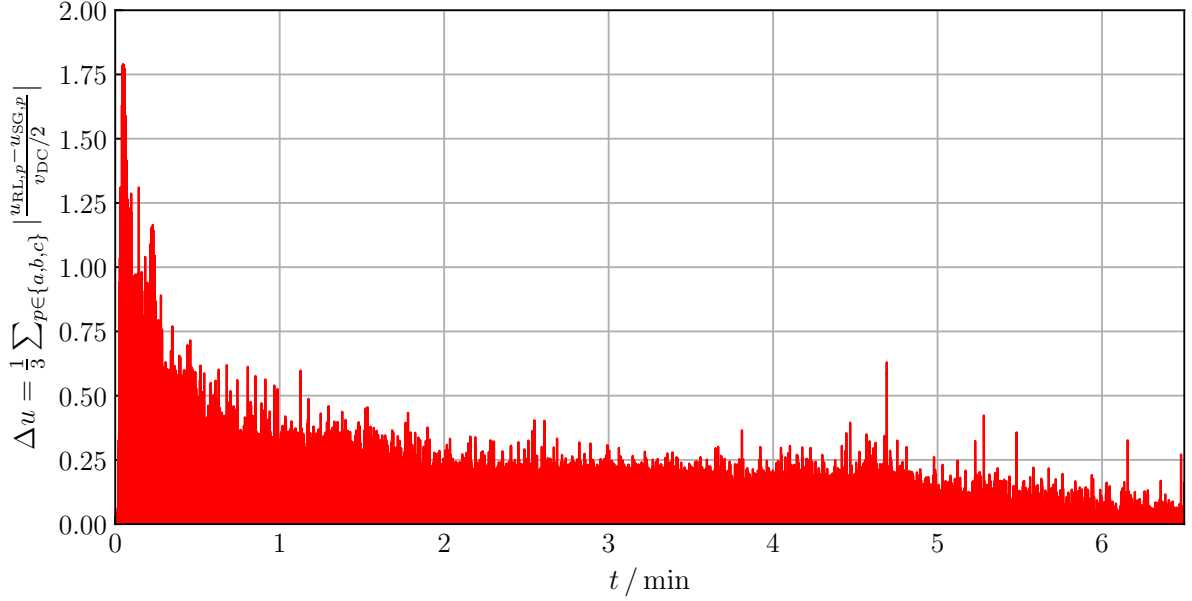


Fig. 5.3: Average safeguard intervention magnitude of the RL_{1000} over the course of one training.

each RL-based controller achieves a high reward close to the theoretical maximum of 1 (which is practically unattainable due to random load changes during training). In addition, the influence of $\Delta_k i$ on the learning curve can be seen. As the permissible load current slope increases, the dispersion of the blue curve increases. This is because the learning problem becomes more difficult with larger $\Delta_k i$. Furthermore, a drop in grid voltage cannot always be avoided when the load current slope is large, which is explained in detail in Sec. 5.4. Since the feasible set $\mathbb{F}_{\infty \rightarrow 0}$ here contains the load current, as explained in Sec. 5.1, a graphical interpretation of the functionality of the safeguard comparable to that presented in Sec. 4.3 is not possible. Instead, Fig. 5.3 shows the effect of the safeguard during a training run of the RL_{1000} agent, corresponding to the best agent from the bottom learning curve in Fig. 5.2. The figure shows the (average) magnitude of safeguard intervention Δu over the course of the training. It can be seen that very large changes are necessary at the beginning of training, which decreases almost entirely towards the end. This shows that the RL agent learns through the penalty term in the reward function to operate the plant independently of superimposed safety measures.

To demonstrate operation under full load variability, the best-performing agent RL_{1000} from the trainings with $\Delta_k i = 1000 \frac{A}{ms}$ is used for the following investigations, while $\Delta_k i = 1000 \frac{A}{ms}$ is applied as the limit for the load MPC.

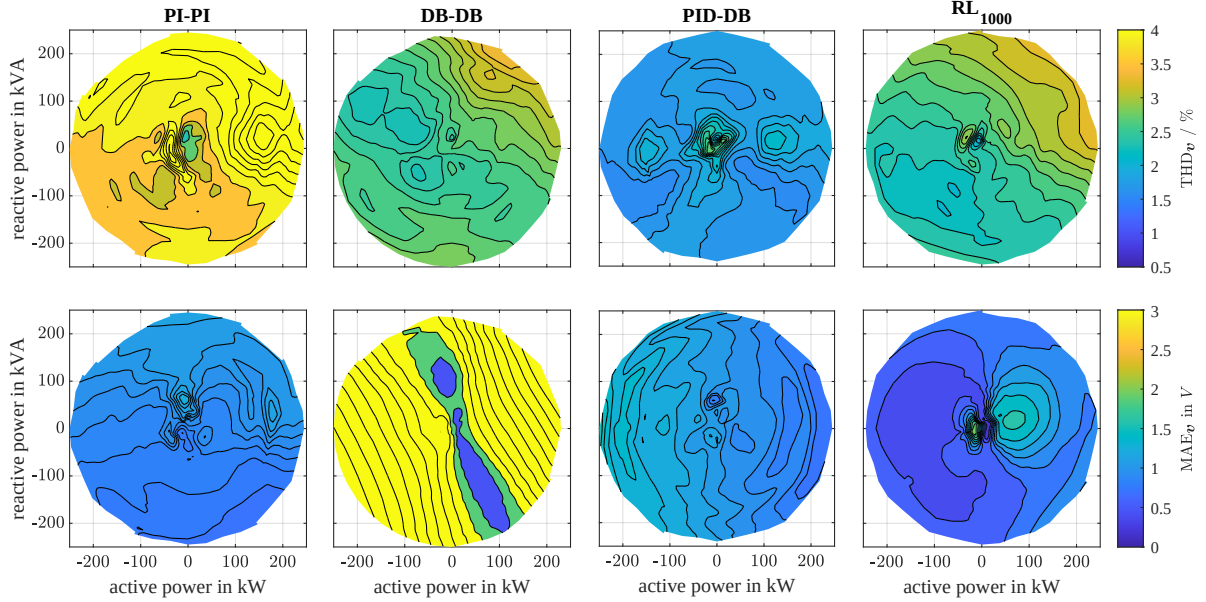


Fig. 5.4: Comparison of the steady-state behavior between three model-based, expert-driven control approaches and an RL-based controller.

5.3 Steady-State Behavior

After training, the steady-state performance of the RL-based controller is compared to a cascaded PI controller (PI-PI, with outer voltage control loop and inner current control loop, implemented in a rotating reference frame, as already described in Sec. 4.6), a predictive dead-beat controller (DB-DB) and a PID-voltage controller with a subordinate dead-beat current controller (PID-DB). The PI-based controllers are designed according to [90]. A more conservative tuning than in Sec. 4.6 was chosen, because the deliberately aggressive tuning possible with the constant, resistive load would lead to errors in the FPGA with the load MPC used here. Using the parameters defined in Tab. 3.1, the following controller configuration results according to [90]:

$$\begin{aligned} K_{p,c} &= 0.232 \text{ mV/A} & \text{and} & \quad K_{i,c} = 0.073 \text{ V/(As)}, \\ K_{p,v} &= 0.863 \text{ A/V} & \text{and} & \quad K_{i,v} = 1224.021 \text{ A/(Vs)}. \end{aligned} \quad (5.6)$$

The predictive dead-beat controllers are implemented according to [91, 92], while the required state-space model is calculated on the basis of the parameters defined in Tab. 3.1. The basic equations required for the implementation of the predictive dead-beat controller are summarized in Sec. A.3. For further implementation details on the considered benchmark controllers, the reader is referred to [90–92]. In order to investigate the performance

Tab. 5.1: Average steady-state performance $\overline{\text{MAE}}$ and $\overline{\text{THD}}$ over the full operation range for different controllers. In case of RL control, three agents trained per setting, additionally the best performing agent's result $\overline{\text{MAE}}_{\text{B}}$ and $\overline{\text{THD}}_{\text{B}}$ are listed.

	PI-PI	DB-DB	PID-DB	RL ₂₅₀	RL ₅₀₀	RL ₁₀₀₀
$\overline{\text{MAE}}/\text{V}$	0.93	7.06	1.11	0.81	1.62	0.85
$\overline{\text{THD}}/\%$	4.12	2.68	1.89	3.21	3.06	2.76
$\overline{\text{MAE}}_{\text{B}}/\text{V}$				0.67	1.18	0.81
$\overline{\text{THD}}_{\text{B}}/\%$				3.18	2.90	2.64

of the RL controller in the steady state, various operating points are approached in the range of $S_{\text{Load}} \in [0, 250 \text{ kVA}]$, with $\varphi_{\text{Load}} \in [0, 2\pi]$. The MAE of the measured voltages $\mathbf{v}_{\text{abc}}$ and the respective THD are calculated according to (4.8), measured over a period of $T_{\text{meas}} = 3 \text{ s}$. On the right side of Fig. 5.4, it can be seen that the RL controller achieves an average MAE of 0.81 V and an average THD of 2.64 % over the whole operation range. The achieved MAE of the RL agent is comparable to that of the PI-PI controller, which performs slightly worse than the RL. The higher error of the PI controller is caused by the high dynamic distortions from the FCS-MPC load controller. Tests showed that if the load side is also implemented by a PI-based controller, the MAE of the cascaded PI controller shown in Fig. 5.4 is significantly reduced. Nevertheless, the FCS-MPC was retained on the load side, as a load PI controller does not have the necessary transient performance to ensure the desired training cases for the RL-based controller and to enable the following transient tests. As depicted in Fig. 5.4, the DB-DB controller achieves a better THD than the PI-PI controller, which is comparable to that of the RL controller, but shows a higher MAE, which increases linearly with increasing power. This is likely caused by mismatches between the model used in the controller and the plant. This can be improved by the PID-DB controller, while this effect is not present in the error plot of the RL controller, which benefits from the data-driven controller design (assuming safe operation). To investigate the influence of the different $\Delta_k i \in \{250, 500, 1000\} \frac{\text{A}}{\text{ms}}$, three RL _{$\Delta_k i$} agents per setting are trained and the average performance of the three agents over the full operation range is listed in Tab. 5.1.

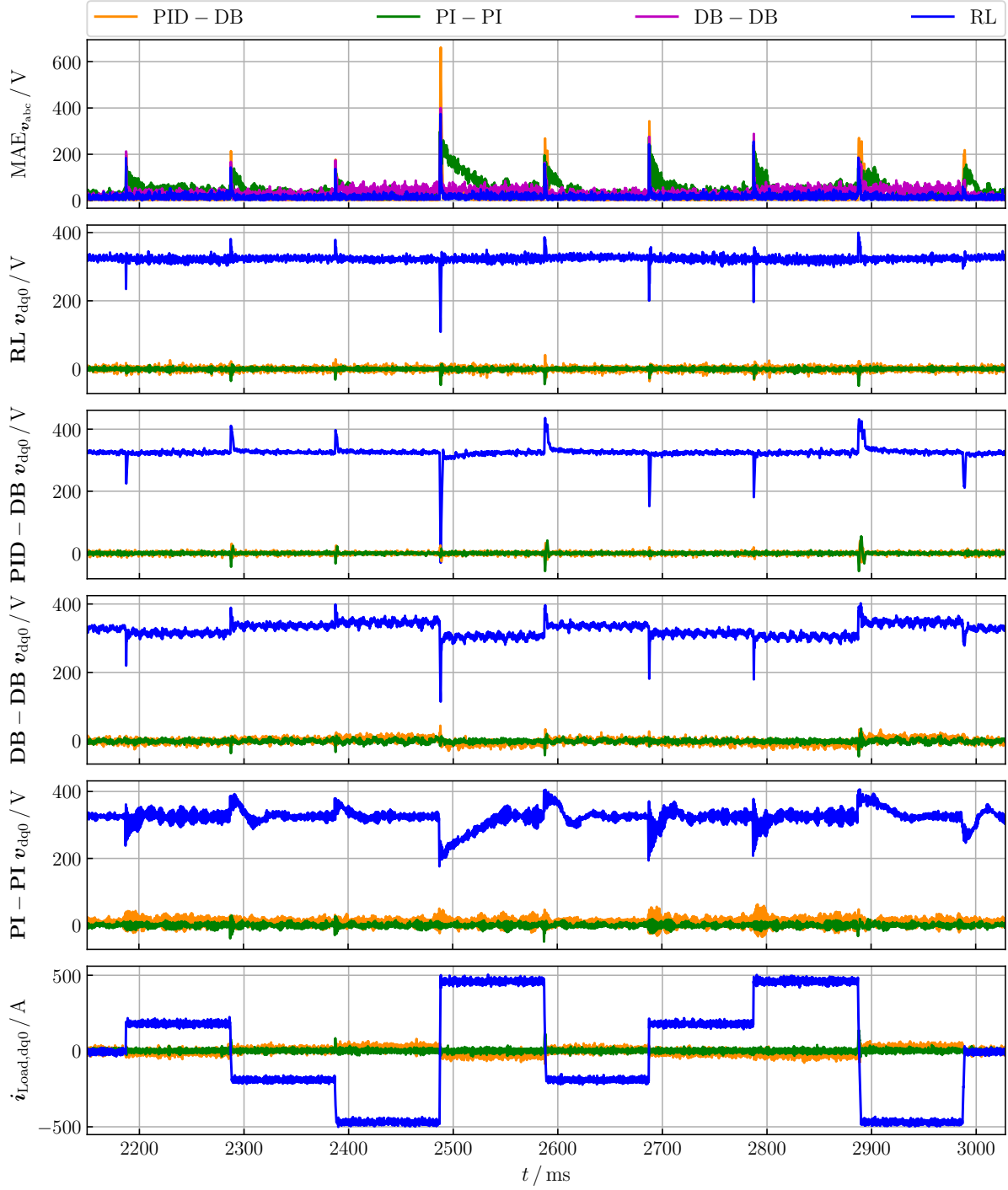


Fig. 5.5: Transient control performance comparison of RL₁₀₀₀ and baseline controllers for different consecutive load steps.

5.4 Transient Behavior

To investigate the transient behavior of the developed RL-based controller and to compare it to the expert-driven controllers introduced above, different scenarios are tested in the following.

5.4.1 Active Load

To evaluate the performance of the RL controller during transients caused by an active load, a test case with different load steps and load sheddings in the d-component of the load current is used, while $i_{q,0}^* = 0$ A. The part of the test case with the rate of change limit $\Delta_k i_{\text{load}} = 1000 \frac{\text{A}}{\text{ms}}$ is shown in the bottom subplot of Fig. 5.5, while the whole test case investigated for comparison is repeated in a similar form for $\Delta_k i = 250 \frac{\text{A}}{\text{ms}}$ and $\Delta_k i = 500 \frac{\text{A}}{\text{ms}}$. Tab. 5.2 lists the average performance over the whole test case $\overline{\text{MAE}}$ of the three RL controllers as well as the performance of the three expert-driven control approaches. Additionally, $\overline{\text{MAE}}_{\text{B}}$ and the rise time t_{rise} are provided for the best trained agent per setting. The rise time is calculated according to [104] for the period of time after a load step when 0.95 % of the reference value is reached in order to take a steady-state error into account.

Tab. 5.2: Average transient performance $\overline{\text{MAE}}$ of different controllers measured with the test case. For RL control, three agents trained per setting, additionally, $\overline{\text{MAE}}_{\text{B}}$ and t_{rise} of the best performing agent on the test case is listed.

	PI-PI	DB-DB	PID-DB	RL ₂₅₀	RL ₅₀₀	RL ₁₀₀₀
$\overline{\text{MAE}}/\text{V}$	14.30	11.04	5.38	6.02	7.69	5.68
$\overline{\text{MAE}}_{\text{B}}/\text{V}$				5.91	6.31	5.35
$t_{\text{rise}}/\text{ms}$	11.50	2.12	2.71	0.70	0.72	0.87

Fig. 5.6 shows the RL₁₀₀₀ controller performing a load step of the d-current from 500 A to −500 A. As can be seen from the inconsistent slope of $i_{\text{Load,d}}$, physical constraints limit the speed of the step response performed by the load FCS-MPC, while the RL controller is able to perform the load step with an acceptable error. A load step in the other direction is depicted in Fig. 5.7. Here, the load controller realizes the full $\Delta_k i$, resulting in a load change of 1 kA within 1 ms. A clear voltage drop can be seen here, where the voltage in phase a is just at its maximum, which leaves less room to supply the current required to maintain the capacitor voltage, when the load changes from −500 A to 500 A. This corresponds to a load step from 98 % rated current to supplied to 98 % rated current absorbed. The prediction model in the safeguard within the control structure assumes the load current amplitude to be constant. However, in this example the amplitude of the load current changes by 100 A per time step, which is already above

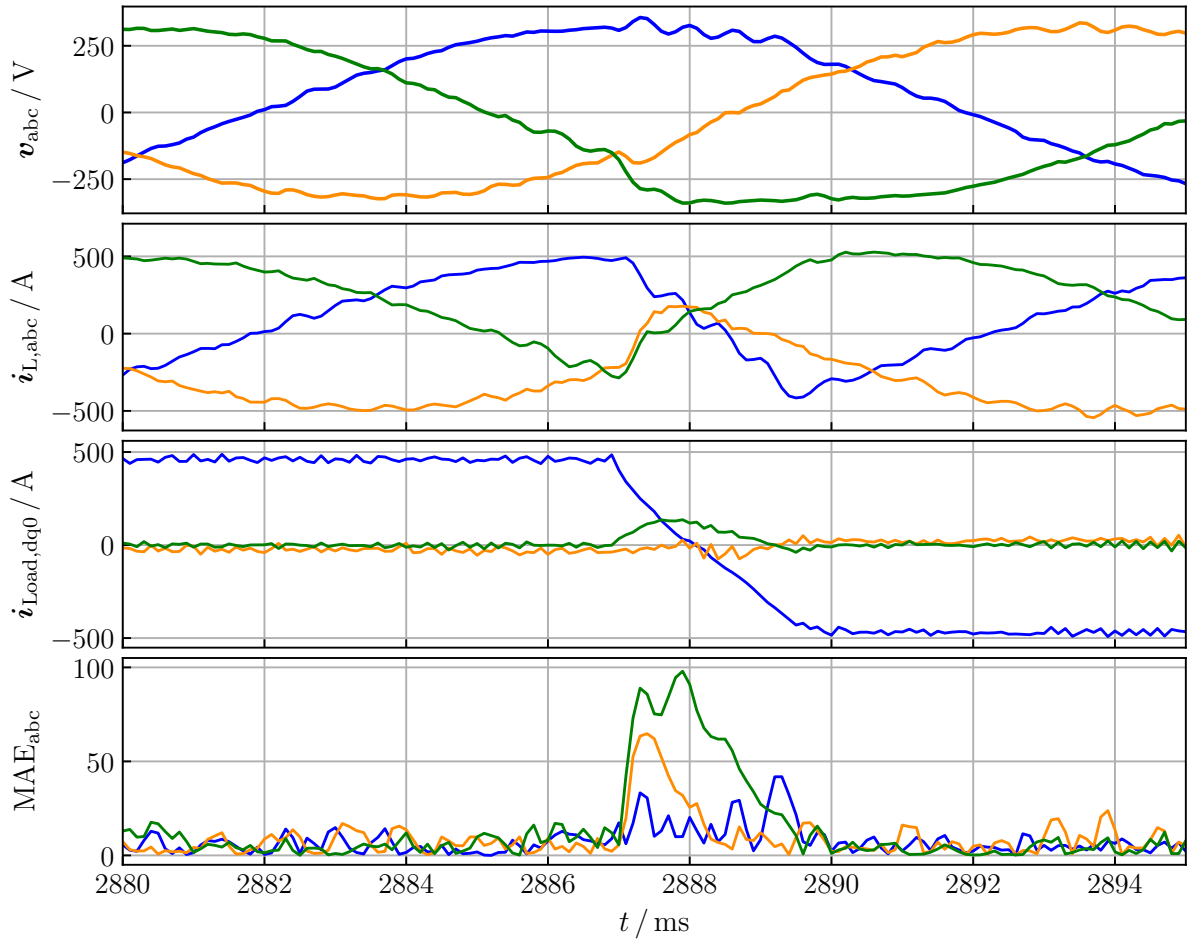


Fig. 5.6: RL₁₀₀₀ controller performing a load step of the d-current from 500 A to −500 A with $\Delta_k i = 1000$ A/ms.

the current limit $I_{C,max} = 90$ A of the capacitor, resulting in a prediction error and non-ideal optimization results within the safeguard. The same load step performed by the other control approaches can be seen in Fig. 5.7, too. The DB-DB controller is able to perform the load step with a similar voltage MAE as the RL controller, like depicted in the lowest subplot, while the PI-based controllers show a higher error.

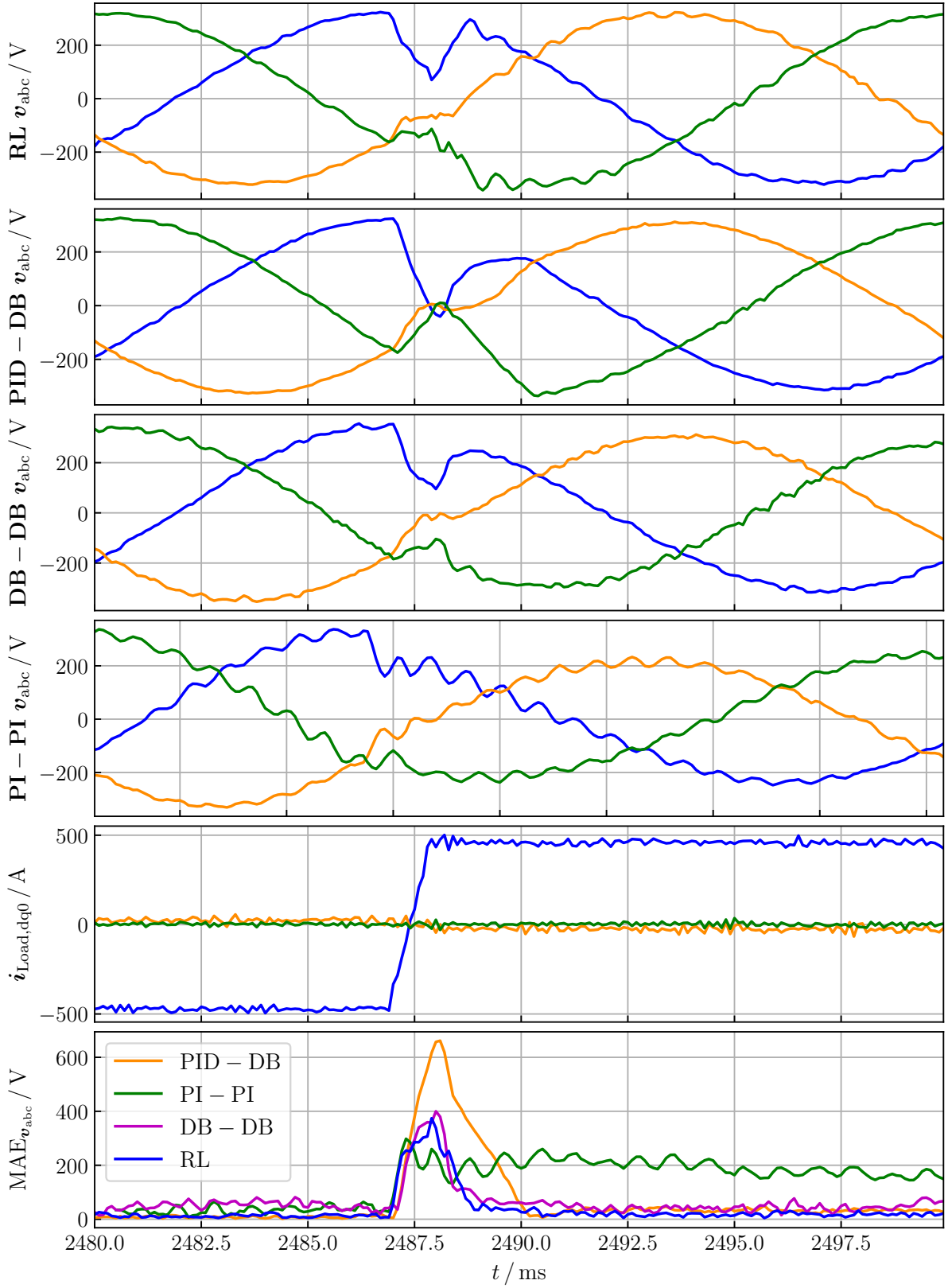


Fig. 5.7: RL_{1000} controller and expert-driven control approaches performing a load step of the d-current from -500 A to 500 A with $\Delta_k i_{load} = 1000 \frac{A}{ms}$.

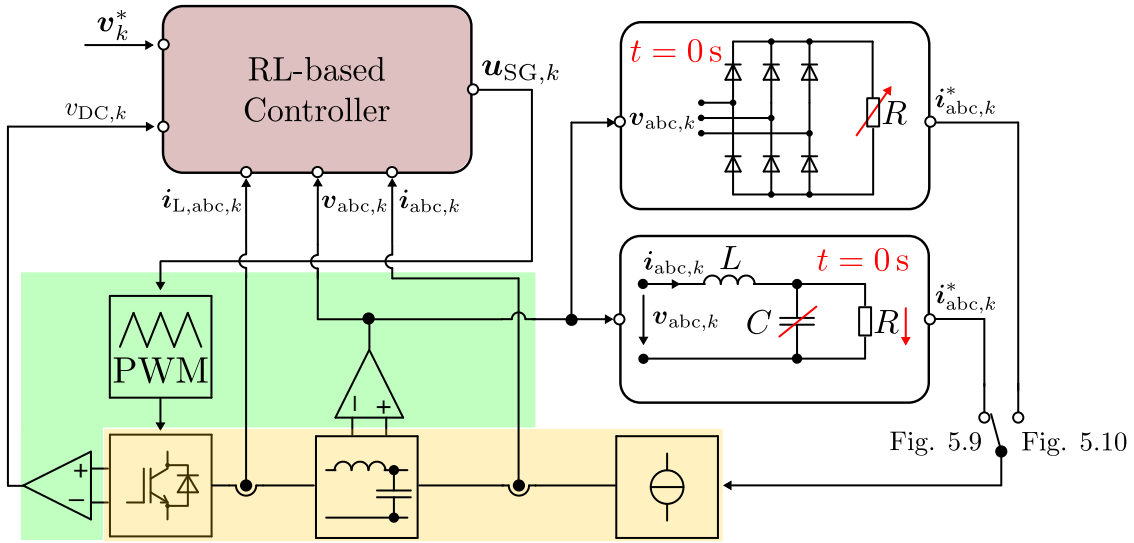


Fig. 5.8: Virtual load impedance (experiment setting Fig. 5.9) and virtual passive rectifier (experiment setting Fig. 5.10) approach for calculating the reference value for the load MPC based on the measured grid voltage, so that the load behaves like a passive variable load or a non-linear diode rectifier. For both scenarios, at $t = 0$ s a load step is injected: experiment setting for Fig. 5.9: the capacitor disconnects (emulates a failure) and the resistor is drastically reduced; experiment setting for Fig. 5.10: the resistive load changes.

5.4.2 Passive Linear Time-Variant Load

The performance of the controllers supplying a passive variable load will be tested in the following. The current setpoint for the load MPC is calculated below as a function of the measured grid voltage in such a way that the load inverter emulates a passive load. According to Fig. 5.8, the admittance of an LCR load:

$$\underline{H}(j\omega) = \frac{\underline{i}(j\omega)}{\underline{v}(j\omega)} = \underline{Y}(j\omega) = \frac{1 + j\omega RC}{R - \omega^2 RLC + j\omega L}, \quad (5.7)$$

delivers the transfer function on whose basis the current setpoint:

$$i^* = \text{Re}(\underline{v}(j\omega) \cdot \underline{Y}(j\omega)), \quad (5.8)$$

can be calculated. This setpoint is then used for the load MPC, which ensures that the load behaves according to the LCR parameterization. A corresponding experiment is shown in Fig. 5.8. Until $t = 0$ s the LCR load for the setpoint calculation is parameterized with $L = 4$ mH, $C = 10$ μ F and $R = 2$ Ω . At the time $t = 0$ s, the load changes to an LR load with the parameters $L = 4$ mH and $R = 1$ Ω . This could be caused, e.g., by a defect of the capacitor. As can be seen in Fig. 5.9, this changes the load behavior and the load current, respectively. In the grid voltage $\underline{v}_{abc}$ provided by the RL-based controller, only a marginal effect can be seen, which is also made clear by the low MAE in the lowest subplot.

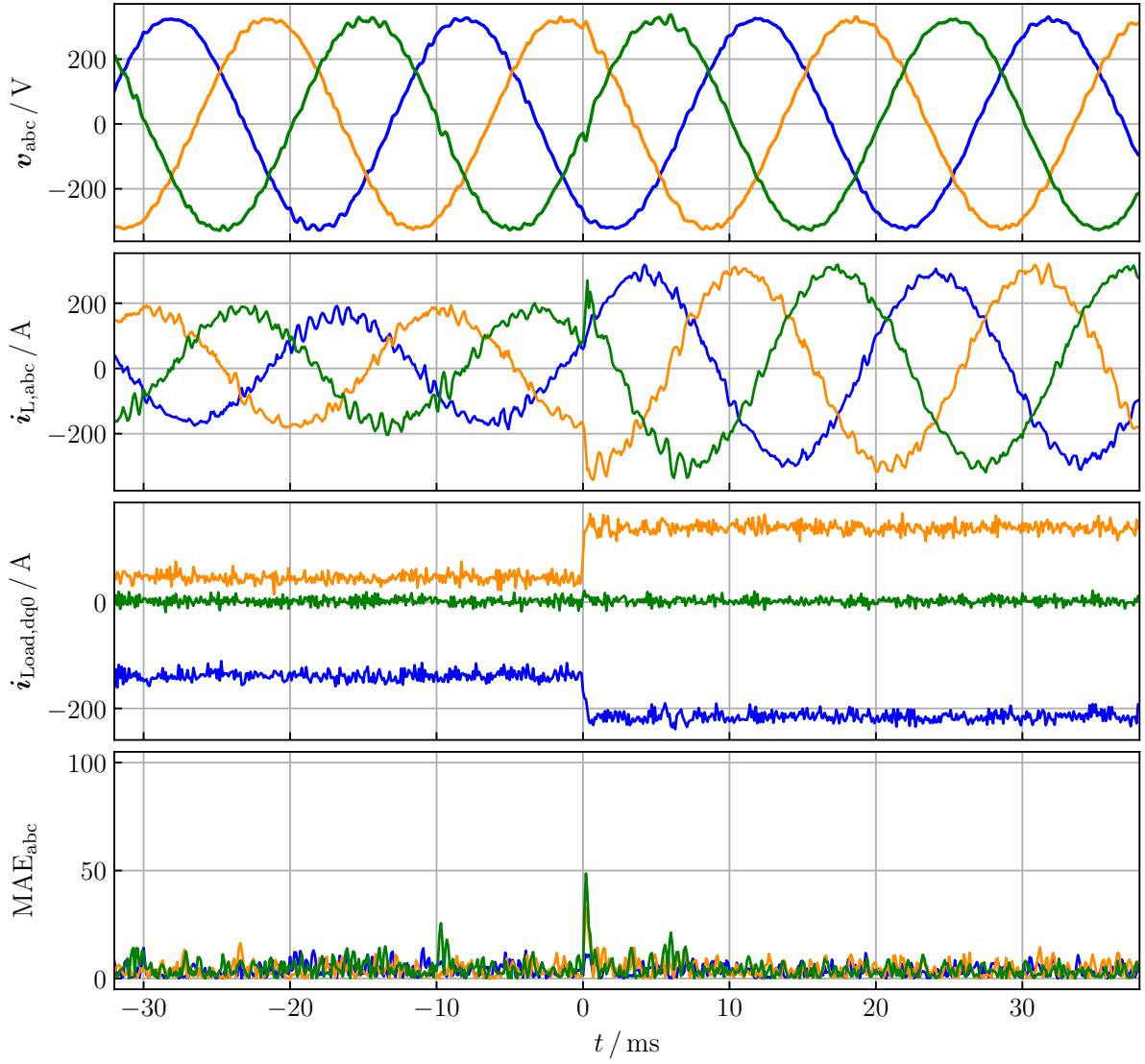


Fig. 5.9: RL₁₀₀₀ controller performing a load step at $t = 0$ s caused by the impedance change depicted in Fig. 5.8.

5.4.2.1 Passive Non-Linear Time-Variant Load

In order to prove its applicability to arbitrary loads, the controller should also be tested in supplying non-linear loads. As shown in Fig. 5.8, analogous to the experiment with the virtual impedance in the previous paragraph, the reference for the load MPC is calculated on the basis of a diode rectifier with ohmic load, representing a non-linear load behavior. At time $t = 0$ s, a load step from $R = 3\Omega$ to $R = 1.5\Omega$ occurs. The resulting controller performance of the RL₁₀₀₀ controller can be seen in Fig. 5.10. Compared to the virtual load impedance, the MAE is higher in this case. This is because the load is no longer sinusoidal, but changes steplike, as already seen in Sec. 5.4.1 for the corresponding load steps. The MAE in the experiment shown here is within the range of the results shown in Sec. 5.4.1. Thus, it can be concluded that the RL-based controller can supply the non-linear load with the desired voltage, even though it has only been trained using sinusoidal loads.

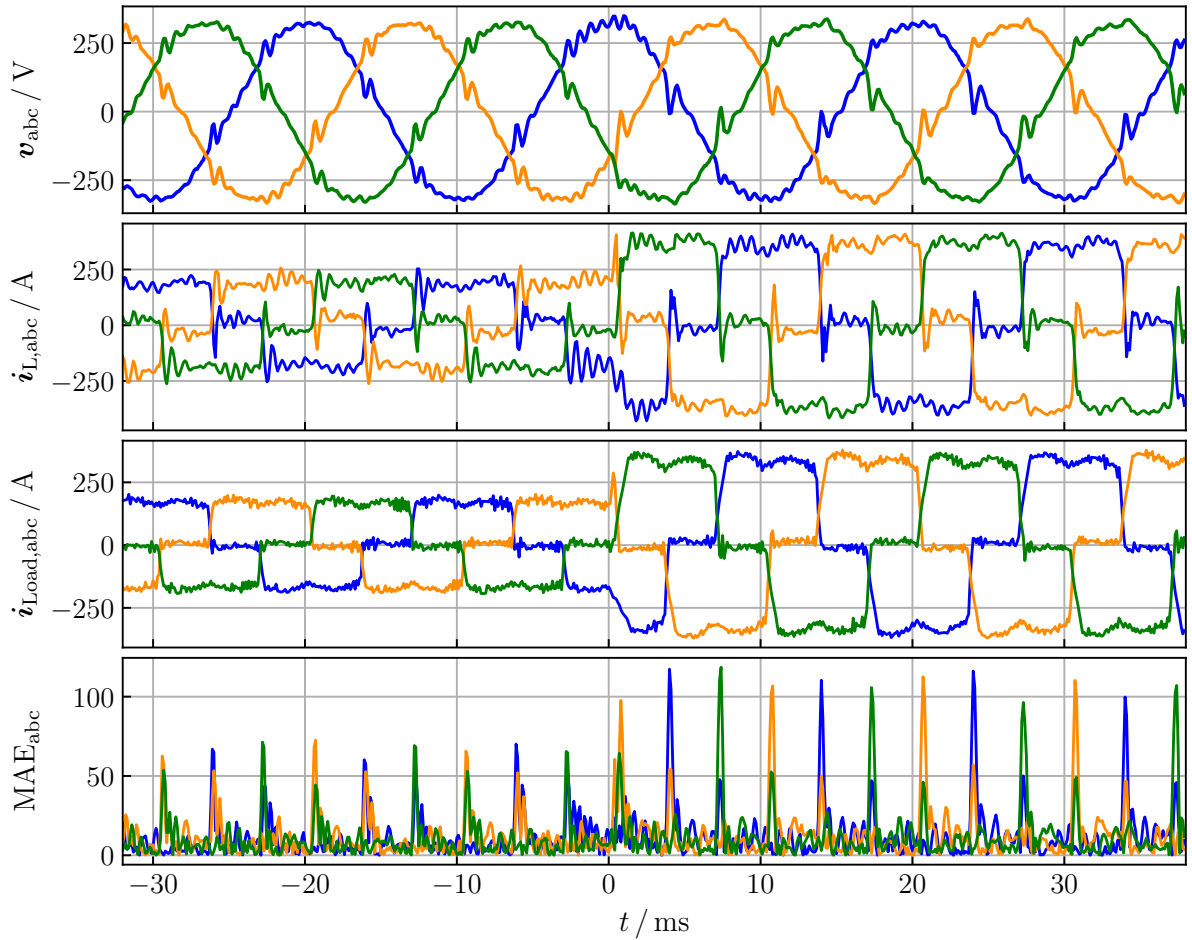


Fig. 5.10: RL₁₀₀₀ controller performing a non-linear load step at $t = 0$ s caused by the load change depicted in Fig. 5.8.

5.4.3 Black-Start and Emergency Shutdown Capability

To ensure the applicability of the RL-based controller for voltage-forming inverters for grid operation, black-start and emergency shutdown capability is required. To this end, it is first shown how the RL_{1000} controller performs a black-start under load on the left in Fig. 5.11. Here, the active load according to Sec. 5.4.1 is used. As can be seen, the desired grid voltage can be provided after a black-start under load within the millisecond range without violating constraints. The same applies to the shutdown event shown on the right side of Fig. 5.11, which is safely performed by the RL_{1000} controller in the millisecond range.

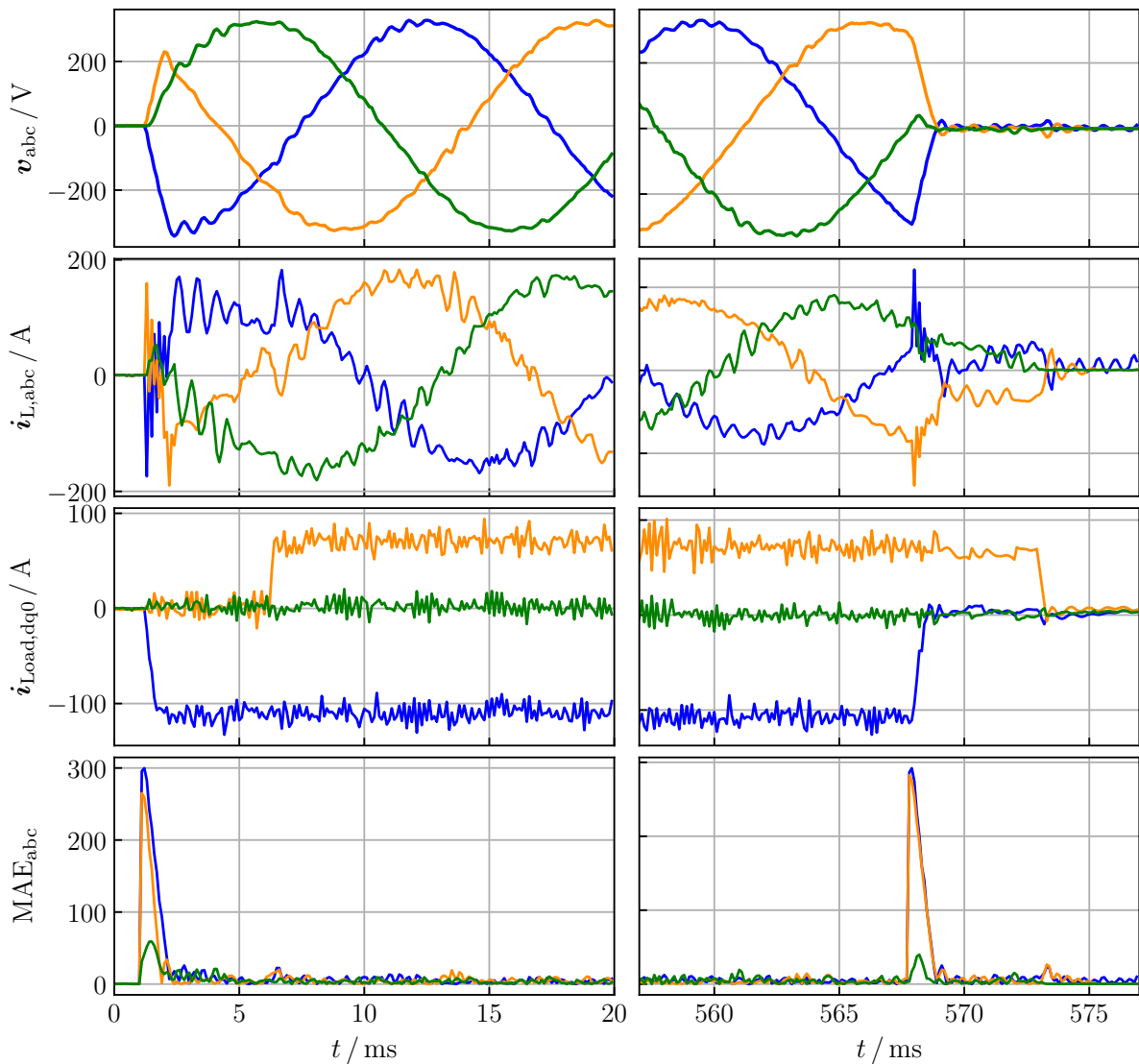


Fig. 5.11: RL_{1000} controller performing a black start (left) and a shutdown (right) under load.

knowledge of the plant system. Moreover, the interaction of heterogeneous converters with different component-level controllers, such as MPC, is also evaluated empirically.

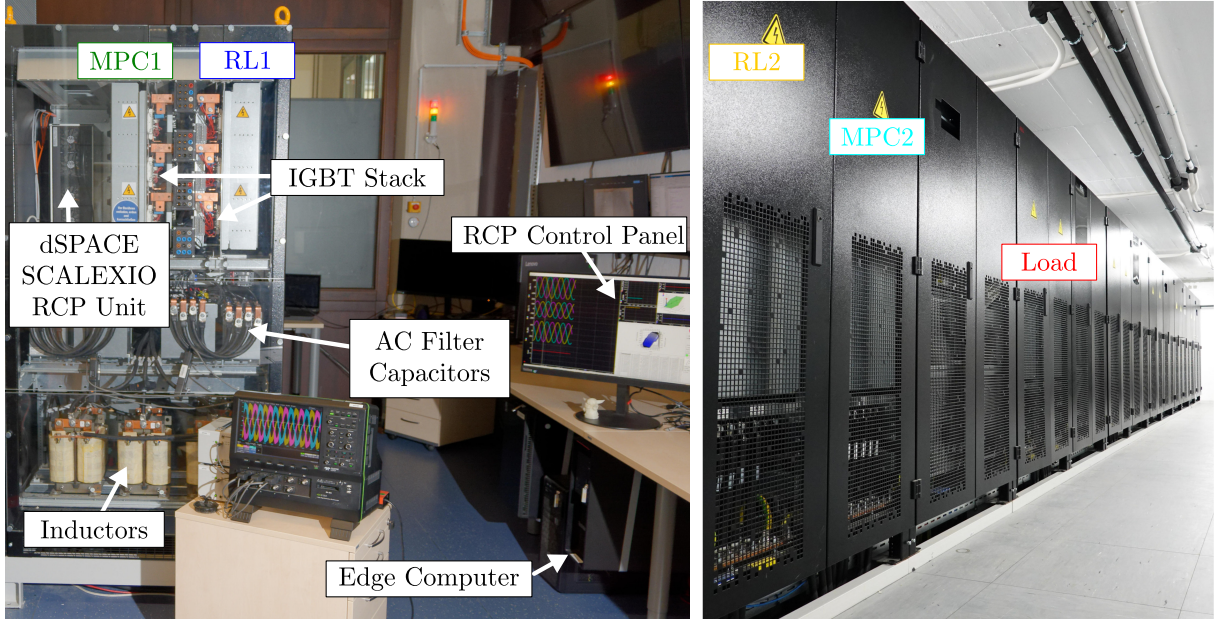


Fig. 6.2: Experiment MG setting with up to five 250 kVA industrial-grade inverters in the MG-lab.

During the following experiments, up to five inverters in the MG-lab depicted in Fig. 6.2 are used. Four of them are utilized as grid-forming inverters – two RL-based controlled (RL1 and RL2), and two FCS-MPC voltage controlled (MPC1 and MPC2) as introduced in [3] – while the fifth inverter acts as active load operating in current control mode and deployed as an FCS-MPC as already used in the experiments in Sec. 5. In the following, the two RL-based controllers are individually trained online utilizing the toolchain according Sec. 5. The droop and RL controllers are implemented on the CPU-card of the RCP unit and executed once per switching period at a switching frequency $f_s = 10$ kHz,

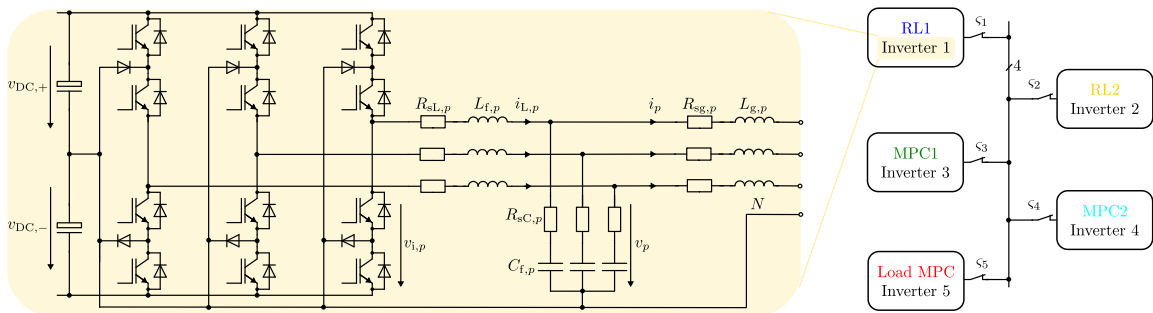


Fig. 6.3: Control plant of the voltage controllers (source) and grid configuration. The control plant (L filter) of the load controller on inverter 5 is not depicted.

as already introduced in the previous investigations. The FCS-MPCs are implemented on the FPGA card and operate at a control frequency of $f_{s,\text{MPC}} = 50 \text{ kHz}$. While the control plant of the grid-forming controllers, shown in Fig. 6.3, is an LC filter, the load current controller only utilizes an L filter (not depicted in Fig. 6.3). All hardware specifications of the experiments are listed in Tab. 3.1.

During the black-start of the islanded MG, the switches (cf. Fig. 6.1 and Fig. 6.3) are closed one after the other by means of PLL-based synchronization, while after synchronization the PLL is no longer actively used for control. The PLL and droop controllers are based on preliminary work [94]. Their development is outside the scope of this work. Independent control of active and reactive power, e.g., with the concept introduced in Sec. 2.5, is not possible if the reactance-to-resistance ratio (X/R) of the grid is low, as is typically the case in low-voltage MGs [98]. To counteract that, an inductive coupling between the inverters is realized by the inductor L_g before the point of common coupling (PCC) of each inverter as shown in Fig. 6.1 and Fig. 6.3 [94].

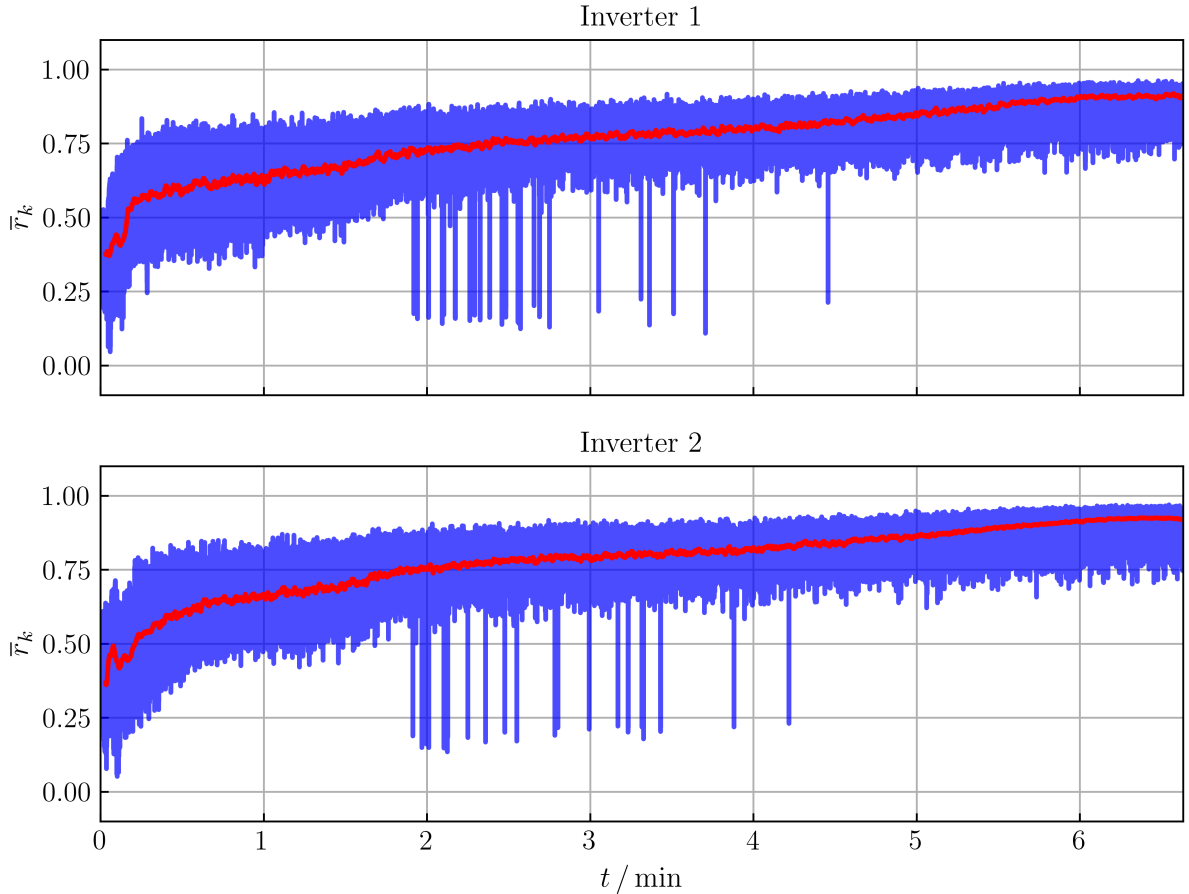


Fig. 6.4: Undiscounted, mean reward per learning step of the 3 trained agents in blue. A moving average with window size 1000 is shown in red.

6.1 Training Behavior

To train an RL-based controller for a voltage control task utilizing the toolchain described in Sec. 5, RL1 and RL2 are trained individually one after the other. Thereby, the current-controlled FCS-MPC on inverter 5 serves as load and ς_1 and ς_5 or ς_2 and ς_5 (cf. Fig. 6.3) are closed over the course of the training. The RL-based controller serves as a reference bus in each case, while the load current changes its amplitude and phase based on a random process as introduced in Sec. 5. The HPs listed in Tab. A.2 are used for each of the RL-based controllers. During training of the voltage control on component level, the top-level droop control is deactivated and a fixed reference voltage amplitude and frequency is used. The learning curves of the trained RL agents in Fig. 6.4 show that after about 6 minutes of training, it is possible to roll out RL-based voltage controllers to different inverters, indicated by a high reward close to the theoretical unity maximum. As can be seen in Fig. 6.2, RL1 is trained on the inverter that was also used for the previous experiments. However, RL2 is trained on a different inverter from the MG-lab, including system identification and safeguard calculation without prior knowledge of system parameters. The comparability of the learning curves confirms the general applicability of the developed toolchain. Three agents are trained per inverter and the best-performing agent of these trainings per inverter is used for the following investigations.

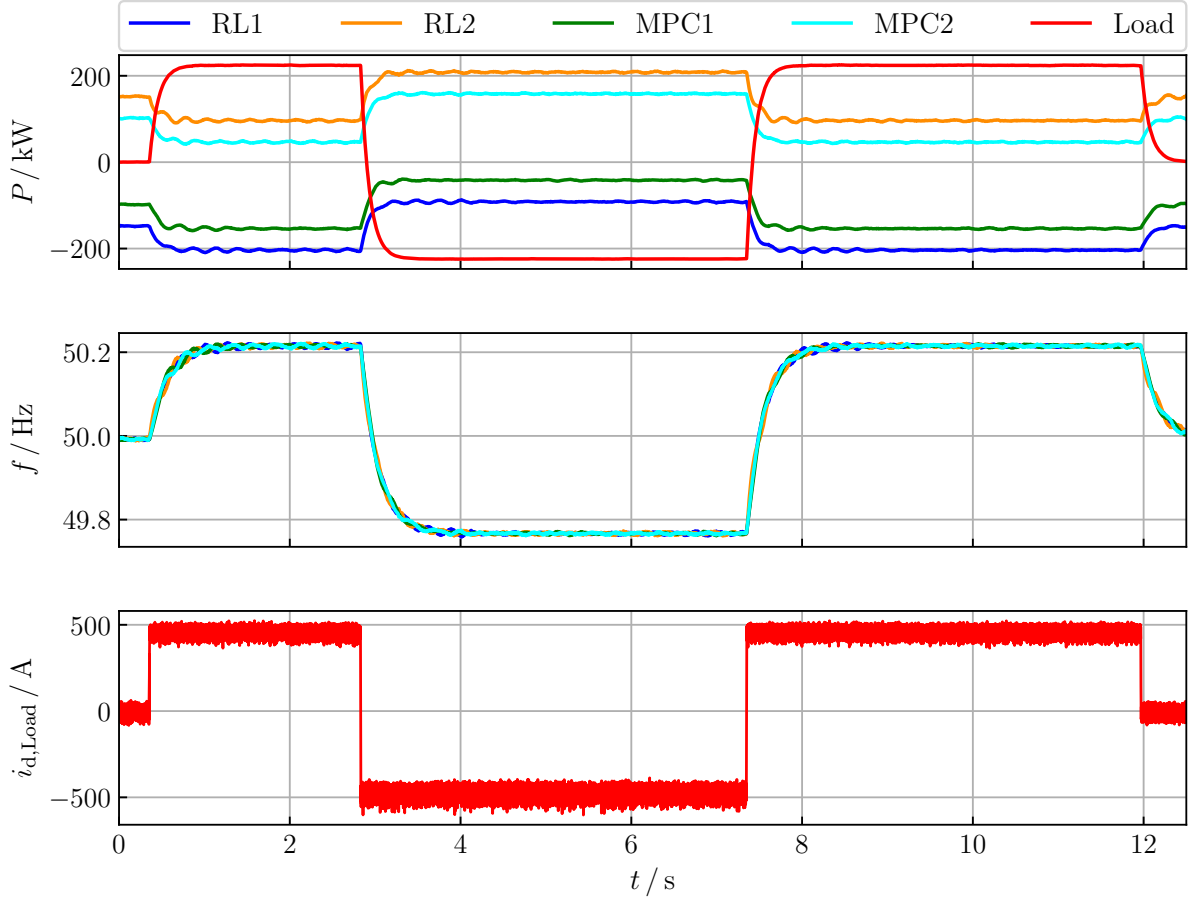


Fig. 6.5: Grid experiment for different active power loadings realized by the load, while the setpoints are configured as follows: $P_{RL1}^* = -150$ kW, $P_{RL2}^* = 150$ kW, $P_{MPC1}^* = -100$ kW and $P_{MPC2}^* = 100$ kW.

6.2 Islanded Grid Experiment

In order to investigate the functionality of the RL-based voltage controllers in an islanded MG, all five inverters are connected to form an islanded grid. For this purpose, the switches $\varsigma_{1,\dots,5}$ (cf. Fig. 6.3) are closed one after the other using the PLL-based synchronization. In the following, ς_1 is closed first and thus inverter 1 acts as a reference bus.

6.2.1 P-Droop Evaluation

To evaluate the functionality of the droop controllers, different setpoints P^* are approached. As shown in Fig. 6.5, two inverters act as power sources ($P_{RL2}^* = 150$ kW and $P_{MPC2}^* = 100$ kW) and two inverters act as power sinks, ($P_{RL1}^* = -150$ kW and $P_{MPC1}^* = -100$ kW) such that a power balance is given at $t = 0$ s. Therefore, the mea-

sured frequency at each inverter is 50 Hz initially. These setpoints were exemplary selected to ensure better graphical comparability of the active power curves of the various converters. Over the course of the shown experiment, the active load draws power from and delivers power to the grid. According to the droop-based control concept, the grid-forming inverters share the demanded load equally, regardless of whether they are controlled via an MPC or an RL-based controller, with the frequency changing according to (2.45). This can also be seen in the measured currents and voltages at the individual grid nodes, which are shown in Fig. 6.6 for the step at 0.855 s in Fig. 6.5. Prior to the load step, there is a power balance in which all inverters provide their power setpoints, meaning that two inverters act as a source and two inverters act as a load. During the load step performed by the load MPC, it can be seen that the current waveforms of the other four inverters change accordingly to share the demanded load. While RL2 and MPC2 reduce their current amplitude, RL1 and MPC1 increase their current amplitude. This can also be seen in Fig. 6.5. The voltage curves of all four grid-forming inverters do not feature any steps and each inverter can provide the target voltage despite changing loads. The allowed frequency deviation, which is used for power distribution, can be set according to (2.45) and is defined in Tab. 3.1.

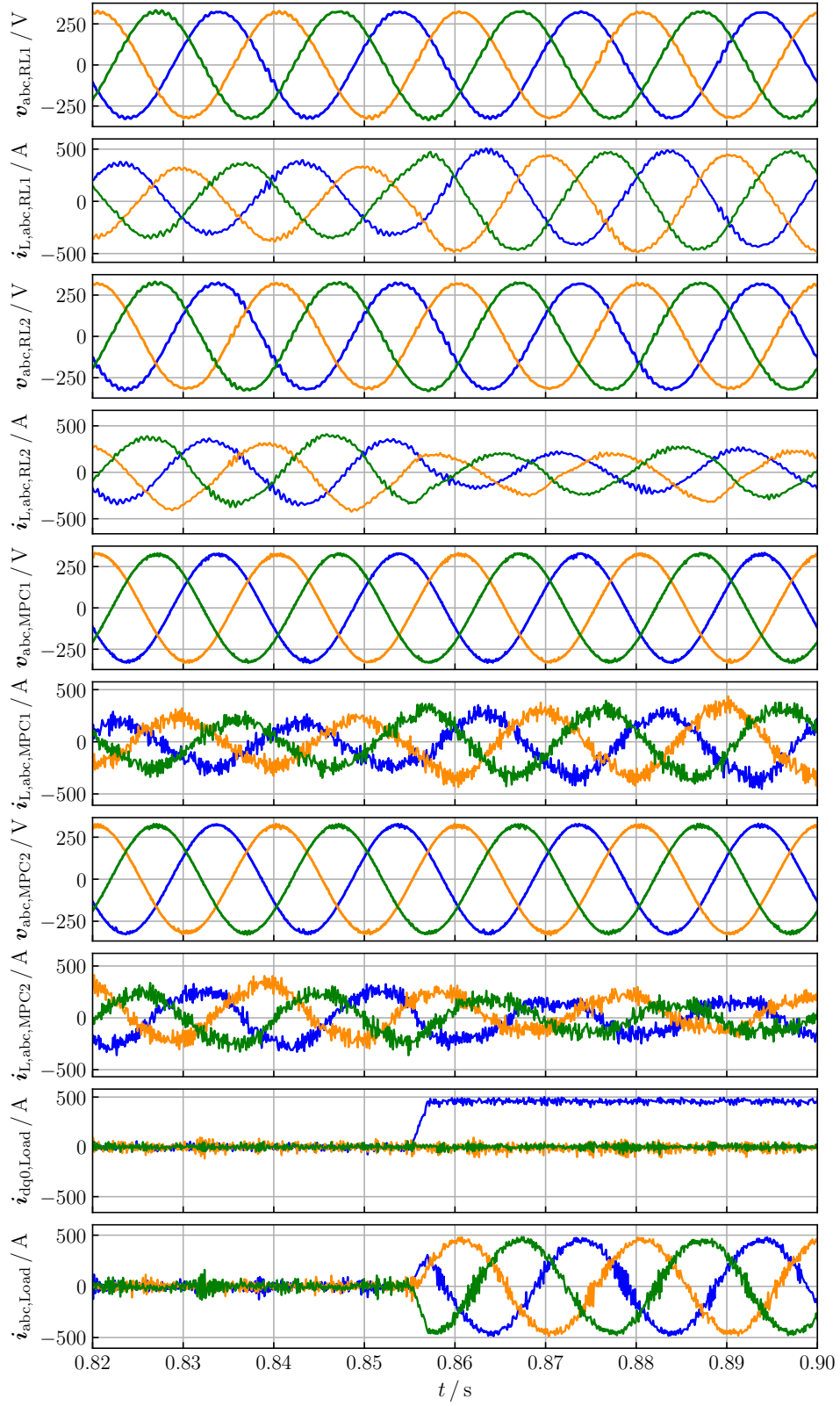


Fig. 6.6: Measured currents and voltages at the individual grid nodes for the step at 0.855 s in Fig. 6.5.

6.2.2 Q-Droop Evaluation

To evaluate the share of reactive power between the inverters, different magnitudes of reactive power are drawn from and provided to the grid by the load on inverter 5. Thereby, the setpoints $Q^* = 0$ VA are chosen for all grid-forming inverters. As shown in Fig. 6.7, the four inverters share the reactive power, while the voltage amplitude at the different inverters changes according to (2.45). It does not matter whether the subordinate component control was realized by means of expert-driven model-based optimal control methods such as MPC or by means of data-driven control methods without prior knowledge of the model parameters via RL-based control.

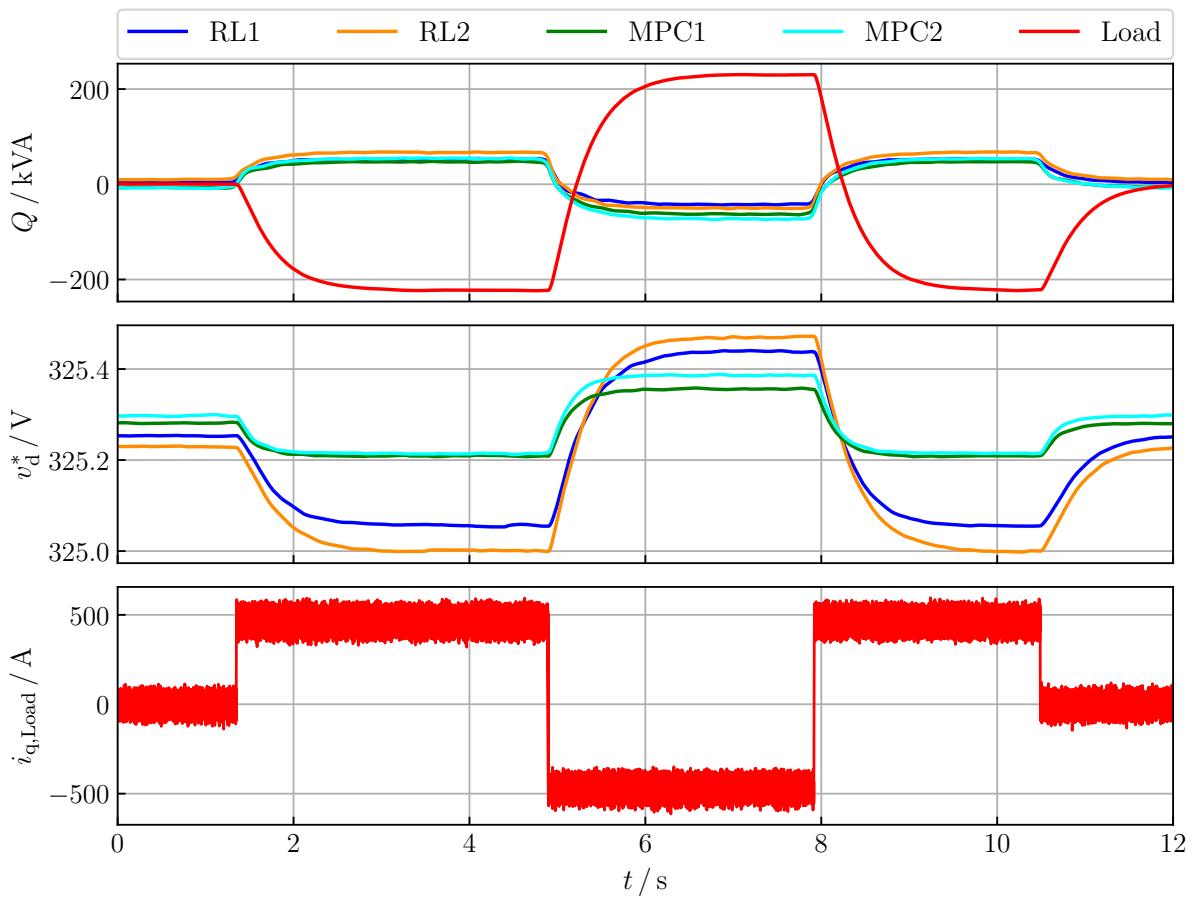


Fig. 6.7: Grid experiment for different reactive-power loading realized by the load, while $Q^* = 0$ VA for all droop controllers was chosen.

6.2.3 Non-Linear Load

To test the behavior of the controls with abnormal grid states, a current reference value with 1000 A peak is requested from the load MPC. As this exceeds the current limitation implemented within the load FCS-MPC, this results in a non-linear, trapezoidal load current curve as depicted in Fig. 6.8. As can be seen, the desired sinusoidal voltage can still be supplied by all inverters despite the non-sinusoidal load current. It can therefore be concluded that both RL-based controllers and MPCs can cope with atypical, non-sinusoidal loads in a grid application. This has already been demonstrated for a single RL-based voltage controller in Sec. 5.4.2.1.

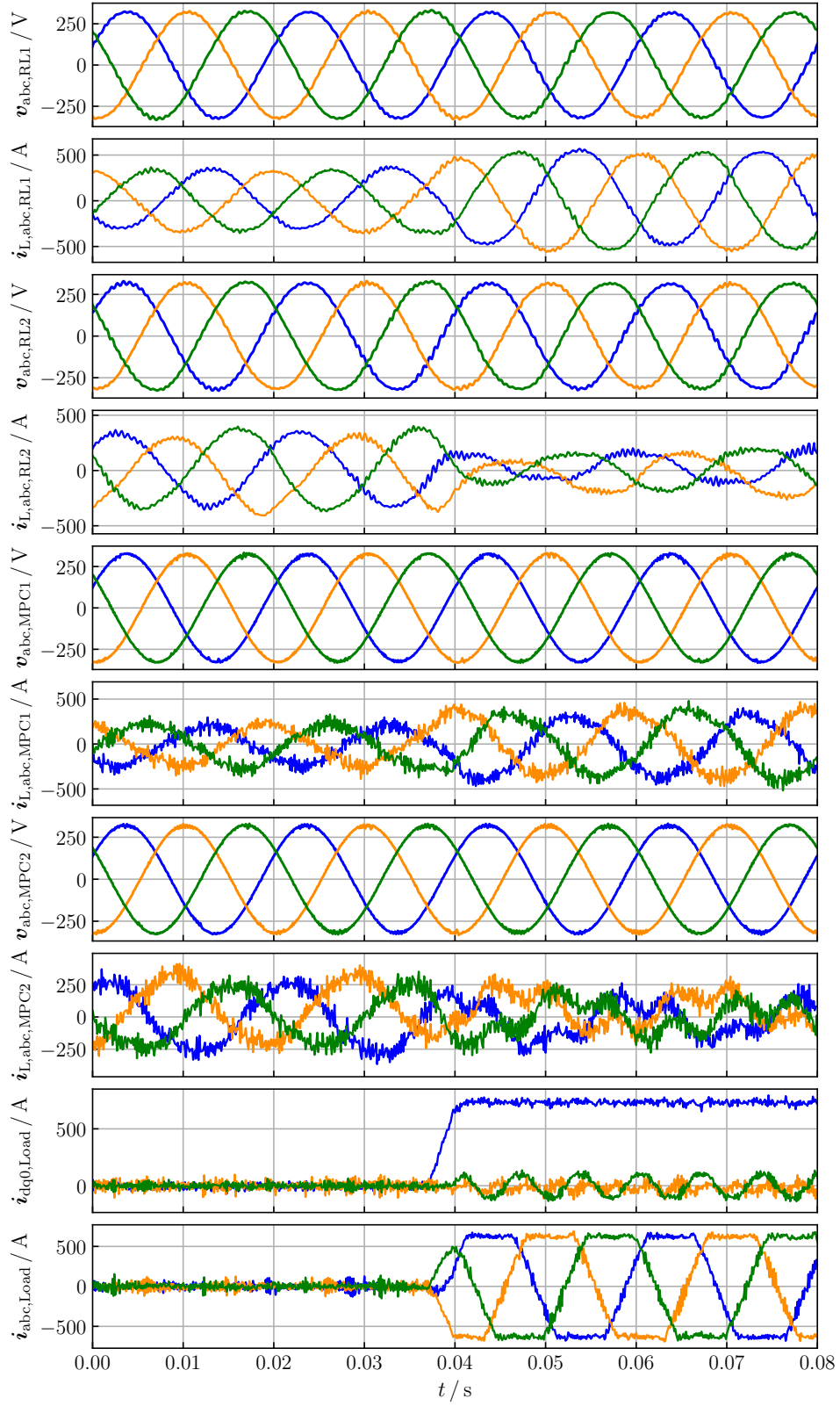


Fig. 6.8: Currents and voltages at the individual grid nodes during a trapezoidal load current.

6.2.4 Failure Event

An experiment with another MG configuration consisting of the two RL-based sources, one MPC source and the load is shown in Fig. 6.9. At time $t = 0.035$ s, the MPC source goes into an error mode, causing its inverter transistors to stop operating. As the switch is still closed, the voltage at that grid node is determined by the grid and the RL-based controllers jointly take over the load which was applied by the MPC without any noticeable change in the grid voltage. This experiment shows that it is also possible to operate a decentralized inverter-based MG in other grid configurations, even those consisting solely of RL-based controllers.

This, together with the previously shown experiments in this chapter, demonstrates the applicability of RL-based controllers in MG applications and enables the already mentioned advantages of a model-free, self-adaptive control approach, which opens up the possibility of fast, non-linear control and enables a largely automatic controller development process with minimum human effort.

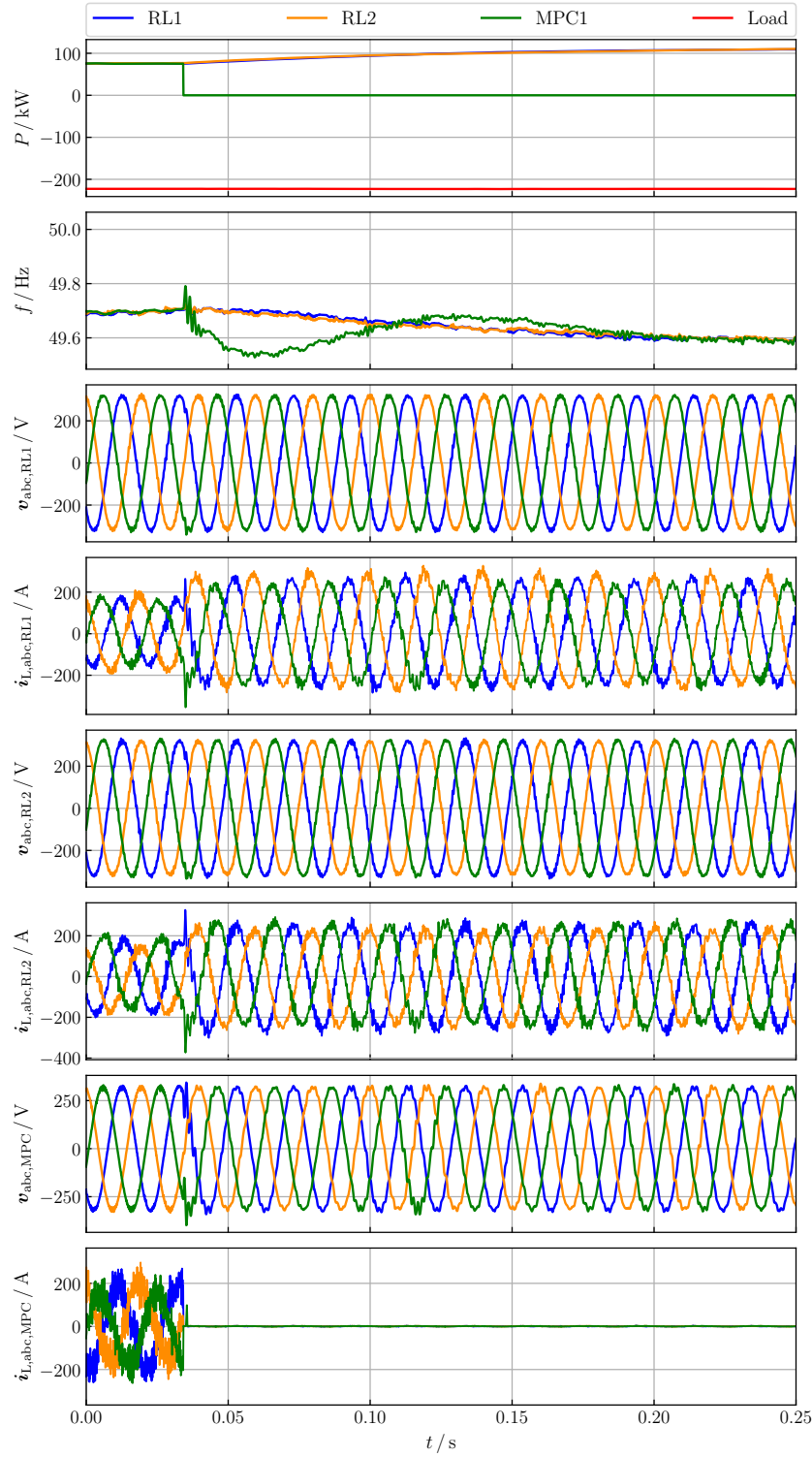


Fig. 6.9: Currents and voltages at the individual grid nodes for an experiment with three sources and one load, whereas one source stops operating at $t = 0.035$ s.

7 Conclusion and Outlook

7.1 Conclusion

The self-adaptive and model-free behavior of reinforcement learning (RL) represents a candidate solution for the challenges arising from advancing decentralization of the electric power grid. This work demonstrated that RL-based control meets the higher demands on the control engineering at the component level to benefit from the component's transient capability to allow a successful integration that optimally utilizes the inverter-based resources (IBR). In order to take advantage of the data-driven behavior of RL, the training should not be carried out in simulation, but in direct interaction with the hardware system. To make this possible, a toolchain was investigated to outsource the numerical optimization used to tune the artificial neural networks (ANNs) of the RL-based controller to an edge computer, while the artificial neural network (ANN) utilized for control was implemented on the real-time-capable rapid control prototyping (RCP) unit. Safe operation during real-world online training was ensured by extending the RL control framework by a proposed safeguard algorithm, which is based on optimal control concepts, and connects the data-driven and model-driven domains. To achieve independence of system parameter knowledge, system identification (ID) methods were used in the safeguard to ensure compliance with the system limits during training. A hardware-in-the-loop (HIL)-based hyperparameter optimization (HPO) framework was proposed to determine the best configuration and architecture for the RL algorithm and reduce the simulation-reality gap in terms of real-time capability.

The RL-based controller learned the voltage control task in less than 8 minutes and showed comparable or even better performance than several benchmark controllers. The improved performance is likely due to parasitic effects resulting from the interlock time, combined with non-linearities of the current sensors, which the RL agent can learn to compensate for. Taking these effects into account in a model of the expert-driven controller can become difficult and computationally intensive. To expand the operating range, the safeguard was extended for arbitrary loads, demonstrating the applicability of RL-based control for grid-forming inverters in microgrid (MG) applications. To evaluate the functionality in a decentralized inverter-based MG application, in a hardware experiment of an islanded MG consisting of five inverters, it was shown that the data-driven RL-based controllers

are applicable and can act as subordinate component controllers of superimposed, droop-based controllers enabling a quick dynamic response.

The results presented in this work demonstrate the applicability of RL-based controllers in MGs and enable the mentioned advantages of a self-adaptive control approach and a largely automatic controller development process with minimum human effort, which is promising for evolving plants such as the electric grid. This is the first milestone in the training of an RL-based controller on hardware in electrical power grids. Although high-performance hardware was utilized for computation, future developments in both computing hardware and artificial intelligence (AI) are expected to make such systems more affordable. Further points must be clarified before industrial integration can be considered. For use in the electric power grid or other critical infrastructure, several key requirements must be met. Most importantly, the contemporary formulation of grid codes and the necessity of analytical stability proofs does not yet account for the availability of exclusively or primarily data-driven approaches and, therefore, changes in legislation and standardization would be needed prior to wide-scale distribution of RL-based control to grid-forming applications.

7.2 Outlook

While the presented results provide the basis for the use of self-adaptive RL control methods at the component level in energy grids, enabling a fully automatic and model-free commissioning of inverter-based MGs, there are open challenges left for future research:

- In order to enable safe use in decentralized MGs in the event of externally caused faults, such as the failure of a cable or inverter components, safe handling of those failures should be subject of future investigations.
- Introducing an option for time- or temperature-variant limitations to make use of components' overloading capability could allow high currents for a limited time, which enables even faster step responses but still protects the capacitor. This would require changes and further investigations regarding the implementation of the safeguard and could be handled, e.g., via cumulative constraints (cf. [6, 24]).
- Investigating the possibilities and advantages of integrating the superimposed, currently droop-based control into the learning process of the RL control structure. Such a direct droop control concept would reduce the cascaded control structure even further, but would require further development not only of the reward function but also of the safeguard approach.
- The positive aspects of RL and MPC can be combined, e.g., by incorporating a priori knowledge and inherent constraint-related safety from MPC as well as learning from real-world feedback and fast explicit control policy inference from RL, e.g., by differential predictive control (DPC) [114, 115].

- In order to enable widespread use, it should be investigated whether the toolchain can run on low-cost control hardware.

Appendix

A.1 Deep Deterministic Policy Gradient Algorithm - Basics

In the following, the most important details about the deep deterministic policy gradient (DDPG) algorithm are described, which is introduced in [82] and learns a deterministic policy function μ in an off-policy fashion [116].

DDPG utilizes ANNs as function approximators for the policy $\hat{\mu}_\delta$ - the actor - and for the action value function $\hat{q}_\rho$ - the critic -, which enables learning a control task in continuous state and action spaces, like the application investigated. Here, δ and ρ represent the ANNs' trainable parameters. In the following, it is assumed that the Markov property (2.11) is satisfied or, if this is not the case, that this property is approximately satisfied by sufficient feature engineering according to (3.3).

To learn the optimal policy, the action value approximator $\hat{q}_\rho$ is adapted by training the parameters ρ using the experience ξ (cf. (2.16)) to minimize the mean squared Bellman error:

$$\min_{\rho} \frac{1}{|\mathcal{B}|} \sum_{\xi \in \mathcal{B}} \left(\hat{q}_\rho(\tilde{\mathbf{y}}_k, \mathbf{u}_k) - \left(r_{k+1} + \gamma(1 - t_{k+1}) \hat{q}_{\rho_{\text{target}}}(\tilde{\mathbf{y}}_{k+1}, \hat{\mu}_{\delta_{\text{target}}}(\tilde{\mathbf{y}}_{k+1})) \right) \right)^2. \quad (\text{A.1})$$

Here, $\mathcal{B}$ is a batch of experiences ξ taken from a replay buffer in which the experiences from the interaction with the environment are stored after each step. The replay buffer acts as a ring buffer of limited capacity, discarding the oldest data sample whenever an overflow occurs by adding a new sample. As can be seen in (A.1), $\hat{q}$ is used to estimate the target of the cost function that is used for its own update, which is called bootstrapping. To stabilize the learning process in that regard, DDPG makes use of target networks for actor ($\hat{\mu}_{\delta_{\text{target}}}$) and critic ($\hat{q}_{\rho_{\text{target}}}$) which are updated in a low-pass fashion, based on the trained actor and critic:

$$\begin{aligned} \rho_{\text{target}} &\leftarrow (1 - \tau)\rho_{\text{target}} + \tau\rho, \\ \delta_{\text{target}} &\leftarrow (1 - \tau)\delta_{\text{target}} + \tau\delta, \end{aligned} \quad (\text{A.2})$$

wherein $\tau \in [0, 1]$ represents a damping parameter.

With $\hat{q}_\rho$ being differentiable with respect to $\mathbf{u}$, the policy network $\hat{\mu}_\delta$ can be trained in such a way that it maximizes the return (cf. (2.15)) approximated by $\hat{q}_\rho$:

$$\max_{\delta} \frac{1}{|\mathcal{B}|} \sum_{\tilde{\mathbf{y}}_k \in \mathcal{B}} \hat{q}_{\rho}(\tilde{\mathbf{y}}_k, \hat{\mu}_{\delta}(\tilde{\mathbf{y}}_k)). \quad (\text{A.3})$$

Since a deterministic policy is learned by the DDPG algorithm, action noise has to be added during the training process to enable exploration of the state and action space. Therefore, a discrete-time Ornstein-Uhlenbeck noise [117]

$$\boldsymbol{\nu}_k = \boldsymbol{\nu}_{k-1} + \lambda \boldsymbol{\nu}_{k-1} \frac{T_s}{1s} + \sigma \sqrt{\frac{T_s}{1s}} D_N(0, 1), \quad (\text{A.4})$$

with zero mean is used, which is superimposed on the action determined by the policy (cf. (2.17)). Herein, T_s is normalized for the sake of physical unit consistency, while λ and σ define the stiffness and diffusion, respectively. $D_N(0, 1)$ denotes a realization from a standard normal distribution. Any necessary action clipping to accommodate for the limits of $\mathbb{U}$ is to be performed after this noise is added.

A.2 Hyperparameter Settings of the Used DDPG Agents

In the following, the hyperparameters (HPs) used in the various experiments are listed. In this thesis, the LeakyReLU function $y = \max(\beta_{\mu/q} x, x)$ is selected as activation function within each hidden layer. The output layer activation function of the critic is linear while the one of the actor can be chosen by the hyperparameter optimization (HPO) (cf. Sec. 3.6) to be linear or hyperbolic tangent. In case of a linear actor output, an additional clipping is defined at the actor's output and the cost function utilized during the training process is modified to increase the cost for actions much greater than one. A linear action noise scheduler is applied, reducing the variance of the action noise σ during training. In the same way, the learning rates of actor α_{μ} and critic α_q are reduced during training from step $k_{\mu/q,0}$ to $k_{\mu/q,1}$.

A.2.1 HPs for the Constant, Resistive Load Application

Tab. A.1 describes the HPs of the DDPG agent, which is trained in Sec. 4.5 to supply a constant, resistive load with the desired voltage. To determine these parameters, the HPO introduced in Sec. 3.6 was used, which was carried out with the experiment described in Sec. 4.4.

Tab. A.1: Hyperparameter optimization setting and resulting best configuration set for the RL agent used to learn a voltage control task to supply a constant, resistive load. The corresponding experiments are presented in Sec. 4.

Parameter	Search space	Best
Training steps		$2 \cdot 10^4$
Activation function parameter β_μ	$[1 \cdot 10^{-3}, 0.5]$	0.46
Activation function parameter β_q	$[1 \cdot 10^{-3}, 0.5]$	0.299
Actor layers z_μ	$\{1, \dots, 4\}$	3
Neurons hidden layer actor η_μ	$\{10, \dots, 40\}$	39
Critic layers z_q	$\{3, \dots, 10\}$	5
Neurons hidden layer critic η_q	$\{20, \dots, 300\}$	59
Learning rate actor α_μ	$[1 \cdot 10^{-5}, 0.05]$	$2.68 \cdot 10^{-3}$
Learning rate critic α_q	$[1 \cdot 10^{-5}, 0.05]$	$3.75 \cdot 10^{-3}$
α_μ reduction start $k_{\mu,0}$	$\{0, \dots, M\}$	13675
α_q reduction start $k_{q,0}$	$\{0, \dots, M\}$	13888
α_μ reduction end $k_{q,1}$	$\{k_{\mu,0}, \dots, M\}$	M
α_q reduction end $k_{q,1}$	$\{k_{\alpha,0}, \dots, M\}$	M
Final learning rate $\alpha_{\mu,\text{final}}$	$[1 \cdot 10^{-10}, \alpha_\mu]$	$1.95 \cdot 10^{-4}$
Final learning rate $\alpha_{q,\text{final}}$	$[1 \cdot 10^{-10}, \alpha_q]$	$2.77 \cdot 10^{-4}$
Discount factor γ	$[0.5, 0.999]$	0.777
Replay buffer size $ \mathcal{M} $	$\{200, \dots, 1 \cdot 10^6\}$	$7.37 \cdot 10^5$
Mini batch size $ \mathcal{B} $	$2^{\{1, \dots, 8\}}$	8
Target update τ	$[1 \cdot 10^{-4}, 0.5]$	0.234
Action noise stiffness λ	$[1, 1 \cdot 10^4]$	$7.07 \cdot 10^3$
Action noise diffusion σ	$[0.01, 5]$	3.73
Dropout Υ	$[0, \dots, 0.5]$	$7.54 \cdot 10^{-3}$
Output activation function f_μ	$[\text{linear}, \text{tanh}]$	linear

A.2.2 HPs for the Arbitrary Load Application

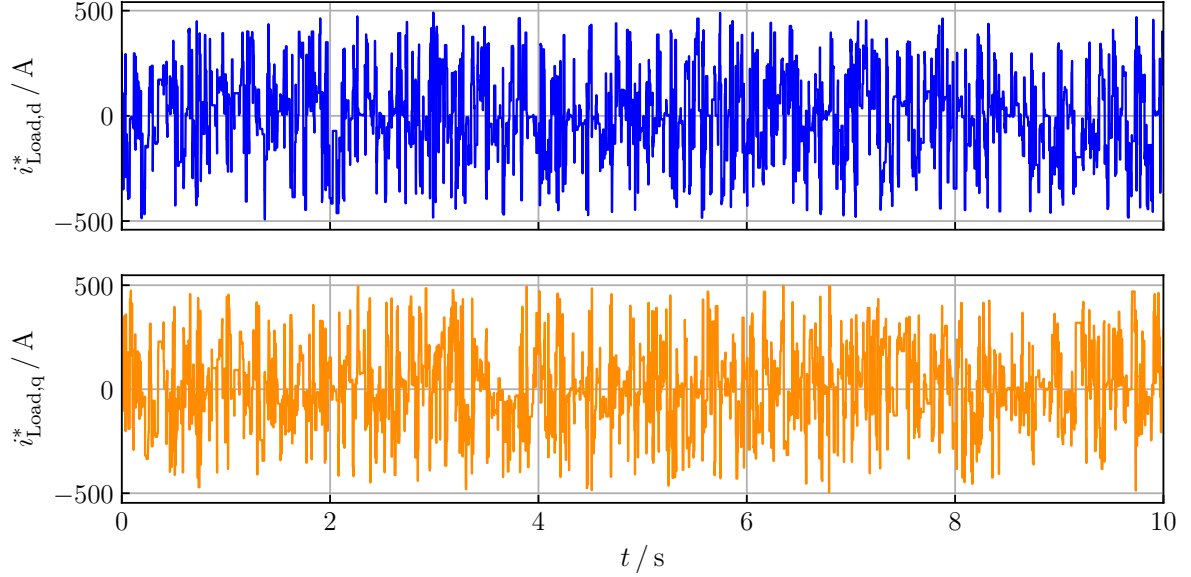


Fig. A.1: Test case, showing the reference $i_{Load,dq}^*$, which is used to evaluate the different HPs during the HPO after training ($i_0^* = 0$ A over the entire course).

To determine the HPs for the DDPG agent, which is trained to supply an arbitrary load, the HPO introduced in Sec. 3.6 was used. The experiment is described in Sec. 5 and the training process in Sec. 5.2. During the HPO, the experiments are real-time simulations on the RCP units that replicate the targeted real-world experiment. After completing the training process with one HP set the standardized test scenario depicted in Fig. A.1 is used as central performance metric, which is excluded from any training.

The learning curve of the HPO is depicted in Fig. A.2. The lines that can be identified can be explained by diverged training runs and poorly drawn HP samples. The RL agent either outputs excessively large positive or negative actions, which results in the safeguard being permanently active or the actions output by the agent approaching zero.

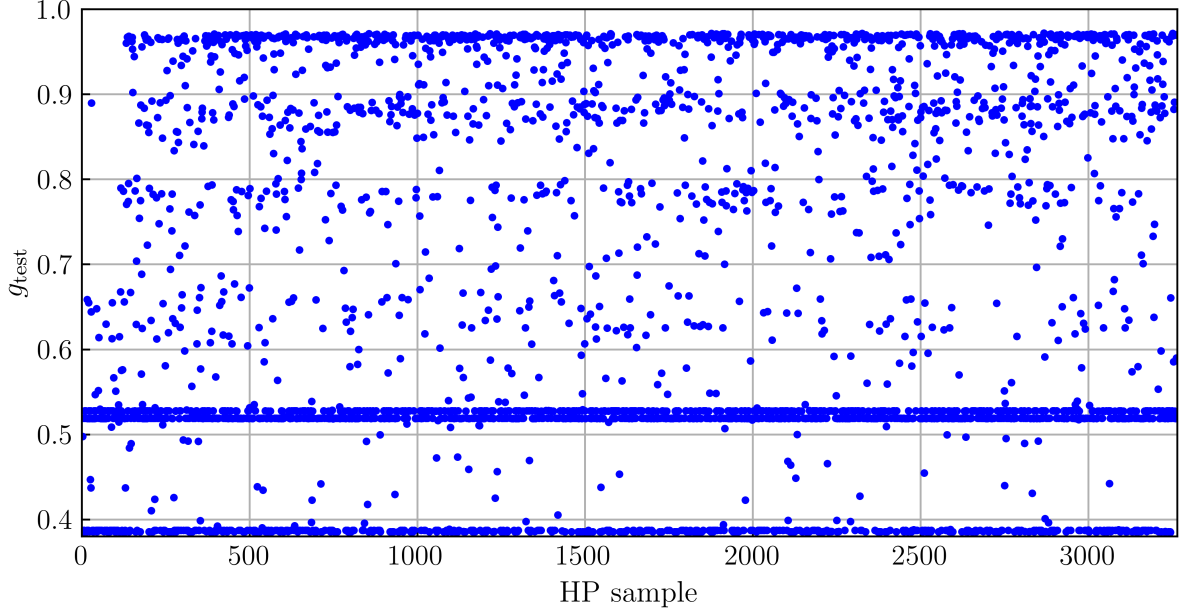


Fig. A.2: HPO learning curve showing the undiscounted, average return of the deterministic test case in Fig. A.1 performed by a trained agent for each HP sample.

Similar to the results shown in Sec. 4.4 for the constant load experiment, the HPO results and the resulting parameters are investigated. Since the experiment here is changed to an arbitrary load, the resulting parameters differ from the ones found in Sec. 4.4. The ten most important HPs during the HPO are depicted with their corresponding relative importance in Fig. A.3, which is calculated as introduced in Sec. 4.4. The ten most important HPs listed in Fig. A.3 are:

1. The output layer activation function of the actor f_{μ} ,
2. step size (learning rate) of the optimizer used for training the actor α_{μ} ,
3. learning rate of the optimizer used for training the critic α_q ,
4. Time step at which the reduction of the learning rate of the critic ends $k_{q,1}$
5. number of neurons per hidden ANN layer η_q of the critic,
6. number of layers of the critic ANN z_q ,
7. time step at which the reduction of the learning rate of the actor starts $k_{\mu,0}$
8. action noise parameter σ adjusting the exploration (cf. (A.4)),
9. the number of neurons per hidden ANN layer η_{μ} of the actor,
10. final learning rate $\alpha_{\mu,final}$ of the actor's optimizer.

The learning rates are among the most important HPs of the optimizers for training the ANNs as already seen in Sec. 4.4. With regard to all parameters listed in Tab. A.2, 76 % of the relative importance is attributable to these first ten parameters.

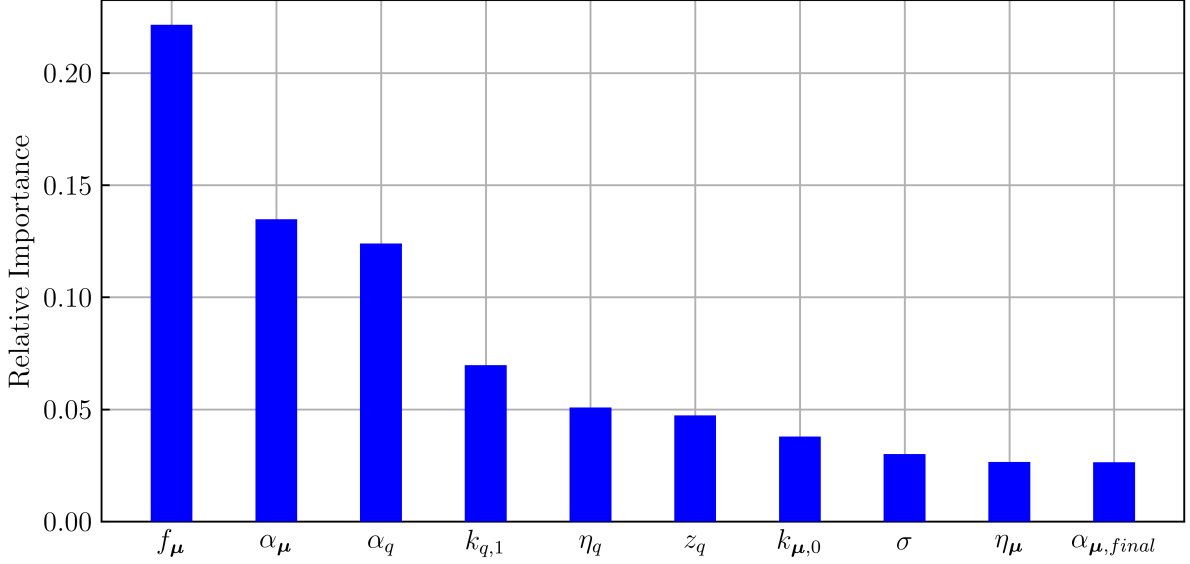


Fig. A.3: Relative importance of the ten most important HPs within the HPO calculated according to [105].

Fig. A.4 shows the four most important HPs with their respective drawn value of the HP and the performance g_{test} achieved. For the last three parameters, a clear trend can be seen as to how the parameter should be selected in order to achieve good performance. For the first parameter, this statement is not quite as clear. It can be seen that many samples were drawn with $f_\mu = \tanh$, which results in both good and poor performance. In contrast, $f_\mu = \text{linear}$ was not drawn as frequently. More HPO samples would be necessary to obtain a clearer picture, but good control performance can be achieved with the best HP set obtained from the HPO examined. Therefore, this was not investigated further. The best HP set listed in Tab. A.2 is used within the experiments presented in Sec. 5.

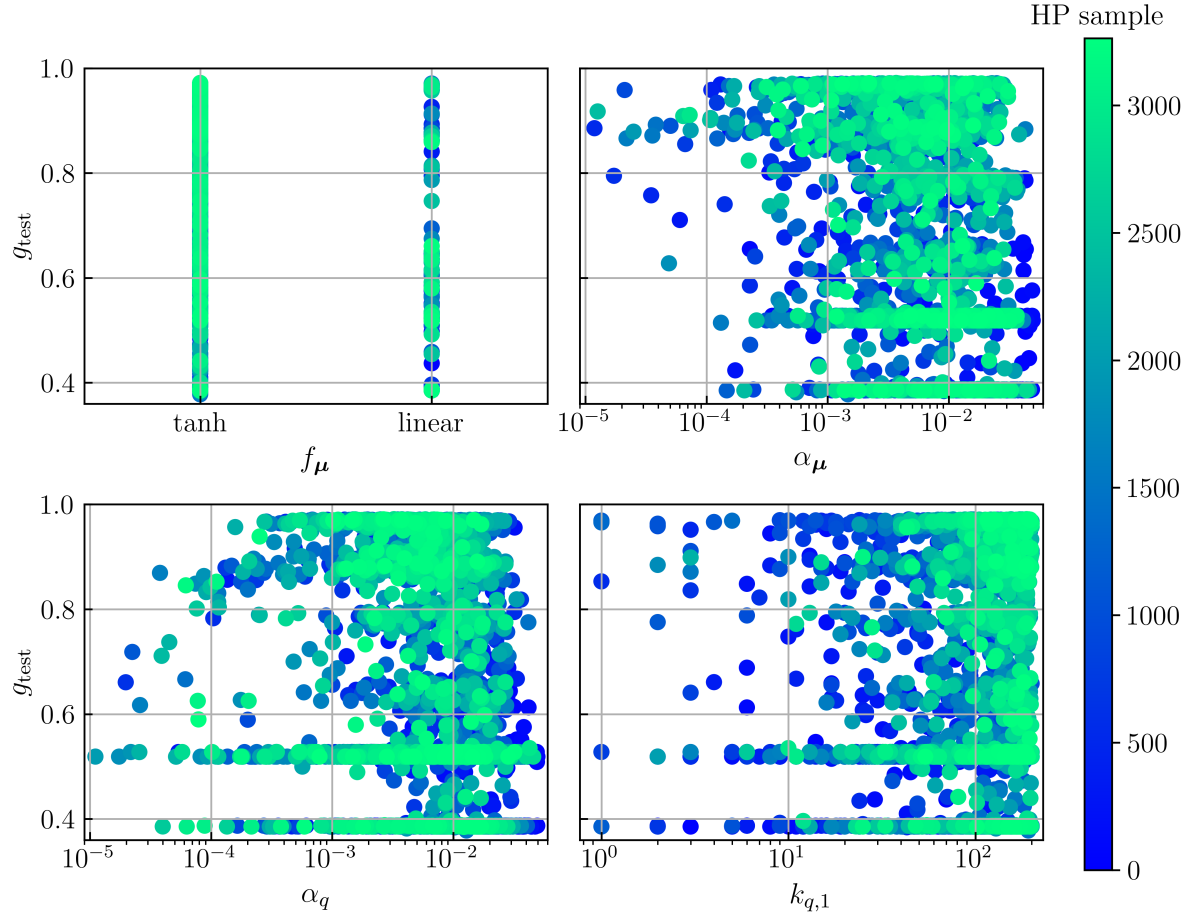


Fig. A.4: Performance g_{test} achieved with the different values of the corresponding HP during the HPO. Here, blue samples indicate early samples within the HPO, while green samples indicate more recent ones.

Tab. A.2: Hyperparameter optimization setting and resulting best configuration set for the RL agent used to learn a voltage control task to supply an arbitrary load. The corresponding experiments are presented in Sec. 5.

Parameter	Search space	Best
Training steps		$2 \cdot 10^4$
Activation function parameter β_μ	$[1 \cdot 10^{-3}, 0.5]$	0.33
Activation function parameter β_q	$[1 \cdot 10^{-3}, 0.5]$	0.28
Actor layers z_μ	$\{1, \dots, 4\}$	3
Neurons hidden layer actor η_μ	$\{10, \dots, 40\}$	22
Critic layers z_q	$\{3, \dots, 10\}$	4
Neurons hidden layer critic η_q	$\{20, \dots, 300\}$	169
Learning rate actor α_μ	$[1 \cdot 10^{-5}, 0.05]$	$2.2 \cdot 10^{-3}$
Learning rate critic α_q	$[1 \cdot 10^{-5}, 0.05]$	$2.65 \cdot 10^{-3}$
α_μ reduction start $k_{\mu,0}$	$\{0, \dots, M\}$	8763
α_q reduction start $k_{q,0}$	$\{0, \dots, M\}$	8638
α_μ reduction end $k_{q,1}$	$\{k_{\mu,0}, \dots, M\}$	14663
α_q reduction end $k_{q,1}$	$\{k_{\alpha,0}, \dots, M\}$	M
Final learning rate $\alpha_{\mu,\text{final}}$	$[1 \cdot 10^{-10}, \alpha_\mu]$	$4.41 \cdot 10^{-5}$
Final learning rate $\alpha_{q,\text{final}}$	$[1 \cdot 10^{-10}, \alpha_q]$	$2.21 \cdot 10^{-4}$
Discount factor γ	$[0.5, 0.999]$	0.869
Replay buffer size $ \mathcal{M} $	$\{200, \dots, 1 \cdot 10^6\}$	$8.04 \cdot 10^5$
Mini batch size $ \mathcal{B} $	$2^{\{1, \dots, 8\}}$	8
Target update τ	$[1 \cdot 10^{-4}, 0.5]$	0.376
Action noise stiffness λ	$[1, 1 \cdot 10^4]$	25.33
Action noise diffusion σ	$[0.01, 5]$	2.43
Dropout Υ	$[0, \dots, 0.5]$	0.072
Output activation function f_μ	$[\text{linear}, \tanh]$	$\tanh$

A.3 Predictive Dead-Beat Controller

In the following, the implementation of the predictive dead-beat (DB) controller is described, which is used as a benchmark controller within this thesis. The detailed derivation can be found in [91, 92]. The DB controller uses a model to calculate the required system input to reach a certain reference in as few steps as possible. This model is derived from (2.2), utilizing the parameters defined in Tab. 3.1. Utilizing the forward Euler discretization method according to [91, 92], while neglecting the parasitic resistances, the discrete-time model from which the control laws are derived results in:

$$\begin{bmatrix} i_{L,k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} 1 - \frac{T_s}{L_f} & \frac{T_s}{C_f} \\ \frac{T_s}{L_f} & 1 \end{bmatrix} \begin{bmatrix} i_{L,k} \\ v_k \end{bmatrix} + \begin{bmatrix} \frac{T_s}{L_f} & 0 \\ 0 & -\frac{T_s}{C_f} \end{bmatrix} \begin{bmatrix} v_{i,k} \\ i_k \end{bmatrix}. \quad (\text{A.5})$$

From (A.5), the voltage $v_{i,k+1}$ to be applied to the plant within the next time step should be calculated as a function of a voltage reference v^* and the measured quantities: $v_{i,k+1} = f(v_k, i_{L,k}, i_k, v^*)$. To compensate for the control delay introduced in Sec. 3.2, the first line of (A.5) is shifted one time step into the future, resulting in:

$$i_{L,k+2} = i_{L,k+1} + \frac{T_s}{L_f} (v_{i,k+1} - v_{k+1}). \quad (\text{A.6})$$

Using the first line from (A.5) to replace $i_{L,k+1}$, results in the control law for the current controller:

$$i_{L,k+2} = i_{L,k} + \frac{T_s}{L_f} (v_{i,k+1} + v_{i,k} - v_{k+1} - v_k). \quad (\text{A.7})$$

Assuming that the capacitor voltage does not change significantly within one step $v_{k+1} \approx v_k$ (cf. [91]), the required voltage that must be applied to reach the current at $k+2$, which is equal to $i_L^* = i_{L,k+2}$, is:

$$v_{i,k+1} = \frac{L_f}{T_s} (i_{L,k+2} - i_{L,k}) + 2v_k - v_{i,k} \quad (\text{A.8})$$

Here, $i_L^* = i_{L,k+2}$ can be interpreted as a reference value, while (A.8) calculates the required voltage to reach the i_L^* within two time steps. Since the investigated application is a voltage control task, i_L^* has to be calculated based on the voltage reference value v^* . Following the same concept like above by taking the second row of (A.5) and compensating for the control delay leads to:

$$v_{k+2} = v_k + \frac{T_s}{C_f} (i_L^* + i_{L,k} - i_{k+1} - i_k). \quad (\text{A.9})$$

Assuming the load current does not change significantly within one step $i_{k+1} \approx i_k$ (cf. [91]), the required current that must be applied to achieve the reference voltage at $v_{k+2} = v^*$ after two steps is:

$$i_L^* = \frac{C_f}{T_s} (v^* - v_k) + 2i_k - i_{L,k}. \quad (\text{A.10})$$

Utilizing (A.10) and (A.8), the required voltage to be applied to the filter input for a voltage reference v^* can be calculated as a function of the measured quantities:

$$v_{i,k+1} = \frac{L_f}{T_s} \left(\frac{C_f}{T_s} (v^* - v_k) + 2i_k - 2i_{L,k} \right) + 2v_k - v_{i,k}. \quad (\text{A.11})$$

For more details regarding the mathematical derivation, cf. [91, 92].

A.4 Artificial Intelligence use Declaration

To ensure transparency and uphold academic integrity, the following table outlines the artificial intelligence (AI) tools utilized during the preparation of a thesis.

Tab. A.3: AI use declaration

Tool	Use Case	Scope
DeepL Translate	Translation of text passages	Entire manuscript

Lists

List of Tables

1.1	Reinforcement learning-based control in electric grid applications	4
3.1	Specifications of test hardware, physical limitations and experiment setting	30
4.1	Steady-state performance – MAE and THD – of different controllers for the steady-state from the experiment shown in Fig. 4.10.	50
4.2	Transient performance of different controllers for the step response shown in Fig. 4.11. Listed are the $\int \text{MAE}_v dt / V_s$ measured over 10 ms after the step, the rise time t_{rise} and the settling time t_{settle} in a 5% tolerance band. .	52
4.3	Performance and required calculation time of different controllers on the test case shown in Fig. 4.5.	57
5.1	Average steady-state performance $\overline{\text{MAE}}$ and $\overline{\text{THD}}$ over the full operation range for different controllers. In case of RL control, three agents trained per setting, additionally the best performing agent's result $\overline{\text{MAE}}_B$ and $\overline{\text{THD}}_B$ are listed.	73
5.2	Average transient performance $\overline{\text{MAE}}$ of different controllers measured with the test case. For RL control, three agents trained per setting, additionally, $\overline{\text{MAE}}_B$ and t_{rise} of the best performing agent on the test case is listed.	75
A.1	Hyperparameter optimization setting and resulting best configuration set for the RL agent used to learn a voltage control task to supply a constant, resistive load. The corresponding experiments are presented in Sec. 4. . . .	101
A.2	Hyperparameter optimization setting and resulting best configuration set for the RL agent used to learn a voltage control task to supply an arbitrary load. The corresponding experiments are presented in Sec. 5. . .	106
A.3	AI use declaration	108

List of Figures

2.1	Three-phase VSI with <i>LC</i> -filter, while one phase is labeled for the plant components with the index $p \in \{a, b, c\}$	8
-----	--	---

LIST OF FIGURES

2.2	Reinforcement learning agent in the interaction with the environment (derivative of [76]).	10
2.3	Reinforcement learning framework extended by a safeguard component to ensure safe interaction with the environment (derivative of [76]).	14
2.4	Example of the projection of the three-dimensional feasible state-action space $\mathbb{F}_{6 \rightarrow 0}$, while considering $k_f = 6$ future time steps shown in green onto the corresponding two-dimensional feasible state space $\mathbb{Y}_{6 \rightarrow 0}$ shown in yellow. The corresponding application example is presented in Sec. 2.4.2.	16
2.5	State-action space $\mathbb{F}$ and feasible sets $\mathbb{F}_{k_f \rightarrow 0}$ for increasing k_f with corresponding volume for the double integrator described in (2.26) and (2.27).	19
2.6	Feasible set $\mathbb{F}_{1 \rightarrow 0}$ of the double integrator considering $k_f = 1$, with $[\mathbf{y}_k^\top \ u_k]^\top = [0 \ -5 \ 1]^\top$ resulting in $[\mathbf{y}_{k+1}^\top \ u_{k+1}]^\top = [-5 \ -4 \ -1]^\top$ ($u_{k+1} \in \mathbb{U}$ can be selected within the next time step and is chosen for graphical representation to be -1).	21
2.7	Droop controller used to calculate the reference voltage per inverter (derived from [94]).	24
3.1	Implemented reinforcement learning framework and structural overview of various application scenarios.	27
3.2	Experimental setup in the MG-lab, showing two 250 kVA industrial-grade inverters connected back-to-back and a resistor, which can be used as a passive load.	28
3.3	Digital control delay compensation during the safeguarding (derived from [Own1]).	31
3.4	Graphical representation of the undiscounted, single-phase reward function defined in (3.1) without taking the penalty of the safeguard intervention into account (with $p \in \{a, b, c\}$).	32
3.5	Distributed HPO framework: utilizing parallelized HIL environments for real-time hyperparameter tuning on ten RCP units with more than fifty cores in total. Only the communication needed for the HPO is shown, while the training communication (cf. Fig. 3.1) is omitted for clarity. . . .	35
4.1	Control plant and control structure of the investigated RL-based controller and the MPC, derived from [Own1].	37
4.2	Feasible set $\mathbb{F}_{\infty \rightarrow 0}$ for the voltage-forming grid inverter with LC filter and resistive load defined in (4.1) in green. A safeguard intervention is showcased for the case of $\mathbf{y}_k^\top = [180 \text{ A} \ -20 \text{ V}]$	39
4.3	Measured system excitation process in a real-world experiment with pulsating noise inputs for phase a.	41
4.4	Feasible set $\mathbb{F}_{\infty \rightarrow 0}$ based on the identified matrices in green and measurement points in blue resulting from random control outputs to evaluate the functionality of the safeguard.	42

4.5	Test case, showing the reference voltage v_d^* , which is used to evaluate the different HPs during the HPO after training ($v_{q,0}^* = 0$ V over the entire course).	43
4.6	HPO learning curve showing the undiscounted, average return of a deterministic test case performed by a trained agent for each HP sample.	44
4.7	Relative importance of the ten most important HPs within the HPO.	45
4.8	Performance g_{test} achieved with the different values of the corresponding HP during the HPO. Here, blue samples indicate early samples within the HPO, while green samples indicate more recent ones.	47
4.9	Undiscounted, averaged mean reward per learning step of the five trained agents per approach in blue. A moving average with window size 1000 is shown in red.	48
4.10	Comparison of RL's, MPC's and PI-PI steady-state behavior for different setpoints.	49
4.11	Comparison of the step responses of the different control approaches.	51
4.12	Step response, from Fig. 4.11 in abc coordinates, performed by the MPC.	53
4.13	Step response, as shown in Fig. 4.11 in abc coordinates, performed by the RL controller.	54
4.14	Proof of the black-start and emergency stop capability with corresponding comparison to MPC.	55
4.15	Comparison of the current waveforms at active constraints with $R = 2\ \Omega$ and the maximum current set to $I_{L,\text{max}} = 100$ A for RL and MPC for $v_d^* = 325$ V.	56
4.16	Ranges of the turnaround time T_{TA} for the different control algorithms determined using the test scenario depicted in Fig. 4.5. The median turnaround time $T_{\text{TA,med}}$ is highlighted in blue.	58
4.17	Extension to an unbalanced, resistive load of the plant highlighted in yellow shown in Fig. 4.1 by connecting a second resistor in parallel. The switch changes the experiment between balanced and unbalanced load.	59
4.18	Undiscounted, average reward of 3 trained agents per approach with a moving average with window size of 1 s.	60
4.19	Best RL agent per setting performing a reference step in v_d in the balanced load setup (cf. Fig. 4.17) plotted in the dq0 reference frame.	61
4.20	Best RL agent per setting performing a reference step in v_d in the unbalanced load setup (cf. Fig. 4.17) plotted in the dq0 reference frame.	62
4.21	Measured results of the best RL agent trained on an unbalanced load setting (abc_{UB}) performing the same reference step in the unbalanced load setting shown in Fig. 4.20 plotted in the abc reference frame.	63
4.22	Best RL agent per approach performing in steady state for different setpoints v_d^* in the balanced load setting.	64
4.23	Best RL agent per approach performing in steady state for different setpoints v_d^* in the unbalanced load setting.	65

LIST OF FIGURES

5.1	Control plant of the voltage controller with variable description for a single phase, while the load is emulated by a current control MPC (with $p \in \{a, b, c\}$).	67
5.2	Learning curves for different load settings, represented by the undiscounted, mean reward per learning step of the 3 agents trained per setting in blue. A moving average with window size 1000 is shown in red. .	70
5.3	Average safeguard intervention magnitude of the RL ₁₀₀₀ over the course of one training.	71
5.4	Comparison of the steady-state behavior between three model-based, expert-driven control approaches and an RL-based controller.	72
5.5	Transient control performance comparison of RL ₁₀₀₀ and baseline controllers for different consecutive load steps.	74
5.6	RL ₁₀₀₀ controller performing a load step of the d-current from 500 A to -500 A with $\Delta_k i = 1000$ A/ms.	76
5.7	RL ₁₀₀₀ controller and expert-driven control approaches performing a load step of the d-current from -500 A to 500 A with $\Delta_k i_{\text{load}} = 1000 \frac{\text{A}}{\text{ms}}$	77
5.8	Virtual load impedance (experiment setting Fig. 5.9) and virtual passive rectifier (experiment setting Fig. 5.10) approach for calculating the reference value for the load MPC based on the measured grid voltage, so that the load behaves like a passive variable load or a non-linear diode rectifier. For both scenarios, at $t = 0$ s a load step is injected: experiment setting for Fig. 5.9: the capacitor disconnects (emulates a failure) and the resistor is drastically reduced; experiment setting for Fig. 5.10: the resistive load changes.	78
5.9	RL ₁₀₀₀ controller performing a load step at $t = 0$ s caused by the impedance change depicted in Fig. 5.8.	79
5.10	RL ₁₀₀₀ controller performing a non-linear load step at $t = 0$ s caused by the load change depicted in Fig. 5.8.	80
5.11	RL ₁₀₀₀ controller performing a black start (left) and a shutdown (right) under load.	81
6.1	Control structure of the proposed model-free droop and RL-based framework.	83
6.2	Experiment MG setting with up to five 250 kVA industrial-grade inverters in the MG-lab.	84
6.3	Control plant of the voltage controllers (source) and grid configuration. The control plant (L filter) of the load controller on inverter 5 is not depicted.	84
6.4	Undiscounted, mean reward per learning step of the 3 trained agents in blue. A moving average with window size 1000 is shown in red.	85
6.5	Grid experiment for different active power loadings realized by the load, while the setpoints are configured as follows: $P_{\text{RL1}}^* = -150$ kW, $P_{\text{RL2}}^* = 150$ kW, $P_{\text{MPC1}}^* = -100$ kW and $P_{\text{MPC2}}^* = 100$ kW.	87

6.6	Measured currents and voltages at the individual grid nodes for the step at 0.855 s in Fig. 6.5.	89
6.7	Grid experiment for different reactive-power loading realized by the load, while $Q^* = 0$ VA for all droop controllers was chosen.	90
6.8	Currents and voltages at the individual grid nodes during a trapezoidal load current.	92
6.9	Currents and voltages at the individual grid nodes for an experiment with three sources and one load, whereas one source stops operating at $t = 0.035$ s.	94
A.1	Test case, showing the reference $\mathbf{i}_{\text{Load,dq}}^*$, which is used to evaluate the different HPs during the HPO after training ($i_0^* = 0$ A over the entire course).	102
A.2	HPO learning curve showing the undiscounted, average return of the deterministic test case in Fig. A.1 performed by a trained agent for each HP sample.	103
A.3	Relative importance of the ten most important HPs within the HPO calculated according to [105].	104
A.4	Performance g_{test} achieved with the different values of the corresponding HP during the HPO. Here, blue samples indicate early samples within the HPO, while green samples indicate more recent ones.	105

LIST OF FIGURES

References

- [1] CIGRÉ C6.22 Working Group (Marnay et al.) “Microgrid Evolution Road-map”. In: *International Symposium on Smart Electric Distribution Systems and Technologies (EDST15)*. 2015.
- [2] J. Rocabert, A. Luna, F. Blaabjerg, and P. Rodríguez. “Control of Power Converters in AC Microgrids”. In: *IEEE Transactions on Power Electronics* 27.11 (2012), pp. 4734–4749. DOI: 10.1109/TPEL.2012.2199334.
- [3] J. Lange, D. Schmies, K. S. Stille, J. Böcker, and O. Wallscheid. “Experimental Comparison of FPGA-Implemented Model Predictive Voltage Control to Cascaded Proportional Resonant Control for a Three-Phase Four-Wire Three-Level Grid-Forming Inverter of 250 kVA”. In: *European Conference on Power Electronics and Applications (EPE ECCE Europe)*. 2022.
- [4] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. 2nd ed. Nob Hill Publishing, LLC, 2017.
- [5] M. S. Massaoudi, H. Abu-Rub, and A. Ghrayeb. “Navigating the Landscape of Deep Reinforcement Learning for Power System Stability Control: A Review”. In: *IEEE Access* 11 (2023), pp. 134298–134317. DOI: 10.1109/ACCESS.2023.3337118.
- [6] T. Su, T. Wu, J. Zhao, A. Scaglione, and L. Xie. “A Review of Safe Reinforcement Learning Methods for Modern Power Systems”. In: *Proceedings of the IEEE* 113.3 (2025), pp. 213–255. DOI: 10.1109/JPROC.2025.3584656.
- [7] D. Görges. “Relations between Model Predictive Control and Reinforcement Learning”. In: *IFAC-PapersOnLine* 50.1 (2017). 20th IFAC World Congress, pp. 4920–4928. DOI: <https://doi.org/10.1016/j.ifacol.2017.08.747>.
- [8] B. She, F. Li, H. Cui, J. Zhang, and R. Bo. “Fusion of microgrid control with model-free reinforcement learning: Review and vision”. In: *IEEE Transactions on Smart Grid* 14.4 (2023), pp. 3232–3245.
- [9] R. Reiter, J. Hoffmann, D. Reinhardt, et al. *Synthesis of Model Predictive Control and Reinforcement Learning: Survey and Classification*. 2025. arXiv: 2502.02133 [eess.SY]. URL: <https://arxiv.org/abs/2502.02133>.
- [10] S. M. Miraftabzadeh, A. D. Martino, M. Longo, and D. Zaninelli. “Deep Learning in Power Systems: A Bibliometric Analysis and Future Trends”. In: *IEEE Access* 12 (2024), pp. 163172–163196. DOI: 10.1109/ACCESS.2024.3491914.
- [11] L. Brunke, M. Greeff, A. W. Hall, et al. “Safe Learning in Robotics: From Learning-Based Control to Safe Reinforcement Learning”. In: *Annual Review of*

- Control, Robotics, and Autonomous Systems* 5. Volume 5, 2022 (2022), pp. 411–444. DOI: <https://doi.org/10.1146/annurev-control-042920-020211>.
- [12] V.-H. Bui, S. Mohammadi, S. Das, A. Hussain, G. V. Hollweg, and W. Su. “A critical review of safe reinforcement learning strategies in power and energy systems”. In: *Engineering Applications of Artificial Intelligence* 143 (2025), p. 110091. DOI: <https://doi.org/10.1016/j.engappai.2025.110091>.
- [13] S. Gao, Y. Xu, Z. Zhang, Z. Wang, X. Zhou, and J. Wang. “Multi-Agent Imitation Learning Based Energy Management of a Microgrid With Hybrid Energy Storage and Real-Time Pricing”. In: *IEEE Internet of Things Journal* (2025), pp. 1–1. DOI: 10.1109/JIOT.2025.3541350.
- [14] Y. Li, J. Gao, Y. Li, et al. “Physical Informed-Inspired Deep Reinforcement Learning Based Bi-Level Programming for Microgrid Scheduling”. In: *IEEE Transactions on Industry Applications* 61.1 (2025), pp. 1488–1500. DOI: 10.1109/TIA.2024.3522486.
- [15] W.-G. Lee and H.-M. Kim. “Deep Reinforcement Learning-Based Dynamic Droop Control Strategy for Real-Time Optimal Operation and Frequency Regulation”. In: *IEEE Transactions on Sustainable Energy* 16.1 (2025), pp. 284–294. DOI: 10.1109/TSTE.2024.3454298.
- [16] Q. Liu, Y. Guo, L. Deng, H. Liu, D. Li, and H. Sun. “Residual Deep Reinforcement Learning With Model-Based Optimization for Inverter-Based Volt-Var Control”. In: *IEEE Transactions on Sustainable Energy* 16.1 (2025), pp. 269–283. DOI: 10.1109/TSTE.2024.3454080.
- [17] M. B. Frasetyo, F. D. Wijaya, H. R. Ali, and M. Zharfan Wiranata. “Adaptive Synchronverter Algorithm Using Q-learning for Microgrid Stability Control”. In: *IEEE Access* (2025), pp. 1–1. DOI: 10.1109/ACCESS.2025.3551287.
- [18] A. Ahmadian, K. Sedghisigarchi, and R. Gadh. “Empowering Dynamic Active and Reactive Power Control: A Deep Reinforcement Learning Controller for Three-Phase Grid-Connected Electric Vehicles”. In: *IEEE Access* 12 (2024), pp. 66068–66084. DOI: 10.1109/ACCESS.2024.3396449.
- [19] T. Liu, Y. Song, L. Zhu, and D. J. Hill. “Stability and Control of Power Grids”. In: *Annual Review of Control, Robotics, and Autonomous Systems* 5. Volume 5, 2022 (2022), pp. 689–716. DOI: <https://doi.org/10.1146/annurev-control-042820-011148>.
- [20] Y. Wan, Q. Xu, and T. Dragičević. “Reinforcement Learning-Based Predictive Control for Power Electronic Converters”. In: *IEEE Transactions on Industrial Electronics* (2024), pp. 1–12. DOI: 10.1109/TIE.2024.3472299.
- [21] H. Krasowski, J. Thumm, M. Müller, L. Schäfer, X. Wang, and M. Althoff. “Provably Safe Reinforcement Learning: Conceptual Analysis, Survey, and Benchmarking”. In: *Transactions on Machine Learning Research* (2023). Survey Certification. URL: <https://openreview.net/forum?id=mcN0ezbnzO>.
- [22] M. Schenke and O. Wallscheid. “A Deep Q-Learning Direct Torque Controller for Permanent Magnet Synchronous Motors”. In: *IEEE Open Journal of the Industrial Electronics Society* 2 (2021), pp. 388–400.

-
- [23] J. Zhao, F. Li, S. Mukherjee, and C. Sticht. “Deep Reinforcement Learning-Based Model-Free On-Line Dynamic Multi-Microgrid Formation to Enhance Resilience”. In: *IEEE Transactions on Smart Grid* 13.4 (2022), pp. 2557–2567. DOI: 10.1109/TSG.2022.3160387.
 - [24] Q. Zhang, K. Dehghanpour, Z. Wang, F. Qiu, and D. Zhao. “Multi-Agent Safe Policy Learning for Power Management of Networked Microgrids”. In: *IEEE Transactions on Smart Grid* 12.2 (2021), pp. 1048–1062. DOI: 10.1109/TSG.2020.3034827.
 - [25] Z. Yan and Y. Xu. “Real-Time Optimal Power Flow: A Lagrangian Based Deep Reinforcement Learning Approach”. In: *IEEE Transactions on Power Systems* 35.4 (2020), pp. 3270–3273. DOI: 10.1109/TPWRS.2020.2987292.
 - [26] T. Wu, A. Scaglione, and D. Arnold. “Constrained Reinforcement Learning for Predictive Control in Real-Time Stochastic Dynamic Optimal Power Flow”. In: *IEEE Transactions on Power Systems* 39.3 (2024), pp. 5077–5090. DOI: 10.1109/TPWRS.2023.3326121.
 - [27] G. Hao, Y. Li, Y. Li, L. Jiang, and Z. Zeng. “Lyapunov-Based Safe Reinforcement Learning for Microgrid Energy Management”. In: *IEEE Transactions on Neural Networks and Learning Systems* 36.6 (2025), pp. 9985–9999. DOI: 10.1109/TNNLS.2024.3496932.
 - [28] Y. Ye, H. Wang, P. Chen, Y. Tang, and G. Strbac. “Safe Deep Reinforcement Learning for Microgrid Energy Management in Distribution Networks With Leveraged Spatial–Temporal Perception”. In: *IEEE Transactions on Smart Grid* 14.5 (2023), pp. 3759–3775. DOI: 10.1109/TSG.2023.3243170.
 - [29] J. Zhang, Y. Guan, L. Che, and M. Shahidehpour. “EV Charging Command Fast Allocation Approach Based on Deep Reinforcement Learning With Safety Modules”. In: *IEEE Transactions on Smart Grid* 15.1 (2024), pp. 757–769. DOI: 10.1109/TSG.2023.3281782.
 - [30] G. Ceusters, L. R. Camargo, R. Franke, A. Nowé, and M. Messaie. “Safe reinforcement learning for multi-energy management systems with known constraint functions”. In: *Energy and AI* 12 (2023), p. 100227. DOI: <https://doi.org/10.1016/j.egyai.2022.100227>.
 - [31] X. Chen, G. Qu, Y. Tang, S. Low, and N. Li. “Reinforcement Learning for Selective Key Applications in Power Systems: Recent Advances and Future Challenges”. In: *IEEE Transactions on Smart Grid* 13.4 (2022), pp. 2935–2958. DOI: 10.1109/TSG.2022.3154718.
 - [32] M. Zhang, G. Guo, S. Magnússon, R. C. N. Pilawa-Podgurski, and Q. Xu. “Data Driven Decentralized Control of Inverter Based Renewable Energy Sources Using Safe Guaranteed Multi-Agent Deep Reinforcement Learning”. In: *IEEE Transactions on Sustainable Energy* 15.2 (2024), pp. 1288–1299. DOI: 10.1109/TSTE.2023.3341632.
 - [33] P. Yu, H. Zhang, Y. Song, H. Hui, and G. Chen. “District Cooling System Control for Providing Operating Reserve Based on Safe Deep Reinforcement Learning”. In: *IEEE Transactions on Power Systems* 39.1 (2024), pp. 40–52. DOI: 10.1109/TPWRS.2023.3237888.

- [34] N. Kochdumper, H. Krasowski, X. Wang, S. Bak, and M. Althoff. “Provably Safe Reinforcement Learning via Action Projection Using Reachability Analysis and Polynomial Zonotopes”. In: *IEEE Open Journal of Control Systems* 2 (2023), pp. 79–92. DOI: 10.1109/OJCSYS.2023.3256305.
- [35] V.-H. Bui, A. Hussain, and W. Su. “A Dynamic Internal Trading Price Strategy for Networked Microgrids: A Deep Reinforcement Learning-Based Game-Theoretic Approach”. In: *IEEE Transactions on Smart Grid* 13.5 (2022), pp. 3408–3421. DOI: 10.1109/TSG.2022.3168856.
- [36] Z. Yi, X. Wang, C. Yang, C. Yang, M. Niu, and W. Yin. “Real-Time Sequential Security-Constrained Optimal Power Flow: A Hybrid Knowledge-Data-Driven Reinforcement Learning Approach”. In: *IEEE Transactions on Power Systems* 39.1 (2024), pp. 1664–1680. DOI: 10.1109/TPWRS.2023.3262843.
- [37] E. Balouji, Ö. Salor, and S. Al Khatib. “Deep Reinforcement Learning Enabled Inverters: Strengthening RES Integration in Grids With Electric Arc Furnaces”. In: *IEEE Transactions on Industry Applications* 61.1 (2025), pp. 1318–1325. DOI: 10.1109/TIA.2024.3462918.
- [38] H. Liu, M. Peng, L. Feng, X. Diao, Y. Li, and F. Liu. “Comprehensive Performance-Oriented Control for Grid-forming Converters”. In: *2024 IEEE 19th Conference on Industrial Electronics and Applications (ICIEA)*. 2024, pp. 1–5. DOI: 10.1109/ICIEA61579.2024.10664813.
- [39] Z. Wu, M. Zhang, B. Fan, Y. Shi, and X. Guan. “Deep Synchronization Control of Grid-Forming Converters: A Reinforcement Learning Approach”. In: *IEEE/CAA Journal of Automatica Sinica* 12.1 (2025), pp. 273–275. DOI: 10.1109/JAS.2024.124824.
- [40] O. Stanojev, O. Kundacina, U. Markovic, E. Vrettos, P. Aristidou, and G. Hug. “A Reinforcement Learning Approach for Fast Frequency Control in Low-Inertia Power Systems”. In: *2020 52nd North American Power Symposium (NAPS)*. 2021, pp. 1–6. DOI: 10.1109/NAPS50074.2021.9449821.
- [41] J. Gao, S. Chen, X. Li, and J. Zhang. “Transient Voltage Control Based on Physics-Informed Reinforcement Learning”. In: *IEEE Journal of Radio Frequency Identification* 6 (2022), pp. 905–910. DOI: 10.1109/JRFID.2022.3213895.
- [42] S. Jiang, Y. Zeng, Y. Zhu, J. Pou, and G. Konstantinou. “Stability-Oriented Multiobjective Control Design for Power Converters Assisted by Deep Reinforcement Learning”. In: *IEEE Transactions on Power Electronics* 38.10 (2023), pp. 12394–12400. DOI: 10.1109/TPEL.2023.3299979.
- [43] M. Eskandari, A. V. Savkin, and J. Fletcher. “A Deep Reinforcement Learning-Based Intelligent Grid-Forming Inverter for Inertia Synthesis by Impedance Emulation”. In: *IEEE Transactions on Power Systems* 38.3 (2023), pp. 2978–2981. DOI: 10.1109/TPWRS.2023.3242469.
- [44] R. Yan, Q. Xing, and Y. Xu. “Multi-Agent Safe Graph Reinforcement Learning for PV Inverters-Based Real-Time Decentralized Volt/Var Control in Zoned Distribution Networks”. In: *IEEE Transactions on Smart Grid* 15.1 (2024), pp. 299–311. DOI: 10.1109/TSG.2023.3277087.

-
- [45] H. Shuai, B. She, J. Wang, and F. Li. “Safe Reinforcement Learning for Grid-forming Inverter Based Frequency Regulation with Stability Guarantee”. In: *Journal of Modern Power Systems and Clean Energy* 13.1 (2025), pp. 79–86. DOI: 10.35833/MPCE.2023.000882.
 - [46] Y. Shi, X. Gu, X. Yin, S. Feng, and S. Zhang. “Design of droop controller in islanded microgrids using multi-objective optimisation based on accurate small-signal model”. In: *IET Power Electronics* 15.11 (2022), pp. 1093–1109. DOI: <https://doi.org/10.1049/pel2.12293>. eprint: <https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/pel2.12293>.
 - [47] W. Wang, N. Yu, Y. Gao, and J. Shi. “Safe Off-Policy Deep Reinforcement Learning Algorithm for Volt-VAR Control in Power Distribution Systems”. In: *IEEE Transactions on Smart Grid* 11 (2020), pp. 3008–3018. DOI: 10.1109/tsg.2019.2962625.
 - [48] S. Bouchkati, R. Sabirov, S. Kortmann, and A. Ulbig. “Partially Observable Residual Reinforcement Learning for PV-Inverter-Based Voltage Control in Distribution Grids”. In: *2025 IEEE Kiel PowerTech*. 2025, pp. 1–7. DOI: 10.1109/PowerTech59965.2025.11180598.
 - [49] H. Alduaij and Y. Weng. “Reinforcement Learning for Stability-Guaranteed Adaptive Optimal Primary Frequency Control of Power Systems Using Partially Monotonic Neural Networks”. In: *IEEE Open Access Journal of Power and Energy* 12 (2025), pp. 259–269. DOI: 10.1109/OAJPE.2025.3556142.
 - [50] D. Tabas and B. Zhang. “Computationally Efficient Safe Reinforcement Learning for Power Systems”. In: *2022 American Control Conference (ACC)* (2022), pp. 3303–3310. URL: <https://api.semanticscholar.org/CorpusID:239049670>.
 - [51] P. Chen, S. Liu, X. Wang, and I. Kamwa. “Physics-Shielded Multi-Agent Deep Reinforcement Learning for Safe Active Voltage Control With Photovoltaic/Battery Energy Storage Systems”. In: *IEEE Transactions on Smart Grid* 14.4 (2023), pp. 2656–2667. DOI: 10.1109/TSG.2022.3228636.
 - [52] Y. Zhang, P. Wang, L. Yu, N. Li, and J. Qiu. “Decoupled Volt/Var Control With Safe Reinforcement Learning Based on Approximate Bayesian Inference”. In: *IEEE Transactions on Sustainable Energy* 16.2 (2025), pp. 797–811. DOI: 10.1109/TSTE.2024.3485060.
 - [53] M. Zhang, Y. Xu, L. Sang, Y. Guo, and N. D. Hatziargyriou. “Virtual Power Plants for Frequency Regulation: A Learning-based Method with Safety Guarantee”. In: *IEEE Transactions on Smart Grid* (2025), pp. 1–1. DOI: 10.1109/TSG.2025.3618896.
 - [54] A. Oshnoei, H. Sorouri, S. Oshnoei, R. Teodorescu, and F. Blaabjerg. “Grid Impedance Shaping for Grid-Forming Inverters: A Soft Actor-Critic Deep Reinforcement Learning Algorithm”. In: *2024 IEEE 10th International Power Electronics and Motion Control Conference (IPEMC2024-ECCE Asia)*. 2024, pp. 4935–4939. DOI: 10.1109/IPEMC-ECCEAsia60879.2024.10567367.
 - [55] B. She, F. Li, H. Cui, et al. “Inverter PQ Control With Trajectory Tracking Capability for Microgrids Based on Physics-Informed Reinforcement Learning”.

- In: *IEEE Transactions on Smart Grid* 15.1 (2024), pp. 99–112. DOI: 10.1109/TSG.2023.3277330.
- [56] C. Yu, F. Xu, H. Xu, et al. “Reinforcement Learning-based Control Study of Three-phase LCL-type Photovoltaic Grid-connected Inverter”. In: *IEEE China International Youth Conference On Electrical Engineering (CIYCEE)*. 2023. DOI: 10.1109/CIYCEE59789.2023.10401521.
- [57] A. Rajamallaiiah, S. Phani Krishna Karri, M. L. Alghaythi, and M. S. Alshammari. “Direct Current Control of Grid Connected Two Level Inverter With LCL-Filter Using Deep Reinforcement Learning Algorithm”. In: *IEEE Access* 12 (2024), pp. 119840–119855. DOI: 10.1109/ACCESS.2024.3450793.
- [58] A. Rajamallaiiah, S. P. K. Karri, M. L. Alghaythi, and M. S. Alshammari. “Deep Reinforcement Learning Based Control of a Grid Connected Inverter With LCL-Filter for Renewable Solar Applications”. In: *IEEE Access* 12 (2024), pp. 22278–22295. DOI: 10.1109/ACCESS.2024.3364058.
- [59] O. Menéndez, D. López-Caiza, L. Tarisciotti, F. Ruiz, F. Auat-Cheein, and J. Rodríguez. “Assessment of Deep Reinforcement Learning Algorithms for Three-Phase Inverter Control”. In: *2023 IEEE 8th Southern Power Electronics Conference and 17th Brazilian Power Electronics Conference (SPEC/COBEP)*. 2023, pp. 1–8. DOI: 10.1109/SPEC56436.2023.10407331.
- [60] A. Rajamallaiiah, S. P. K. Karri, and Y. R. Shankar. “Deep Reinforcement Learning Based Control Strategy for Voltage Regulation of DC-DC Buck Converter Feeding CPLs in DC Microgrid”. In: *IEEE Access* 12 (2024), pp. 17419–17430. DOI: 10.1109/ACCESS.2024.3358412.
- [61] T. Ji, P. Lin, M. Zhu, Q. Meng, C. Zhang, and J. Zhu. “Deep Reinforcement Learning Based Robust Control Strategy for Power Sharing and Voltage Regulation in DC Grid-Forming Energy Storage System”. In: *2024 4th Power System and Green Energy Conference (PSGEC)*. 2024, pp. 1327–1331. DOI: 10.1109/PSGEC62376.2024.10721061.
- [62] M. Nicola and C.-I. Nicola. “Improved Performance for the DC-AC Converters Control System Based on Robust Controller and Reinforcement Learning Agent”. In: *2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*. 2022, pp. 1–6. DOI: 10.1109/ICECCME55909.2022.9988458.
- [63] X. Xu, L. Ye, J. Qin, and H. Jiang. “Behavior Cloning based Deep Reinforcement Learning for Control of Grid-Connected Photovoltaic Inverter”. In: *2025 IEEE 14th Data Driven Control and Learning Systems (DDCLS)*. 2025, pp. 422–427. DOI: 10.1109/DDCLS66240.2025.11065961.
- [64] S. M. R. Rostami and Z. Al-Shibaany. “Reinforcement Learning-Based Voltage Regulation of DC Boost Converter in a PEM Fuel Cell Electric Vehicle”. In: *2025 17th International Conference on Advanced Computational Intelligence (ICACI)*. 2025, pp. 30–36. DOI: 10.1109/ICACI65340.2025.11096338.
- [65] S. C. Das, T. Vu, D. Ramasubramanian, E. Farantatos, J. Zhang, and T. Ortmeier. “Deep Reinforcement Learning for Optimizing Inverter Control With Fixed and Adaptive Gain Tuning Strategies for Power System Stability”. In:

-
- IEEE Transactions on Smart Grid* 16.6 (2025), pp. 5099–5110. DOI: 10.1109/TSG.2025.3604554.
- [66] C. Gong and C.-S. Lam. “Model and Data Hybrid Reinforcement Learning for Optimal Voltage-Current Control of Hybrid Active Power Filter”. In: *IEEE Transactions on Power Electronics* 40.9 (2025), pp. 12845–12857. DOI: 10.1109/TPEL.2025.3569230.
 - [67] Y. Zhang, P. Li, X. Zhang, et al. “Deep Reinforcement Learning and Deadbeat Hybrid Control Method for Hybrid Energy Storage System Considering Non-linear Power Loss and Model Mismatch”. In: *IEEE Transactions on Industrial Informatics* 21.4 (2025), pp. 3296–3305. DOI: 10.1109/TII.2024.3523585.
 - [68] A. Brosch, S. Hanke, O. Wallscheid, and J. Böcker. “Data-Driven Recursive Least Squares Estimation for Model Predictive Current Control of Permanent Magnet Synchronous Motors”. In: *IEEE Transactions on Power Electronics* 36.2 (2021), pp. 2179–2190. DOI: 10.1109/TPEL.2020.3006779.
 - [69] G. Meinsma and A. van der Schaft. *A Course on Optimal Control*. Springer, 2023.
 - [70] R. Teichmann and S. Bernet. “A comparison of three-level converters versus two-level converters for low-voltage drives, traction, and utility applications”. In: *IEEE Transactions on Industry Applications* 41.3 (2005), pp. 855–865. DOI: 10.1109/TIA.2005.847285.
 - [71] J.-S. Lai and F. Z. Peng. “Multilevel converters-a new breed of power converters”. In: *IEEE Transactions on Industry Applications* 32.3 (1996), pp. 509–517. DOI: 10.1109/28.502161.
 - [72] Y. Wang, A. Poorfakhraei, N. Mehdi, and A. Emadi. “Comparative Analysis of 2-Level and 3-Level Voltage Source Inverters in Traction Applications”. In: *2021 IEEE Transportation Electrification Conference & Expo (ITEC)*. 2021, pp. 614–619. DOI: 10.1109/ITEC51675.2021.9490160.
 - [73] J. Böcker. *Paderborn University, lecture notes: Power Electronics*. https://ei.uni-paderborn.de/fileadmin-eim/elektrotechnik/fg/lea/Lehre/LE/Dokumente/Skript_LE_2020-03-24.pdf. 2020.
 - [74] G. Carrara, S. Gardella, M. Marchesoni, R. Salutari, and G. Sciutto. “A new multilevel PWM method: a theoretical analysis”. In: *IEEE Transactions on Power Electronics* 7.3 (1992), pp. 497–505. DOI: 10.1109/63.145137.
 - [75] P. Antsaklis and A. Michel. *A Linear Systems Primer*. Jan. 2007. DOI: 10.1007/978-0-8176-4661-5.
 - [76] Megajuce. *Reinforcement Learning Diagram*. Creative Commons CC0 1.0 Universal Public Domain Dedication. 2025. URL: https://commons.wikimedia.org/wiki/File:Reinforcement_learning_diagram.svg (visited on 11/10/2025).
 - [77] J. Heinrich and D. Silver. *Deep Reinforcement Learning from Self-Play in Imperfect-Information Games*. 2016. arXiv: 1603.01121 [cs.LG]. URL: <https://arxiv.org/abs/1603.01121>.
 - [78] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018.

- [79] D. Bertsekas. *Dynamic Programming and Optimal Control*. 3rd ed. Vol. 1. Athena Scientific, 2005.
- [80] Y. Jafarian, O. Salari, M. Z. Youssef, A. Bakhshai, and P. Jain. “A Novel Reconstructed Voltage Predictive Control for the Multilevel T-NPC Converter: Design, Simulation, and Experimentation”. In: *IEEE Journal of Emerging and Selected Topics in Industrial Electronics* (2024), pp. 1–14. DOI: 10.1109/JESTIE.2024.3521390.
- [81] O. Wallscheid, W. Kirchgässner, M. Schenke, et al. *UPB LEA Reinforcement Learning Course Materials*. Version 2023_v1. 2023. DOI: 10.5281/zenodo.8159062.
- [82] T. P. Lillicrap, J. J. Hunt, A. Pritzel, et al. *Continuous control with deep reinforcement learning*. 2015. arXiv: 1509.02971 [cs.LG].
- [83] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [84] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.
- [85] F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017. DOI: 10.1017/9781139061759.
- [86] P. Peters. “Combining Model-Based and Model-Free Optimal Control for Dynamic Systems”. Master thesis. Paderborn University, 2022.
- [87] C. N. Jones, E. C. Kerrigan, and J. M. Maciejowski. “Equality set projection: A new algorithm for the projection of polytopes in halfspace representation”. In: *Technical Report CUED Technical Report CUED/F-INFENG/TR.463, Department of Engineering, Cambridge University* (2004).
- [88] M. Herceg, M. Kvasnica, C. Jones, and M. Morari. “Multi-Parametric Toolbox 3.0”. In: *Proc. of the European Control Conference*. <http://control.ee.ethz.ch/~mpt>. Zürich, Switzerland, July 2013, pp. 502–510.
- [89] D. Ching, I. Filippidis, S. C. Livingston, et al. *Polytope 0.2.5*. 2024. URL: <https://github.com/tulip-control/polytope>.
- [90] J. Umland and M. Safiuddin. “Magnitude and symmetric optimum criterion for the design of linear control systems: what is it and how does it compare with the others?” In: *IEEE Transactions on Industry Applications* 26.3 (1990), pp. 489–497. DOI: 10.1109/28.55967.
- [91] S. Buso and P. Mattavelli. *Digital Control in Power Electronics*. Morgan & Claypool, 2006.
- [92] M. Pichan, H. Rastegar, and M. Monfared. “Deadbeat Control of the Stand-Alone Four-Leg Inverter Considering the Effect of the Neutral Line Inductor”. In: *IEEE Transactions on Industrial Electronics* 64.4 (2017), pp. 2592–2601. DOI: 10.1109/TIE.2016.2631459.
- [93] R. Isermann and M. Münchhof. *Identification of Dynamic Systems: An Introduction with Applications*. Advanced Textbooks in Control and Signal Processing Series. Springer Berlin Heidelberg, 2010. URL: <https://books.google.de/books?id=tTwYA6ph9L0C>.

-
- [94] D. Schmies, K. S. Stille, J. Lange, and O. Wallscheid. “Operational Insights into a 4 MVA Microgrid Laboratory for Decentralized Power Electronic Applications”. In: *PCESS 2023; Power and Energy Student Summit*. VDE Verlag, 2023, pp. 106–111.
 - [95] N. Pogaku, M. Prodanovic, and T. C. Green. “Modeling, Analysis and Testing of Autonomous Operation of an Inverter-Based Microgrid”. In: *IEEE Transactions on Power Electronics* 22.2 (2007), pp. 613–625. DOI: 10.1109/TPEL.2006.890003.
 - [96] S. D’Arco and J. A. Suul. “Equivalence of Virtual Synchronous Machines and Frequency-Droops for Converter-Based MicroGrids”. In: *IEEE Transactions on Smart Grid* 5.1 (2014), pp. 394–395. DOI: 10.1109/TSG.2013.2288000.
 - [97] E. Coelho, P. Cortizo, and P. Garcia. “Small-signal stability for parallel-connected inverters in stand-alone AC supply systems”. In: *IEEE Transactions on Industry Applications* 38.2 (2002), pp. 533–542. DOI: 10.1109/28.993176.
 - [98] M. Eskandari and A. V. Savkin. “Interaction of Droop Controllers through Complex Power Network in Microgrids”. In: *2020 IEEE 18th International Conference on Industrial Informatics (INDIN)*. Vol. 1. 2020, pp. 311–314. DOI: 10.1109/INDIN45582.2020.9442097.
 - [99] C. M. Bishop. *Neural Networks for Pattern Recognition*. USA: Oxford University Press, Inc., 1995.
 - [100] M. Schenke, B. Haucke-Korber, and O. Wallscheid. “Finite-Set Direct Torque Control via Edge-Computing-Assisted Safe Reinforcement Learning for a Permanent-Magnet Synchronous Motor”. In: *IEEE Transactions on Power Electronics* 38.11 (2023), pp. 13741–13756. DOI: 10.1109/TPEL.2023.3303651.
 - [101] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. “Stable-Baselines3: Reliable Reinforcement Learning Implementations”. In: *Journal of Machine Learning Research* 22.268 (2021).
 - [102] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. *Optuna: A Next-generation Hyperparameter Optimization Framework*. 2019. arXiv: 1907.10902 [cs.LG].
 - [103] E. C. Kerrigan and J. M. Maciejowski. “Soft Constraints And Exact Penalty Functions In Model Predictive Control”. In: *UKACC International Conference (Control 2000)*. 2000.
 - [104] G. C. Goodwin, S. F. Graebe, and M. E. Salgado. *Control System Design*. 1st. USA: Prentice Hall PTR, 2000.
 - [105] F. Hutter, H. Hoos, and K. Leyton-Brown. “An Efficient Approach for Assessing Hyperparameter Importance”. In: *Proceedings of the 31st International Conference on Machine Learning*. Ed. by E. P. Xing and T. Jebara. Vol. 32. Proceedings of Machine Learning Research 1. Beijing, China: PMLR, 2014, pp. 754–762. URL: <https://proceedings.mlr.press/v32/hutter14.html>.
 - [106] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: <http://jmlr.org/papers/v15/srivastava14a.html>.

- [107] S.-Y. Lee, Y.-K. Son, H.-J. Cho, and S.-K. Sul. “Normalization of Capacitor-Discharge I_{2t} by Short-Circuit Fault in VSC-Based DC System”. In: *IEEE Transactions on Power Electronics* 37.1 (2022), pp. 843–854. DOI: 10.1109/TPEL.2021.3096075.
- [108] A. Hintz, U. R. Prasanna, and K. Rajashekara. “Comparative Study of the Three-Phase Grid-Connected Inverter Sharing Unbalanced Three-Phase and/or Single-Phase systems”. In: *IEEE Transactions on Industry Applications* 52.6 (2016), pp. 5156–5164. DOI: 10.1109/TIA.2016.2593680.
- [109] R. Aboelsaud, A. Ibrahim, and A. Garganeev. “Review of three-phase inverters control for unbalanced load compensation”. In: *International Journal of Power Electronics and Drive Systems (IJPEDS)* 10 (Mar. 2019), p. 242. DOI: 10.11591/ijpeds.v10.i1.pp242-255.
- [110] I. Vechiu, H. Camblong, G. Tapia, B. Dakyo, and O. Curea. “Control of four leg inverter for hybrid power system applications with unbalanced load”. In: *Energy Conversion and Management* 48.7 (2007), pp. 2119–2128. DOI: <https://doi.org/10.1016/j.enconman.2006.12.019>.
- [111] X. Quan. “Improved Dynamic Response Design for Proportional Resonant Control Applied to Three-Phase Grid-Forming Inverter”. In: *IEEE Transactions on Industrial Electronics* 68.10 (2021), pp. 9919–9930. DOI: 10.1109/TIE.2020.3021654.
- [112] V.-T. Doan, K.-Y. Kim, W. Choi, and D.-W. Kim. “Design of a hybrid controller for the three-phase four-leg voltage-source inverter with unbalanced load”. In: *Journal of Power Electronics* 17.1 (2017), pp. 181–189.
- [113] T. Geyer. “Model Predictive Direct Current Control for multi-level converters”. In: *2010 IEEE Energy Conversion Congress and Exposition*. 2010, pp. 4305–4312. DOI: 10.1109/ECCE.2010.5618476.
- [114] J. Drgoňa, K. Kiš, A. Tuor, D. Vrabie, and M. Klaučo. “Differentiable Predictive Control: Deep learning Alternative to Explicit Model Predictive Control for Unknown Nonlinear Systems”. In: *Journal of Process Control* 116 (2022), pp. 80–92. DOI: <https://doi.org/10.1016/j.jprocont.2022.06.001>.
- [115] J. Drgoňa, A. Tuor, and D. Vrabie. “Learning Constrained Parametric Differentiable Predictive Control Policies With Guarantees”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 54.6 (2024), pp. 3596–3607. DOI: 10.1109/TSMC.2024.3368026.
- [116] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. “Deterministic Policy Gradient Algorithms”. In: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. Beijing, China: JMLR.org, 2014.
- [117] G. E. Uhlenbeck and L. S. Ornstein. “On the Theory of the Brownian Motion”. In: *Phys. Rev.* 36 (5 Sept. 1930), pp. 823–841. DOI: 10.1103/PhysRev.36.823.