IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 5, 2021, accepted February 23, 2021, date of publication February 24, 2021, date of current version March 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3062144

Safe Bayesian Optimization for Data-Driven Power Electronics Control Design in Microgrids: From Simulations to Real-World Experiments

DANIEL WEBER[1], STEFAN HEID[2], HENRIK BODE[1], JARREN H. LANGE[1], EYKE HÜLLERMEIER[2], (Senior Member, IEEE), AND OLIVER WALLSCHEID[3], (Member, IEEE)

[1]Department of Power Electronics and Electrical Drives, Paderborn University, 33098 Paderborn, Germany
[2]Department of Intelligent Systems and Machine Learning, Paderborn University, 33098 Paderborn, Germany
[3]Department of Automatic Control, Paderborn University, 33098 Paderborn, Germany

Corresponding author: Oliver Wallscheid (oliver.wallscheid@uni-paderborn.de)

This work was supported by the Paderborn University Research Grant.

ABSTRACT Micro- and smart grids (MSG) play an important role both for integrating renewable energy sources in electricity grids and for providing power supply in remote areas. Modern MSGs are largely driven by power electronic converters due to their high efficiency and flexibility. Controlling MSGs is a challenging task due to requirements of power availability, safety and voltage quality within a wide range of different MSG topologies resulting in a demand for comprehensive testing of new control concepts during their development phase. This applies, in particular, to data-driven control approaches such as reinforcement learning, of which the stability and operating behavior can hardly be evaluated on an analytical basis. Therefore, the OpenModelica Microgrid Gym (OMG) package, an open-source software toolbox for the simulation and control optimization of MSGs, is proposed. It is capable of modeling and simulating arbitrary MSG topologies and offers a Python-based interface for plug & play controller testing. In particular, the standardized OpenAI Gym interface allows for easy data-driven control optimization. The usage and benefits of OMG for designing and testing data-driven controllers are demonstrated utilizing Bayesian optimization. Both the current and voltage control loops of a voltage source inverter operating in standalone, grid-forming mode for a remote MSG are automatically tuned given an uncertain application environment. Finally, the transfer to real-world laboratory experiments is successfully demonstrated.

INDEX TERMS Control, data-driven optimization, microgrids, open-source software, power electronics, safety, simulation, testing.

I. INTRODUCTION

The transition of conventional energy supply systems based on fossil fuels to sustainable energy networks characterized by renewable energies is a central technical and social challenge of the 21st century [1]. To achieve this, the inherent volatility of renewable energy sources requires a shift away from conventional, centralized top-down energy networks towards flexible, cross-sectoral and intelligent energy systems [2]. Therefore, in the course of the energy transition, micro- and smart grids (MSG) represent an important solution component to ensure a clean, efficient and cost-effective energy supply [3], [4]. MSG is the concept of a local network consisting of controllable distributed energy resources (e.g. wind power), energy storage units (e.g. battery) and various types of loads [5]. The local integration of renewable energies by means of MSGs, e.g. within industrial companies or residential areas, relieves energy transmission grids and thus reduces the need for cost- and resource-intensive grid expansion. Moreover, MSGs can provide energy supply for remote areas without connection to a public distribution grid. In this context, power electronic converters became the central component of modern MSGs due to their very high energy conversion efficiency and flexibility in order to directly control the power flow between different MSG components [6].

MSGs are highly heterogeneous, complex systems coming with many different topologies depending on their purpose of application [7], [8]. Moreover, their operation contains

The associate editor coordinating the review of this manuscript and approving it for publication was N. Prabaharan.

a significant stochastic component, which is caused by the uncertainty of: the load demand, the regenerative feed-in and topology changes due to the insertion or removal of components during operation. Furthermore, some MSGs may use hybrid AC/DC sub-grids in order to boost energy efficiency by reducing the number of required energy conversion stages. Consequently, controlling MSGs is a demanding task that comes with several key requirements, which can be summarized as follows:

- **Security of supply**: the continuous availability of energy is of prime importance. Outages or component failures due to control errors (e.g. by overloading) are unacceptable.
- **Adaptivity**: due to the wide range of MSG use cases, a high degree of control flexibility in a plug & play sense is necessary.
- **Resource optimality**: minimizing both energy losses and operation costs utilizing available control degrees of freedom is desirable.
- **Power quality**: providing energy supply at high power quality levels is important for ensuring nominal functionality at load side.

In order to pursue these objectives, MSG control is typically addressed by hierarchical approaches on different time scales including [9], [10]:

- **Inner level**: current (and voltage) control in the micro- to millisecond range including auxiliaries such as protective measures or phase-locked loops for each inverter.
- **Primary level**: (re-)active power balancing between different inverters in the (sub-)second range for voltage and/or frequency control.
- **Secondary level**: energy management (including storage scheduling) focusing on mid-term steady-state correction of key grid parameters (e.g. frequency).
- **Tertiary level**: long-term economic dispatch routines for cost-optimal MSG operation (provided one or multiple MSGs contain the appropriate degrees of freedom).

At the various levels, the following control approaches can be summarized [11]:

- Linear feedback controllers such as PID or droop-based characteristics (e.g. [12], [13]),
- (Meta-)heuristic rules and optimization (e.g. [14], [15]),
- Model predictive control (e.g. [16], [17]),
- Data-driven reinforcement learning (e.g. [18], [19]).

In most publications, arbitrary test scenarios are used for the validation of the presented control methods, which are often reduced to a single experimental or simulated MSG example. This inhibits the ability to compare different control procedures on a common test setup. Moreover, there is a lack of checks whether a method remains functional under different operating conditions or within different MSG topologies. Hence, there is a demand for common and open test and development platforms to compare MSG control algorithms with each other and to support the development of novel control approaches.

A. CONTRIBUTION

We present the OpenModelica Microgrid Gym (OMG) package, an open-source software toolbox for the simulation and control optimization of MSGs based on energy conversion by power electronic converters [20], [21]. The main contributions and features of the OMG toolbox are:

- Flexible and scalable simulations of arbitrary MSG topologies using OpenModelica [22] back end;
- Python-interface for easy access, configuration and evaluation of arbitrary controllers;
- OpenAI Gym [23] interface for training reinforcement learning agents or similar data-driven approaches;
- Single and three phase configurations with AC or DC power supply;
- Time domain resolution in the micro- and millisecond range targeting inner and primary level control;
- Fully open-source and collaborative project under GNU GPLv3 license.

The toolbox is under active development and currently focusing on component-oriented simulations targeting the inner and primary control level. Extensions to simplified and lightweight model frameworks for extended simulation horizons, e.g. single-line swing equation models [24], [25], will be added. Selected background information on implementation of the OMG toolbox are presented in the following and more details can be found in the user guide and API documents [20].

Additionally, we present an extended use case of applying safe Bayesian controller optimization [26] to a voltage source inverter (VSI) operating under uncertain application conditions in islanded mode. First, the OMG toolbox is used to deploy the data-driven Bayesian optimization algorithm in simulation to tune the parameters of the VSI's inner current and the outer voltage control loops. Here, the hyperparameters of the optimizer [26] are set appropriately in order to be able to find the optimal controller parameters quickly but also safely. The latter is of prime importance in the context of data-driven control optimization, since such adaptive methods generally explore the unknown search space. In this process, unsafe controller configurations may occur, which are, for example, unstable or tend to overshoot to such an extent that an inverter safety shutdown is triggered. Finally, we demonstrate the feasibility of the proposed OMG toolbox and the safe Bayesian optimization approach by transferring and comparing both to real-world experiments.

B. RELATED WORK

In the domain of power system simulations, the following software toolboxes are often mentioned:

- MATPOWER [27], an open-source, Matlab-based project targeting static power flow simulation and optimization on distribution grid level. Since dynamic modelling is completely omitted, the focus is on secondary and tertiary control level assuming simplified quasi-stationary operation of all components. An OpenAI

Gym interface could be easily added, but is not available yet.
- Pandapower [28], an open-source, Python-based project targeting static power flow simulation and optimization on distribution grid level. It has a similar scope and functionality as MATPOWER.
- PyPSA [29], an open-source, Python-based project targeting static power flow simulation and optimization on distributed grid level. It has a similar scope and functionality as the aforementioned packages.
- PSAT [30], an open-source, Matlab-based project for simplified single line general power system simulation including optimized scheduling. Public user guide or code documentation is not available. It comes with a limited, fixed number of pre-defined primary level controllers. An external controller interfacing is not provided.

Due to the lack of both interfaces and dynamic simulation, the aforementioned packages cannot be considered for the control engineering treatment of MSGs on inner and primary control level. Besides the above mentioned open-source solutions, there is also a range of commercial software with similar functionalities focusing on static grid simulations, which is not reported in detail here. Furthermore, there is a variety of energy market-oriented packages (open-source and commercial) available (e.g. [31]), but since this work is focusing on technical control-oriented problems, these are not discussed here. In the field of dynamic grid and power electronic simulations, the following software packages have to be mentioned:

- Simscape [32] is a commercial Matlab/Simulink extension offered by Mathworks. It enables a wide range of physics-oriented, dynamic modelling applications including power systems and in particular power electronics. Its functionality and scope is similar to the OMG toolbox, but closed-source and interfacing to non-Matlab software products comes with significant calculation overhead (cf. Matlab engine API for Python [33]).
- SPICE-related software such as LTspice, PLECS or ngspice focus on integrated circuit simulations often with nanosecond range time steps. Therefore, it is suitable to accurately simulate single power electronic converters on small simulation durations, but computationally not feasible for MSGs with multiple power units.

Therefore, OMG is currently the only available open-source toolbox for dynamic power electronics-driven MSG evaluations on small time scales. Due to the offered interfaces, it is particularly suitable for control development and testing, including training and evaluation of recent data-driven control techniques.

II. SOFTWARE DESCRIPTION

A. TOOLBOX STRUCTURE

The overall structure of the software package is inspired by the Tensorforce library [34]. OMG contains wrappers for

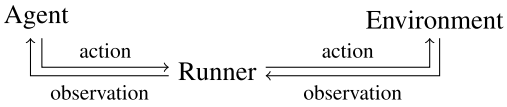

FIGURE 1. High level code architecture.

OpenAI Gym environments as well as fully implemented control agents. One of the main contributions of this toolbox is an OpenAI Gym instance in which data-driven controllers, in particular reinforcement learning (RL) agents, can be trained and tested. For ease of use, predefined agents for controlling the environment, as well as a service class that handles the execution of the specified agent on its environment, are provided. This reduces the boilerplate code and allows developers and engineers to focus on designing and testing controllers.

The *runner* class is responsible for the initialization and termination of agents and environments, as well as the execution of multiple episodes. The class also handles information exchange between agent and environment, as shown in Fig. 1. This functionality is convenient, as the training of an agent usually spans multiple training epochs.

The *agent* class encapsulates all states related to the learning process. For example, it may contain a base controller such as a linear feedback controller that will be parameterized by external agents during learning and provide the control actions that the agent plays out on the environment (cf. Sec. III). This example corresponds to a hybrid approach mixing expert-driven and data-driven control. Nevertheless, the definition of the OMG interfaces is completely open, allowing the toolbox to be connected to a wide range of solutions, between entirely data-driven and entirely expert-driven. The agent also provides tools to log debugging data from the learning process.

The configurable *environment* class provides an interface from OpenAI Gym to the internal simulation model. It will record data for monitoring and visualizing each epoch, as well as analyzing the control performance in more depth.

B. MODELICA INTEGRATION

For modeling on MSG component level, OpenModelica (OM) [22] is used in the back end. OM is also based on an open-source policy and comes with a graphical user interface (GUI) for easy and clear MSG modeling. The overall integration of OM in the OMG toolbox is shown in Fig. 2. For the model transfer from OM to the OMG Python front end, the functional mock-up interface (FMI) [35] is used. FMI is a tool-independent, open-source standard for the exchange of dynamic models. According to this standard, the model-including objects created for the exchange are called functional mock-up units (FMU), which support two flavors of simulation types: co-simulation (CS) and model exchange (ME). In CS, the numerical solver is embedded and supplied by the exporting tool, in this case, OpenModelica. On the other side, in ME, the importing tool supplies the solver, while the FMU only provides the ordinary differential equation (ODE) system.

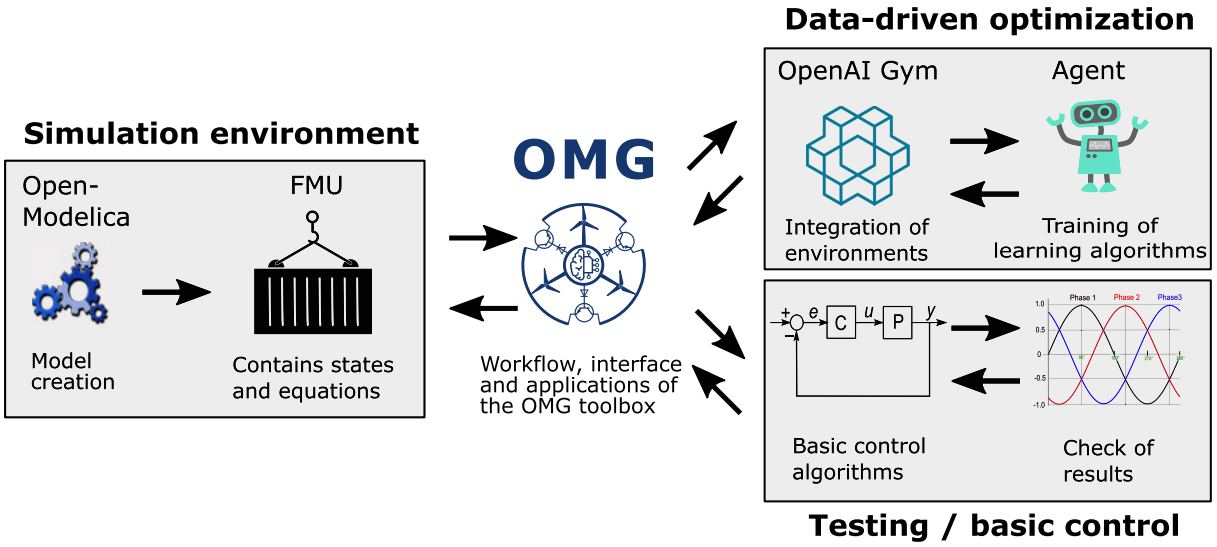

FIGURE 2. Overview of the interconnections between the different parts of the OMG toolbox. The OpenModelica and OpenAI Gym logos are the property of their respective owners.

To avoid unnecessarily small simulation step sizes to ensure numerical stability, implicit solvers are required. Currently, OpenModelica only provides explicit solvers in CS, therefore, ME is used in OMG. Since the Python package SciPy [36] provides several implicit solvers, it is used for solving the resulting ODE systems describing power electronics-driven MSGs modeled in the OM back end.

The FMU containing the model is imported via PyFMI [37]. This library interfaces between the FMU and Python. After extracting the initial states and the equation system, the controller/agent selects the action vector (i.e. control input) for the following step. Next, the equation system is solved, and the new states of the model are saved for the next simulation step.

Executing the simulation in such a step-by-step manner adds computational overhead, in comparison to a continuous simulation, however, it is crucial to be able to select actions freely after each simulation step. This allows the integration of arbitrary discrete-time controllers via the Python front end. Furthermore, this corresponds to the stepwise interaction of system model and data-driven controller in the context of RL.

C. MICROGRID MODELICA LIBRARY

Together with the OMG Python package, an OpenModelica library to create customized MSG topologies is provided [38]. It mainly consists of freely connectable components like inverters, filters, and loads. The library is sketched in Fig. 3 together with an example network.

In this example, a DC bus, which can be adjusted via Python, supplies each inverter. The inverters can be connected via filters (e.g. LC or LCL). Moreover, a wide range of different load nodes are pre-defined in the library, which can be extended by the user. The filters and loads can be freely parameterized, either directly in the OpenModelica model or via PyFMI. Besides, the toolbox provides auxiliary

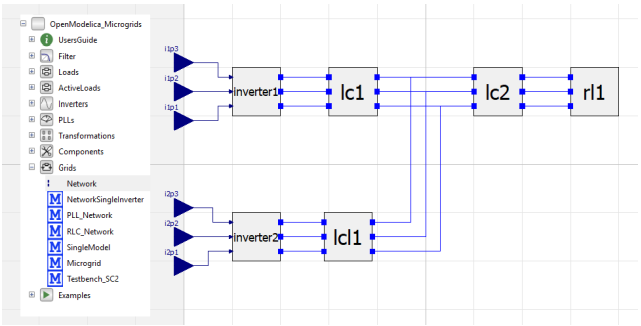

FIGURE 3. OpenModelica microgrid library example.

components (such as phase-locked loops) and pre-defined voltage and current forming inverters, the latter with direct and indirect droop controllers.

For each filter design, models with and without losses are included. Due to their impact on the complexity of the system and the resulting increase of simulation time, it is recommended to use the loss models only if the efficiency and loss behavior is of interest.

A dynamic environment with a variable supply voltage or load steps can be archived through parameter-variation. Any parameter can be adjusted freely at any time during the simulation directly in the Python interface.

III. INVERTER MODEL AND CONTROL FRAMEWORK

The previously introduced OMG toolbox is applied to an exemplary automatic inverter controller tuning process by safe Bayesian optimization to highlight its usage in MSG control scenarios. In particular, the current and voltage control loops of a single VSI operating in islanded mode are investigated under uncertain operation conditions. To this end, we will briefly summarize the fundamentals of the VSI system model followed by the inner level current and voltage

control architecture, before linking this with a data-driven controller optimization. The entire following example can be found as executable code in the OMG repository [20].

A. BASIC SYSTEM MODEL

For this contribution, we consider a three-phase, two-level VSI connected to an unknown load as depicted in Fig. 4. For an arbitrary inverter's phase p assuming a standard LC filter configuration ($L_{\mathrm{f,p}}$, $C_{\mathrm{f,p}}$), with inductor filter current $i_{\mathrm{f,p}}$, voltage over the filter capacitor $v_{\mathrm{f,p}}$ and output load current $i_{\mathrm{o,p}}$, the dynamics can be linearly described as

$$i_{\mathrm{f,p}} = C_{\mathrm{f,p}} \frac{\mathrm{d}v_{\mathrm{f,p}}}{\mathrm{d}t} + i_{\mathrm{o,p}}, \tag{1}$$

with the switch voltage $v_{\mathrm{i,p}}$

$$v_{\mathrm{i,p}} = v_{\mathrm{c,p}} + L_{\mathrm{f,p}} \frac{\mathrm{d}i_{\mathrm{f,p}}}{\mathrm{d}t} + R_{\mathrm{f,p}} i_{\mathrm{f,p}}. \tag{2}$$

Above, $R_{\mathrm{f,p}}$ is the inductors' internal resistance, while it is assumed that the internal resistance of the capacitors is negligible. Moreover, the model can be described linearly in the state space

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{E}\boldsymbol{i}_{\boldsymbol{o}}, \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x}, \end{aligned} \tag{3}$$

where $\boldsymbol{x}$ contains the states, the inputs are $\boldsymbol{u}$ and $\boldsymbol{i}_o$ is the load current (interpreted as a disturbance):

$$\begin{aligned} \boldsymbol{x} &= \begin{bmatrix} i_{\mathrm{f,a}} & i_{\mathrm{f,b}} & i_{\mathrm{f,c}} & v_{\mathrm{f,a}} & v_{\mathrm{f,b}} & v_{\mathrm{f,c}} \end{bmatrix}^{\mathsf{T}}, \\ \boldsymbol{u} &= \begin{bmatrix} v_{\mathrm{i,a}} & v_{\mathrm{i,b}} & v_{\mathrm{i,c}} \end{bmatrix}^{\mathsf{T}}, \quad \boldsymbol{i_o} = \begin{bmatrix} i_{\mathrm{o,a}} & i_{\mathrm{o,b}} & i_{\mathrm{o,c}} \end{bmatrix}^{\mathsf{T}}. \end{aligned} \tag{4}$$

The corresponding state-space matrices are

$$\begin{aligned} \boldsymbol{A} &= \begin{bmatrix} -\frac{R_{\mathrm{f,a}}}{L_{\mathrm{f,a}}} & 0 & 0 & -\frac{1}{L_{\mathrm{f,a}}} & 0 & 0 \\ 0 & -\frac{R_{\mathrm{f,b}}}{L_{\mathrm{f,b}}} & 0 & 0 & -\frac{1}{L_{\mathrm{f,b}}} & 0 \\ 0 & 0 & -\frac{R_{\mathrm{f,c}}}{L_{\mathrm{f,c}}} & 0 & 0 & -\frac{1}{L_{\mathrm{f,c}}} \\ \frac{1}{C_{\mathrm{f,a}}} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{C_{\mathrm{f,b}}} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{C_{\mathrm{f,c}}} & 0 & 0 & 0 \end{bmatrix}, \\ \boldsymbol{B} &= \begin{bmatrix} \frac{1}{L_{\mathrm{f,a}}} & 0 & 0 \\ 0 & \frac{1}{L_{\mathrm{f,b}}} & 0 \\ 0 & 0 & \frac{1}{L_{\mathrm{f,c}}} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{E} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{C_{\mathrm{f,a}}} & 0 & 0 \\ 0 & \frac{1}{C_{\mathrm{f,b}}} & 0 \\ 0 & 0 & \frac{1}{C_{\mathrm{f,c}}} \end{bmatrix}, \\ \boldsymbol{C} &= \boldsymbol{I_6}. \end{aligned} \tag{5}$$

Above, $\boldsymbol{C} = \boldsymbol{I_6}$ is the 6×6 identity matrix assuming that the filter currents and voltages are measured while the load currents are not available as a measurement signals.

So far, the relevant currents and voltages ($v_{\mathrm{c,p}}$, $i_{\mathrm{o,p}}$) are represented as vectors in a fixed abc reference frame. Hence, in steady state they are rotating with the frequency of the sinusoidal supply voltage. Using the Park transformation, the system variables can be mapped into a rotating coordinate

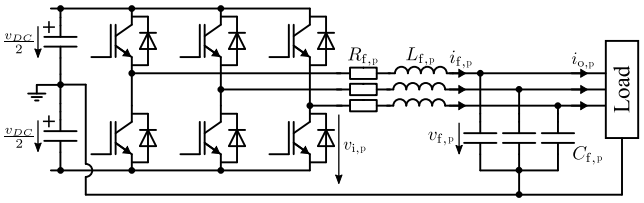

FIGURE 4. Three-phase four-wire inverter with a split DC bus and a LC output filter connected to an unknown load.

systems. Here, the d-axis is aligned with the a-axis of the rotating three-phase system, the q-axis is orthogonal to the d-axis and the third is the zero component:

$$\begin{bmatrix} x_{\mathrm{d}} \\ x_{\mathrm{q}} \\ x_0 \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \cos(\theta) & \cos(\theta - \frac{2\pi}{3}) & \cos(\theta - \frac{4\pi}{3}) \\ -\sin(\theta) & -\sin(\theta - \frac{2\pi}{3}) & -\sin(\theta - \frac{4\pi}{3}) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x_{\mathrm{a}} \\ x_{\mathrm{b}} \\ x_{\mathrm{c}} \end{bmatrix}.$$

If the angular speed ω of the rotating frame is set equal to the grid frequency, the balanced sinusoidal grid voltages and currents become stationary DC-variables. This simplifies the control design, allowing for the effective application of linear feedback controllers. For more information on the basics of power electronic control we refer to [39] and similar textbooks.

Assuming that all LC filter components are ideal and symmetrical, we can neglect the null component (as is common in AC power systems). In this simplified case, the model can be reduced to the dq equivalent:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{E}\boldsymbol{i}_{\boldsymbol{o}}, \\ \boldsymbol{y} &= \boldsymbol{C}\boldsymbol{x}, \end{aligned} \tag{6}$$

where the transformed quantities are

$$\begin{aligned} \boldsymbol{x} &= \begin{bmatrix} i_{\mathrm{f,d}} & i_{\mathrm{f,q}} & v_{\mathrm{c,d}} & v_{\mathrm{c,q}} \end{bmatrix}^{\mathsf{T}}, \\ \boldsymbol{u} &= \begin{bmatrix} v_{\mathrm{i,d}} & v_{\mathrm{i,q}} \end{bmatrix}^{\mathsf{T}}, \quad \boldsymbol{i_o} = \begin{bmatrix} i_{\mathrm{o,d}} & i_{\mathrm{o,q}} \end{bmatrix}^{\mathsf{T}}, \end{aligned} \tag{7}$$

and the transformed matrices yield

$$\begin{aligned} \boldsymbol{A} &= \begin{bmatrix} -\frac{R_{\mathrm{f}}}{L_{\mathrm{f}}} & -\omega & -\frac{1}{3L_{\mathrm{f}}} & 0 \\ \omega & -\frac{R_{\mathrm{f}}}{L_{\mathrm{f}}} & 0 & -\frac{1}{3L_{\mathrm{f}}} \\ \frac{1}{3C_{\mathrm{f}}} & 0 & 0 & -\omega \\ 0 & \frac{1}{3C_{\mathrm{f}}} & \omega & 0 \end{bmatrix}, \\ \boldsymbol{B} &= \begin{bmatrix} \frac{1}{3L_{\mathrm{f}}} & 0 \\ 0 & \frac{1}{3L_{\mathrm{f}}} \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad \boldsymbol{E} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -\frac{1}{3C_{\mathrm{f}}} & 0 \\ 0 & -\frac{1}{3C_{\mathrm{f}}} \end{bmatrix}, \\ \boldsymbol{C} &= \boldsymbol{I_4}. \end{aligned} \tag{8}$$

For the remaining part of the paper, the simulation of the physical grid system is executed in the abc frame within OMG framework allowing to cover realistic operation scenarios like asymmetric loads or non-ideal system configurations due to parameter deviations of individual components.

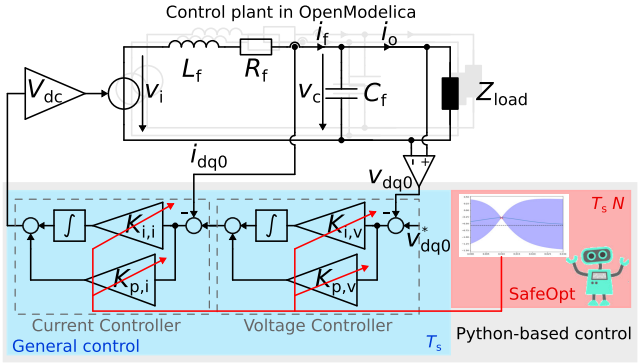

FIGURE 5. Simplified control diagram highlighting the integration of safe Bayesian optimization for data-driven control tuning.

TABLE 1. Considered system parameters.

	Symbol	Value	Std. Dev.	Clip Limits
Filter inductance	L_f	2.3 mH	0.23 mH	±10 %
Filter resistance	R_f	400 mΩ	40 mΩ	±10 %
Filter capacitance	C_f	10 µF	1 µF	±10 %
Current meas. noise	i_noise	0 A	1.8 mA	[0.5, 3.2] mA
Voltage meas. noise	v_noise	0 V	0.42 V	[0, 0.5] V
DC-link voltage	v_DC	600V	-	-
Sample time	T_s	0.1ms	-	-
Timesteps	N	1000	-	-
Episodes	M	40	-	-
Monte-Carlo samples	n_MC	10	-	-

In comparison, the simplified dq frame model is utilized to perform an initial control design as will be discussed next.

B. CONTROL FRAMEWORK AND PROBLEM STATEMENT

In Fig. 5 a simplified control block diagram of the considered system is shown. It is assumed that the inverter is operating in an islanded and grid-forming mode, e.g. setting up a rural and remote MSG.

The control structure is cascaded with an inner current and an outer voltage control loop. Knowledge of the system parameters is generally required to parametrize the controllers. In real-world applications, however, the problem arises that these are not exactly known, for example due to production-related tolerances or uneven aging [40]. Plug & play component insertion or removal, e.g. during repair or maintenance work, can also contribute to drastically system changes [41], [42]. Likewise, the load characteristics can change drastically and are generally not known in advance, especially in the case of remote MSGs. In addition, linear feedback controllers are generally designed without taking into account non-linearities in the control loop, in particular constraints on the manipulated variables and anti-reset windup measures. Due to this multitude of issues, an analytical controller design can lead to significantly different, in particular worse, behavior in the actual system than predicted based on its nominal model.

In order to compensate for these issues, a two-fold procedure is suggested:

1) Assuming that the nominal parameters are roughly known, an initial model-based controller is designed.
2) During operations, the controller design is improved by Bayesian optimization to compensate for deviations between model assumption and actual system behavior.

As an important constraint, this data-driven tuning has to be performed in a safe way, such that unsuitable gain parameters leading to severe overshoots or other unsafe system behavior are prohibited at all times. While the optimization algorithm is explained in Sec. IV, the uncertainty modeling and the model-based control design is addressed in Sec. III-C.

Assuming that a rough system model is available, an analytical, model-based control design method can used to calculate initial controller parameters. For this contribution, the magnitude optimum method is considered [39], [43]. To layout the current controller a cross-over frequency of $f_\mathrm{C} = \frac{1}{6T_\mathrm{s}}$ and a phase margin of 60° was used and the open-loop control plant was considered using the transfer function

$$G_{\mathrm{OL,c}}(s) = \left(K_{\mathrm{p,c}} + \frac{K_{\mathrm{i,c}}}{s}\right)\left(\frac{1 - s\frac{T_\mathrm{s}}{4}}{1 + s\frac{T_\mathrm{s}}{4}}\right)\frac{v_\mathrm{dc}}{R_\mathrm{f}}\frac{1}{1 + s\frac{L_\mathrm{f}}{R_\mathrm{f}}}. \quad (9)$$

The second bracket term is the Pade approximation for representing the delay effect of the inverter modulation scheme, since this quasi-continuous design approach is transferred to the discrete-time domain for computerized control [39]. Using the nominal system parameters from Tab. 1, the initial controller values are

$$K_{\mathrm{p,c}} = 0.04\,\mathrm{V/A} \text{ and } K_{\mathrm{i,c}} = 12\,\mathrm{V/(As)}\,. \quad (10)$$

Likewise, the same design approach is used to find the initial voltage controller parameters. The chosen cross-over frequency was set to 300 Hz and the phase margin to 60°. The open-loop transfer function for the voltage controller layout was considered using the transfer function

$$G_{\mathrm{OL,v}} = \left(K_{\mathrm{p,v}} + \frac{K_{\mathrm{i,v}}}{s}\right)\frac{G_{\mathrm{CL,c}}}{s\,C_\mathrm{f}}, \quad (11)$$

taking the closed-loop current control plant

$$G_{\mathrm{CL,c}} = \frac{G_{\mathrm{OL,c}}}{1 + G_{\mathrm{OL,c}}} \quad (12)$$

using (9). Accordingly, the controller parameters

$$K_{\mathrm{p,v}} = 0.0175\,\mathrm{A/V} \text{ and } K_{\mathrm{i,v}} = 12\,\mathrm{A/(Vs)} \quad (13)$$

were taken as initial parameters. All controllers are transferred to the discrete-time domain using the forward Euler method.

C. MODEL UNCERTAINTIES AND MONTE-CARLO SIMULATIONS

To represent parameter uncertainty in simulations, at the beginning of an episode the device parameters P_i (resistors', indudctors', capacitors' and load values) are drawn from a Gaussian normal distribution $\mathcal{N}(\mu = P_i, \sigma = 0.1 \cdot P_i)$, using the device nominal parameter as an average value and assuming a device tolerance of 10 %. Balanced or unbalanced parameters can be chosen by selecting the same parameter for all three phases or different per phase. For noise and device parameter samples, clipping is implemented to prevent too high deviations and limit the device tolerance.

Moreover noise is added to all measurement signals based on a Gaussian normal distribution $\mathcal{N}(0; \sigma)$. For the upcoming real-world investigation in Sec. V, a current and voltage noise measurement has been performed at a test bench using setpoints of $\boldsymbol{v}^*_{\mathrm{dq0}} = [0, 0, 0]^{\mathsf{T}}$ V and $\boldsymbol{i}^*_{\mathrm{dq0}} = [0, 0, 0]^{\mathsf{T}}$ A, respectively. The measured standard deviation σ_{v} and σ_{i} are listed in Tab. 1. Because of the implementation of automatic offset nulling, the average values are chosen to be zero.

For every episode of the later shown experiments a Monte-Carlo simulation is applied to take a representative performance sample to evaluate the used controller parameters. Thereby, the simulation is executed several times using the same controller parameters but different device and noise parameter samples. At the end of the Monte-Carlo simulation, the average performance of all n_{MC} Monte-Carlo measurements is used for a simulation-based parameter optimization of the control design.

The Monte-Carlo approach is an important intermediate step to address the system uncertainty already when simulating. Moreover, it is the necessary basis for tuning the abstract hyperparameters of the safe Bayesian optimization approach (cf. Sec. IV) to ensure that the algorithm performs only safe controller updates during real-world investigations when interacting with an actual, physical system (cf. Sec. V). Thus, the Monte-Carlo-based simulation is the important middle layer of the proposed simulation to real-world experiment pipeline as shown in Fig. 6.

IV. SAFE BAYESIAN OPTIMIZATION FOR CONTROLLER TUNING

In this section, we first introduce the safe Bayesian optimization approach for data-driven controller tuning. Then, we apply the methodology using the previously introduced simulation framework to determine suitable hyperparameter values, i.e., configure the optimizers degrees of freedom. This is done for several control scenarios:

1) only current controller optimization,
2) sequential current and voltage controller optimization,
3) parallel current and voltage controller optimization.

Another focus is on the application-specific definition of reward functions to ensure consistently high controller performance during and after Bayesian optimization.

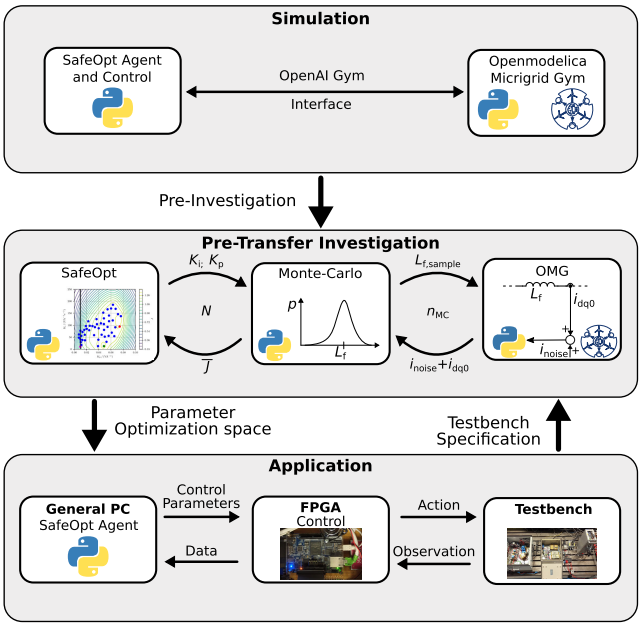

FIGURE 6. Transfer from simulation to real world using extensive pre-investigations for hyperparameter configuration.

A. SafeOpt

The algorithm's task is to find optimal controller parameters during online operation. To avoid failures during this procedure, e.g. encountering unsafe control parameters leading to dangerously high overshoots, an extension of *Bayesian optimization* [44], called *SafeOpt*, has been proposed in [26].

The general concept behind Bayesian optimization is to learn a model of the optimization landscape that can predict expected performance along with an uncertainty measure. In our case the performance landscape is a parameterspace Θ. When evaluating a specific parameter we query the function $J : \Theta \rightarrow \mathbb{R}$ which maps each parameter to a performance value we aim to maximize.

The functions

$$u_m(\boldsymbol{\theta}) = \mu_{m-1}(\boldsymbol{\theta}) + \beta_m \sigma_{m-1}(\boldsymbol{\theta}), \tag{14}$$

$$l_m(\boldsymbol{\theta}) = \mu_{m-1}(\boldsymbol{\theta}) - \beta_m \sigma_{m-1}(\boldsymbol{\theta}), \tag{15}$$

define the upper and lower confidence bound of a parameter vector $\boldsymbol{\theta}$ with $m \in \{1, \ldots, M\}$. $\mu_{m-1}(\cdot)$ predicts the expected performance value with respect to the $m-1$ parameters evaluated so far. $\sigma_{m-1}(\cdot)$ predicts the variance of the performance estimate with respect to the currently evaluated points, hence it is an uncertainty estimate. β_m is a scaling parameter for the variance that can be changed over time to trade-off exploitation and exploration. The next data point, i.e., a promising controller parameter set, to be evaluated by $J(\cdot)$ is calculated with respect to the upper confidence bound [45]

$$\boldsymbol{\theta}_{m+1} = \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmax}}\, u_m(\boldsymbol{\theta}). \tag{16}$$

Gaussian process (GP) regression is commonly used in Bayesian optimization as it is comparatively data efficient

when configured with good prior knowledge. This prior knowledge is expressed in the selection of the covariance function $k(\cdot\,,\,\cdot)$, which is a similarity measure defined over a tuple of data points (parameters). The function expresses how strongly performance values are correlated depending on the difference in two parameter vectors. Various functions can be used which have different hyperparameters, the most prominent being the lengthscale ℓ, which defines how fast the correlation decays with respect to distance. As the performance evaluation is subject to noise, the GP is commonly configured to account for that. Hence, we aim to maximize $\hat{J}_m = J_m + \omega$ with $\omega \sim \mathcal{N}(0, \xi)$. The mean and variance function used in (14) for GP are defined as follows:

$$\mu_m(\boldsymbol{\theta}) = \boldsymbol{k}_m(\boldsymbol{\theta})(\boldsymbol{K}_m + \mathbf{I}_m\xi)^{-1}\hat{\boldsymbol{J}}_m^{\mathsf{T}}, \tag{17}$$

$$\sigma_m^2(\boldsymbol{\theta}) = k(\boldsymbol{\theta},\boldsymbol{\theta}) - \boldsymbol{k}_m(\boldsymbol{\theta})(\boldsymbol{K}_m + \mathbf{I}_m\xi)^{-1}\boldsymbol{k}_m^{\mathsf{T}}(\boldsymbol{\theta}), \tag{18}$$

$$\boldsymbol{k}_m(\boldsymbol{\theta}) = [k(\boldsymbol{\theta},\boldsymbol{\theta}_1),\ldots,k(\boldsymbol{\theta},\boldsymbol{\theta}_m)], \tag{19}$$

$$[\boldsymbol{K}_m]_{(i,j)} = k(\boldsymbol{\theta}_i,\boldsymbol{\theta}_j);\, i,j \in \{1,\ldots,m\}, \tag{20}$$

$$\hat{\boldsymbol{J}}_m = [\hat{J}(\boldsymbol{\theta}_1),\ldots,\hat{J}(\boldsymbol{\theta}_m)]. \tag{21}$$

The matrix $\boldsymbol{K}$ is called covariance matrix and expresses all pairwise similarities between the recorded data points. The vector $\boldsymbol{k}$ expresses the similarity of the current point towards the known data samples. I_m is the identity matrix of size m.

For all later experiments the Matérn kernel

$$k_{\nu=3/2}(r) = \left(1 + \frac{\sqrt{3}r}{\ell}\right)\exp\left(-\frac{\sqrt{3}r}{\ell}\right) \tag{22}$$

where $r = \boldsymbol{\theta}_i - \boldsymbol{\theta}_j$ and the lengthscale ℓ is used as covariance function for the GP regression. Additionally, boundaries $\mathcal{B}$ for the range of the parameters were set application specific. For more details refer [46, 16ff.].

The *SafeOpt* algorithm proposed in [26] defines safety in a probabilistic manner. The set of parameters considered *safe* at episode m is called $\mathcal{S}_m$. This set is also called *safe region*. It consists of all parameters whose lower confidence performance is above our safety threshold J_{min}

$$\mathcal{S}_m = \{\boldsymbol{\theta} \in \Theta \mid l_m(\boldsymbol{\theta}) \geq J_{min}\}, \tag{23}$$

with Θ being the parameter space and $\boldsymbol{\theta}$ a parametrization of the controller.

From this safe set only parameter are considered for evaluation that eighter extend our safe set and therefore the search space or have a chance to improve upon the best known parameter so far. Those two, not necessarily disjoint sets are called *expanders* and *maximizers* and attempt to solve the well-known exploration-exploitation dilemma.

Expanders are parameters which potentially extend the safe set:

$$\mathcal{E}_m = \{\boldsymbol{\theta} \in \mathcal{S}_m \mid g_m(\boldsymbol{\theta}) > 0\}, \tag{24}$$

$$g_m(\boldsymbol{\theta}) = \left|\{\boldsymbol{\theta}' \in \Theta \setminus \mathcal{S}_m \mid l_{m,(\boldsymbol{\theta},u_m(\boldsymbol{\theta}))}(\boldsymbol{\theta}') \geq J_{min}\}\right|. \tag{25}$$

However, here we do calculate the $l_{m,(\boldsymbol{\theta},u_m(\boldsymbol{\theta}))}(\cdot)$ only with the m points know so far, but also included the additional datapoint $\boldsymbol{\theta}$ along with the most optimistic estimate of its performance $u_m(\boldsymbol{\theta})$) (15) Hence, the indicator function $g_m(\cdot)$ predicts the number of points that would be added to the safe set in case the $\boldsymbol{\theta}$ yields the best performance that could be expected.

Maximizers are points whose upper confidence bound exceed the currently highest lower bound of the optimal performance:

$$\mathcal{M}_m = \{\boldsymbol{\theta} \in \mathcal{S}_m \mid u_m(\boldsymbol{\theta}) \geq \max_{\boldsymbol{\theta}' \in \Theta} l_m(\boldsymbol{\theta}')\}. \tag{26}$$

Finally the next parameter selected for evaluation is the point among expanders and maximizers that with maximum uncertainty:

$$\boldsymbol{\theta}_{m+1} = \operatorname*{argmax}_{\boldsymbol{\theta} \in \mathcal{M} \cup \mathcal{E}} \sigma(\boldsymbol{\theta}). \tag{27}$$

The discussed safety concept therefore relies on two assumptions:

- First, the performance in the learning environment must be representative of the real applications faced by the learned agent later on. This is important, as each parameterization is only evaluated a predefined number of time steps in its learning environment.
- Second, the GP must be able to sufficiently fit the observed performance, otherwise the confidence bounds are not reliable.

For this reason, the pre-investigation using Monte-Carlo simulation includes various model uncertainties in order to catch possible worst-case scenarios (cf. Sec. III-C). Furthermore, we will show that the abstract hyperparameters of the discussed algorithm (in particular the lengthscale ℓ) must be fine tuned in the course of the pre-investigation in order to prevent unsafe controller parameterization in the real-world experiments.

B. BASIC SETUP FOR CURRENT CONTROLLER OPTIMIZATION

Safe Bayesian optimization is now used for tuning the current controller based on the Monte-Carlo simulation framework as discussed on Sec. III-C. The load is $Z_{\text{load}} = 0\ \Omega$ to model a short circuit, which is a standard approach when investigating the current control loop. In this exemplary use case, the inverter should supply a current of

$$\boldsymbol{i}_{\text{dq0}}^{*} = \begin{cases} [10\,\text{A}, 0, 0]^{\mathsf{T}}, & t < 20\,\text{ms}, \\ [5\,\text{A}, 0, 0]^{\mathsf{T}}, & \text{afterwards}, \end{cases} \tag{28}$$

while all three-phases operate at a grid frequency of 60 Hz. The inverter is fed by an idealized DC source. Furthermore, for all subsequent training episodes a blackstart is assumed (i.e., all voltages and currents are initially zero).

In this work, the term *safety* is defined as a performance metric J induced by the environments rewards and normalized with respect to an empirically derived critical performance. In the defined reward function (30), used for the current controller layout, the mean-root-error (MRE) between the

measured phase currents[1] i_p and the setpoints i_p^* is provided as the regular performance indicator, for $p \in \{a, b, c\}$. However, the MRE is only an exemplary way to evaluate the control performance. Compared to the classical mean-squared-error (MSE) metric the MRE is penalizing smaller control errors around zero stronger and, therefore, focuses on reducing steady-state control errors. Nevertheless, arbitrary performance measures can be included in the control agent definition within the OMG toolbox.

Additional to the MRE, a barrier function is used as a penalty if the nominal current i_nom is exceeded to avoid that the current limit i_limit is reached. It is assumed that exceeding i_limit will either lead to severe component damage (e.g. by thermal overloading of the power electronic semiconductors) or to an automatic emergency shutdown of the system. For this example, the nominal current and its limit are defined by the physical system like described in Sec. V by

$$i_\mathrm{nom} = 12\,\mathrm{A}, \quad i_\mathrm{limit} = 16\,\mathrm{A}\,.$$

The total reward is then given by

$$r_n = -\frac{1}{N} \sum_{p\in\{\mathrm{a,b,c}\}} \left[\sqrt{\frac{\left|i_{p,n}^* - i_{p,n}\right|}{i_\mathrm{limit}}} - \lambda_\mathrm{c} \cdot \log\left(1 - \frac{\max(|i_{p,n}| - i_\mathrm{nom}, 0)}{i_\mathrm{limit} - i_\mathrm{nom}}\right) \right] \tag{29}$$

for $n \in \{1, \ldots, N\}$ time steps of the discrete-time controller running at $T_s = 100\,\mu\mathrm{s}$ (cf. Tab. 1). Moreover, λ_c is the weight of the barrier function penalty which was set to $\lambda_\mathrm{c} = 80$. The performance J is calculated depending on the average reward per episode over all, N, time steps:

$$J = \frac{\sum_{n=1}^{N} r_n - J_\mathrm{lim}}{J_\mathrm{init} - J_\mathrm{lim}}. \tag{30}$$

Here, J_init is the performance of the initial, safe parameter set and J_lim defines the minimal allowed performance to ensure safety. As a result of this normalization $J_\mathrm{min} = 0$.

To find J_lim for the current controller investigation, (30) is used for two scenarios. First assuming a maximally allowed phase shift of 5°, then assuming an amplitude deviation of 10 % between measurement and setpoint. In both scenarios a blackstart is simulated applying a ramp up to the wanted setpoint. Beyond that the setpoint is changed once in each scenario just like in the later experiments. J_lim is set equal to the maximum return of the two experiment, i.e., using the stricter of the two constraints.

C. AN UNSAFE DEMONSTRATION

Fig. 7 shows a visualization of the GP model with 15 samples to highlight the working principle of safe Bayesian optimization. For a more intuitive demonstration, only the control parameter $K_\mathrm{i,c}$ of the current controller is considered for the optimization and initialized with $5\,\mathrm{V/(As)}$, while $K_\mathrm{p,c} = 0.005\,\mathrm{V/A}$ is fixed. The blue curve line shows the mean function of the Gaussian process (17). The blue region surrounding the mean function shows the 95 % confidence bounds and is defines using (18). J_min is indicated by the dashed line.

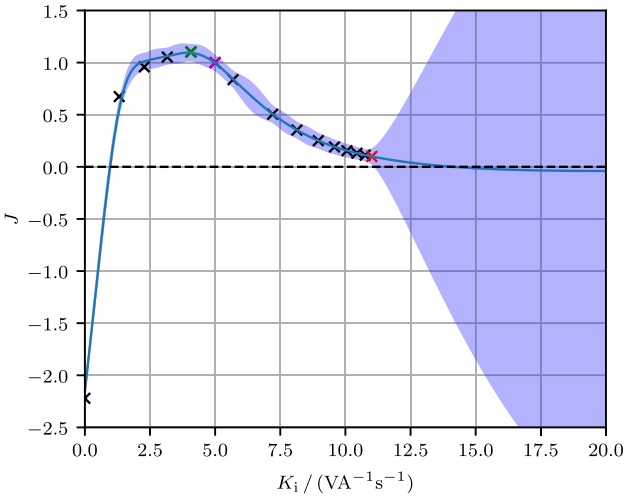

FIGURE 7. Resulting performance measurement and GP model after 15 episodes. Only K_i is adjusted using *SafeOpt* while $K_\mathrm{p} = 0.005\,\mathrm{V/A}$ is kept constant. The dashed line indicates the safe threshold, the blue curve the mean function and the blue region the 95 % confidence bounds of the GP. The red marker represents the last, the magenta the initial and the green the best measurement.

In GP regression, the fitted function is expected to change equally fast over the whole domain. This assumption is expressed in the covariance function and its lengthscale (compare Sec. IV-A). Fig. 7, however, shows that such a performance function can be fairly constant in some regions and shows rapid change in others. To allow the GP to predict confidence bounds in regions of fast change, the lengthscale of the covariance function has to be chosen extremely small. This means that parameters that are expected to have strong correlation in their performance values lay very close to each other. This parameterization, however, also results in very conservative exploration in regions with fairly constant performance. Unfortunately, the GP cannot predict the steep performance drop (cf. again Fig. 7) if neither the prior expectation nor the observed data indicate such a steep change. Therefore, selecting a larger lengthscale results in highly overconfident behavior as the boundaries of the performance plateau are missed.

Accordingly, a default *SafeOpt* algorithm configuration cannot solve the given task satisfactorily and requires additional hyperparameter tuning, in particular, finding lengthscale values ensuring the fastest possible exploration while preventing unsafe trials as demonstrated in Fig. 7. If the optimization algorithm with the shown configuration would had been tested on a real, technical system, this would presumably have caused an emergency shutdown. This illustrates that

[1]For simplified notation we refer to the filter inductor current $i_\mathrm{f,p}$ as the phase current i_p (cf. Fig. 4).

TABLE 2. Hyperparameters for the current controller opt.

	GP lengthscale ℓ	GP boundaries $\mathcal{B}$
$K_{\mathrm{p,c}}$	$0.012\,\mathrm{V/A}$	$[0.001, 0.07]\,\mathrm{V/A}$
$K_{\mathrm{i,c}}$	$30\,\mathrm{V/(As)}$	$[2, 150]\,\mathrm{V/(As)}$
Barrier scaling λ_{c}	80	

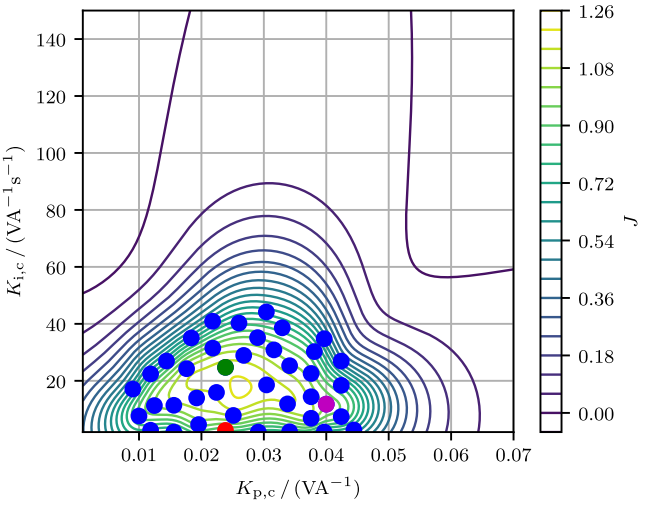

FIGURE 8. Simulated performance landscape (z-axis) for the current controller optimization with variable $K_{\mathrm{p,c}}$ and $K_{\mathrm{i,c}}$ using parameter setting depicted in Tab. 1.

a simulative pre-investigation for hyperparameter tuning is indispensable.

D. CURRENT CONTROLLER OPTIMIZATION

After discussing the optimization setup and a simplified, single parameter demonstration, the application is extended to optimize both $K_{\mathrm{p,c}}$ and $K_{\mathrm{i,c}}$. As explained in before the lengthscale is a critical parameter. To choose a proper value for the real-world experiment the simulation is run with different lengthscales to find a tradeoff between exploration and accurate performance prediction. The bounds for the Gaussian process and the chosen lengthscales are listed in Tab. 2. During the different Monte-Carlo simulation runs the system parameters are drawn from random distributions as discussed in Sec. III-C.

The resulting performance landscape using the hyperparameters listed in Tab. 1 is shown in Fig. 8. The contour lines visualize the performance landscape. The integral gain $K_{\mathrm{i,c}}$ and proportional gain $K_{\mathrm{p,c}}$ are shown on the y- and x-axis, respectively. The safe threshold and the confidence bounds are not indicated in this plot.

Using the best found controller parameters

$$K_{\mathrm{p,c}} = 0.024\,\mathrm{V/A} \text{ and } K_{\mathrm{i,c}} = 24\,\mathrm{V/(As)}$$

can increase the performance about 17 % compared to the analytical, model-based design from Sec. III-B. The current waveforms using the optimal controller parameters are shown in Fig. 9. Presented is the best result using these controller parameter set out of the n_{MC} samples. As can be seen a fast control response with appropriate small overshoot is realized.

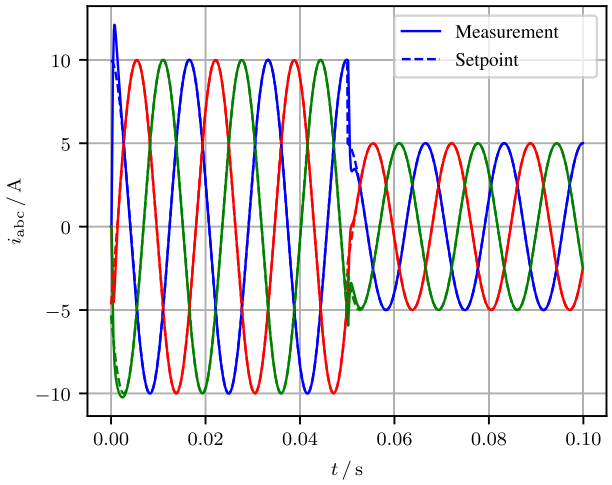

FIGURE 9. Simulated current waveforms using gain values $K_{\mathrm{p,c}} = 0.024\,\mathrm{V/A}$ and $K_{\mathrm{i,c}} = 24\,\mathrm{V/(As)}$.

All presented results of the performance landscape show the relative errors with respect to the initial performance point but using the same J_{lim} and a safe threshold of zero.

E. VOLTAGE CONTROLLER OPTIMIZATION

Now, the *SafeOpt* method, depicted in Fig. 5, is applied to find optimal parameters of the voltage controller. The system parameters from Tab. 1 are still valid but $N = 2000$ was chosen and the exemplary use case differs. An ohmic load performing two load steps

$$\boldsymbol{Z}_{\mathrm{load}} = \begin{cases} 15.4\,\Omega, & 73\,\mathrm{ms}<t<123\,\mathrm{ms}, \\ 28\,\Omega, & \text{else}, \end{cases} \tag{31}$$

is applied to test the controller performance resulting in an exemplary inductor current as shown in Fig. 10.

To show a common distribution grid-like application the voltage setpoint is chosen to

$$\boldsymbol{v}^*_{\mathrm{dq0}} = [120 \cdot \sqrt{2}\,\mathrm{V}, 0, 0]^{\mathsf{T}}. \tag{32}$$

The underlying current controller is not optimized and is fixed using the default parameters (10), but obviously the previous result from Sec. IV-B could be used as well. The developed voltage reward function is

$$r_{\mathrm{n}} = -\frac{1}{N} \sum_{p\in\{\mathrm{a,b,c}\}} \left[\sqrt{\frac{\left|v^*_{p,n} - v_{p,n}\right|}{v_{\mathrm{limit}}}} - \lambda_{\mathrm{v}} \cdot \log\left(1 - \frac{\max(|v_{p,n}| - v_{\mathrm{nom}}, 0)}{v_{\mathrm{limit}} - v_{\mathrm{nom}}}\right) \right] \tag{33}$$

for $n \in \{1, \ldots, N\}$ control steps, using $v_{\mathrm{nom}} = 190\,\mathrm{V}$ and $v_{\mathrm{limit}} = 285\,\mathrm{V}$.

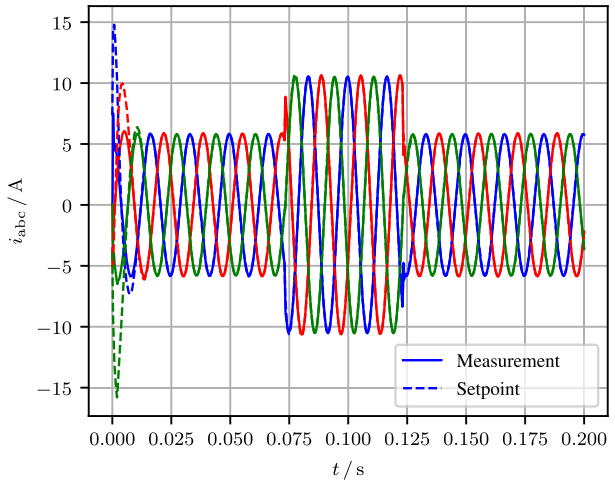

FIGURE 10. Simulated current waveforms using controller gain values $K_{p,v} = 0.022\,\mathrm{A/V}$ and $K_{i,v} = 213\,\mathrm{A/(Vs)}$.

The performance is calculated according to:

$$J = \frac{\left(\sum_{n=1}^{N} r_n + J_{\mathrm{diff}}\right) - J_{\mathrm{lim}}}{J_{\mathrm{init}} - J_{\mathrm{lim}}}. \tag{34}$$

Here, J_{diff} represents a return add-on depending on the gradient of the voltages in dq-frame to punish controller oscillation. It is added after each episode to the return given by the reward function. As can be seen in (35) J_{diff} is set to zero if the voltage differs more than $\pm 20\,\mathrm{V}$ from the setpoint. That should avoid to punish planned steps in the setpoint.

$$J_{\mathrm{diff}} = \begin{cases} -\kappa \nabla v_{\mathrm{dq0}}, & |v_{\mathrm{d}}| < 0.12 \cdot v_{\mathrm{d}}^{*} \\ 0, & \text{else.} \end{cases} \tag{35}$$

Here, ∇v_{dq0} is calculated using finite differences according to [47] and κ a weighting factor (similar to λ_{v} in the barrier function). Choosing κ and λ_{v} appropriately, the punishment can be weighted accordingly to put the focus more on avoiding overshoot or controller oscillation.

Similarly to the already mentioned system parameters, Z_{load} is sampled from a distribution using a standard deviation of $0.1 \cdot Z_{\mathrm{load}}$ and applying a clipping at $\pm 10\,\%$. This leads to a punishment for overshoots using lower controller parameters and avoiding high controller oscillations using higher controller values. The minimal allowed performance J_{lim} was calculated using (33) for worst-case scenario with a maximal phase shift of 5° and assuming an amplitude deviation of 10 % between measurement and setpoint. Again, J_{lim} is set equal to the maximum return of the two experiments. Additionally, a 1.5 kHz sine wave with $2\,\% \cdot v^{*}$ amplitude to trigger the gradient based part of the reward was added to the signal used to calculate J_{lim}. Then, (35) was taken to add a gradient depending return. The choice of J_{lim} strongly influences exploration, because it defines the safe threshold. Therefore, a more conservative J_{lim} suppresses exploration but ensures safety. Choosing a lower threshold enhances the exploration but could lead to safety problems (cf. Sec. IV-B).

TABLE 3. Hyperparameters for the voltage controller opt.

	GP lengthscale ℓ	GP boundaries $\mathcal{B}$
$K_{p,v}$	0.003 A/V	[0, 0.045] A/V
$K_{i,v}$	50 A/(Vs)	[4, 450] A/(Vs)
Barrier scaling λ_{v}	400	
Reward scaling κ	0.5	

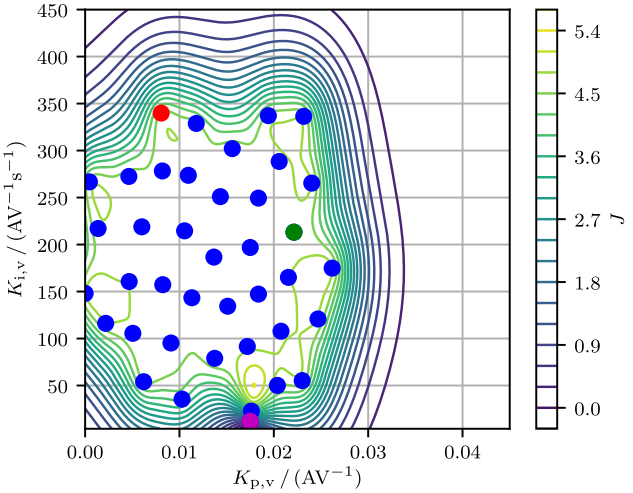

FIGURE 11. Simulated performance landscape (z-axis) for the voltage controller optimization with variable $K_{p,v}$ and $K_{i,v}$.

As discussed in Sec. III-A the load current acts as a disturbance in the voltage control loop. The OMG toolbox allows to add a disturbance observer (e.g. Luenberger observer [48]) enabling to apply feed-forward compensation [20]. However, since this did not lead to any significant positive effect on the performance of the voltage control loop in simulative and real-world pre-investigations, and since the following considerations should be kept vividly simple, a disturbance observer is not implemented.

The resulting performance landscape of the Monte-Carlo simulation is shown in Fig. 11. Here, the controller parameters (13) using the analytical layout were taken as initial parameter set and the chosen GP and reward parameters are listed in Tab. 3.

The best found controller parameters

$$K_{\mathrm{p,v}} = 0.022\,\mathrm{A/V} \text{ and } K_{\mathrm{i,v}} = 213\,\mathrm{A/(Vs)}$$

result in a performance increase of about 480 % compared to the analytical baseline from Sec. III-B. The best result for the simulated capacitor voltage waveforms of the n_{MC} runs using the upper found parameters are shown in Fig. 12. It can be seen, that the blackstart is performed fast with a small overshoot. The load steps lead to transient voltage wave form distortions, but the optimized voltage controller is able to quickly compensate for this disturbance changes.

F. MUTUAL CURRENT AND VOLTAGE CONTROLLER OPTIMIZATION

At last, the experiments from Sec. IV-B and Sec. IV-E are combined and $K_{\mathrm{p,c}}$, $K_{\mathrm{i,c}}$, $K_{\mathrm{p,v}}$ and $K_{\mathrm{i,v}}$ are optimized at the

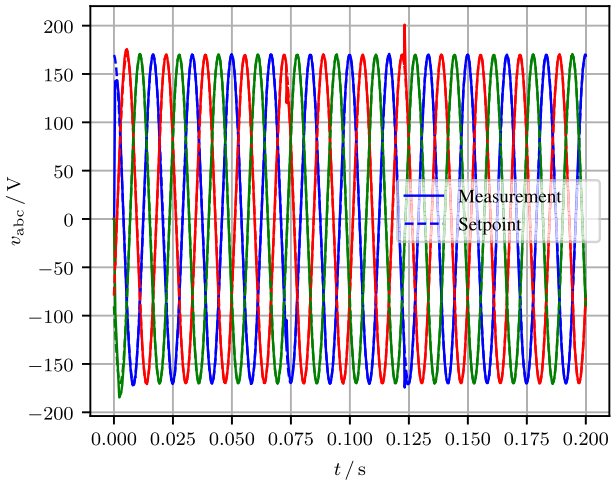

FIGURE 12. Simulated voltage waveforms using gain values $K_{p,v} = 0.022$ A/V and $K_{i,v} = 213$ A/(Vs).

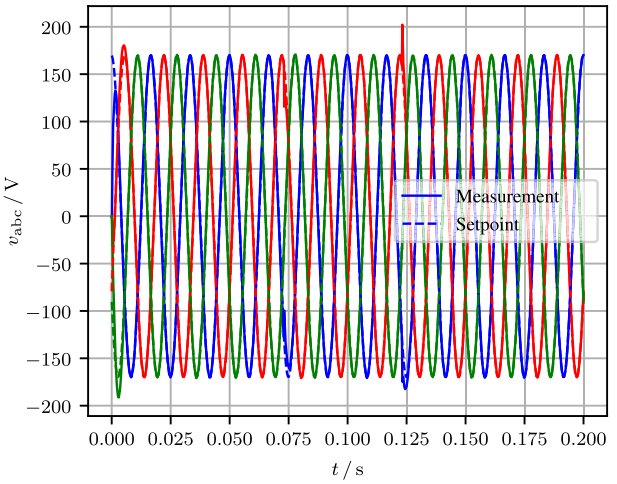

FIGURE 13. Simulated voltage waveforms using gain values $K_{p,c} = 0.04$ V/A, $K_{i,c} = 27$ V/(As), $K_{p,v} = 0.017$ A/V and $K_{i,v} = 105$, A/(Vs).

TABLE 4. Hyperparameters for the mutual controller opt.

	GP lengthscale ℓ	GP boundaries $\mathcal{B}$
$K_{p,c}$	0.012 V/A	$[0.001, 0.07]$ V/A
$K_{i,c}$	30 V/(As)	$[2, 150]$ V/(As)
$K_{p,v}$	0.003 A/V	$[0, 0.045]$ A/V
$K_{i,v}$	50 A/(Vs)	$[4, 450]$ A/(Vs)
Barrier scaling λ_c	80	
Barrier scaling λ_v	400	
Reward scaling κ	2.5	

TABLE 5. Resulting best controller parameters and corresponding performances from the simulative pre-investigation.

	Results Sec. V-B	Results Sec. V-C	Results Sec. V-D
$K_{p,c}$	0.024 V/A	0.04 V/A	0.04 V/A
$K_{i,c}$	24 V/(As)	12 V/(As)	27 V/(As)
$K_{p,v}$	-	0.022 A/V	0.016 A/V
$K_{i,v}$	-	213 A/(Vs)	105 A/(Vs)
ΔJ	17 %	480 %	26 %

same time. Therefore, the reward functions (30) and (33) are simply added together for this mutual optimization. Because of the more complex optimization problem, the number of episodes is increased to $M = 60$. Furthermore, the parameters from Tab. 1 and the application defined in (31) were used. As initial tune, the analytically calculated controller parameters were taken. The chosen GP and reward parameters are listed in Tab. 4. Due to the change in the reward function J_{lim}, λ and κ was adjusted as well, while (34) was used to calculate the performance.

The controller parameters of the best performing tune leading to a performance of $J = 1.26$ are listed in Tab. 5 in the last colon. The resulting voltage waveforms for the upper mentioned best controller tunes are shown in Fig. 13.

V. TRANSFER FROM SIMULATION TO REAL-WORLD EXPERIMENT

In this section, we introduce the transfer to a real-world experiments. Therefore, the OpenModelica part (compare Fig. 5) is replaced by physical laboratory prototype inverter. As indicated in Fig. 14, the control process is executed on a FPGA. The *SafeOpt* algorithm is executed in Python and optimizes the control parameters of the digital controller.

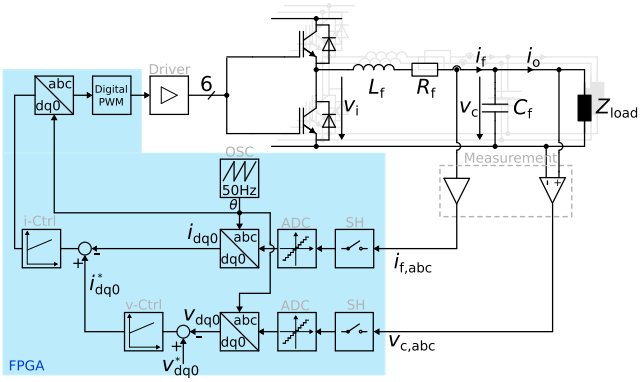

FIGURE 14. General dq-based inverter control using cascaded current and voltage control implemented on an FPGA.

A. TEST BENCH

The test bench shown in Fig. 15 consists of an Intel Cyclone®V SoC FPGA, coupling the capabilities of the FPGA with a Dual Core ARM Cortex-A9 processor. This controls a two-level inverter stack detailed in Tab. 6. The use of an AD7606 ADC ensures all analog channels are sampled at the same time and are synchronized to the pulse width modulation (PWM). The all low-level control functionalists are implemented in the FPGA, while *SafeOpt* is implemented in the ARM processor. It should also be noted that *SafeOpt* can be remotely executed on any network-connected host computer since the communication between the test bench control platform and *SafeOpt* can be executed asynchronously using standard network protocols.

B. CURRENT CONTROLLER INVESTIGATION

First, *SafeOpt* is applied to the test bench shown in Sec. V-A to optimize the parameters of the current controller.

TABLE 6. Test bench hardware.

Control platform	Terasic DE10-Nano
	ARM Cortex-A9, Intel Cyclone V FPGA
	OS: Linux Angstrom
ADC	AD7606 (23 kHz anti-aliasing filter)
IGBT	Semikron SKM50GB12T4
IGBT driver	Semikron SKH 122AR
Voltage sensor	AD AMP02 (20 kHz bandwidth)
Current sensor	LEM LAH 25-NP (transducer)
	TI TL082 (amplifier)
Bus capacitance (nominal)	470 µF
Filter inductance (nominal)	2.3 mH
Filter capacitance (nominal)	10 µF
Switching frequency	10 kHz

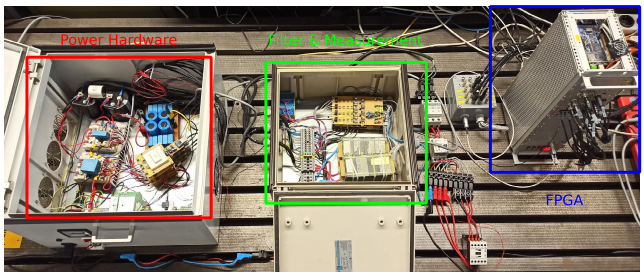

FIGURE 15. Real-world experiment setting showing power hardware consisting of IGBTs and DC-link capacitors connected to the DC-source, LC-filter and measurement going to FPGA.

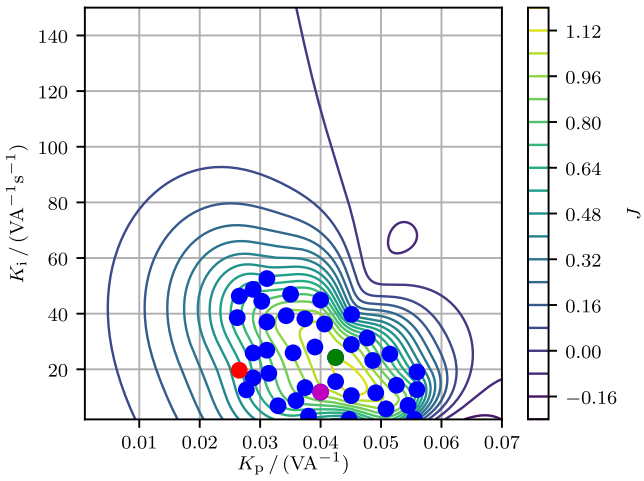

FIGURE 16. Measured performance landscape (z-axis) for the current controller optimization with variable $K_{\mathrm{p,c}}$ and $K_{\mathrm{i,c}}$.

As the initial parameter set (10) is used. The parameters for the Gaussian process, the reward from Tab. 2 and (30) as well as the performance function (30) are transferred from the simulative pre-investigations described in Sec. IV-B. The real-world device parameters from Tab. 6 correspond to the simulation model from Sec. III-A. Furthermore, the short circuit experiment is applied taking the setpoints from (28). The resulting, measured performance landscape is shown in Fig. 16 and can be compared to the one in Sec. IV-B.

Using the best controller parameters

$$K_{\mathrm{p,c}} = 0.042\,\mathrm{V/A} \text{ and } K_{\mathrm{i,c}} = 24\,\mathrm{V/(As)}$$

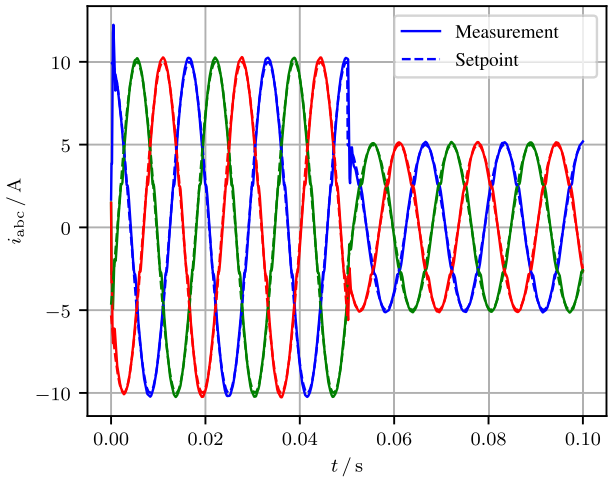

FIGURE 17. Measured current waveforms frame using gain values $K_{\mathrm{p,c}} = 0.042\,\mathrm{V/A}$ and $K_{\mathrm{i,c}} = 24\,\mathrm{V/(As)}$.

increases the performance by about 12 %. Compared to Fig. 8, it can be concluded that the simulation predicts the test bench behavior quite well by using the adjustments described in Sec. III-B.

The current waveforms using the optimal controller parameters are shown in Fig. 17 and are comparable to the simulated ones in Fig. 9. During loadsteps the overshoot behavior differs slightly caused by different controller parameters in the proportional term of the PI controller. That leads to a slightly difference in the performance result. The safe threshold was again chosen to be zero.

C. VOLTAGE CONTROLLER INVESTIGATION

In the following, the optimization is applied to tune the voltage controller at the test bench to compare the real-world to the simulation results from Sec. IV-E. The loadsteps described in (31) are realized using a relay to apply the in (32) defined voltage setpoint to the load. (33) and (34) are used as reward and performance function, respectively. The same lengthscales and boundaries for the Gaussian process and the same reward parameter from Tab. 3 were taken, besides the gradient parameter κ has been set to 5 to find a trade-off in punishing overshoots and controller oscillations. J_{lim} calculated earlier was increased to get a good trade-off between exploration and safety. As initial parameters for the voltage controller again (13) were taken while the current controller parameters from (10) were left unchanged.

The resulting performance landscape is shown in Fig. 18. It can be seen that exploration is reduced in comparison to the simulation result in Fig. 11 to ensure safety in the real-word experiment. The best voltage controller parameters are

$$K_{\mathrm{p,v}} = 0.012\,\mathrm{A/V} \text{ and } K_{\mathrm{i,v}} = 183\,\mathrm{A/(Vs)}$$

resulting in a performance increase of about 258 %.

The measured voltages using these parameters are shown in Fig. 19. In comparison to the best simulation result shown in Fig. 12, a higher voltage overshoot after the blackstart but

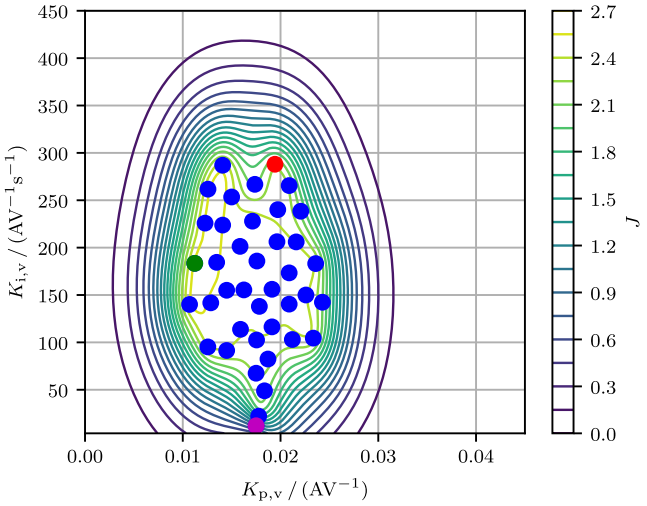

FIGURE 18. Measured performance landscape (z-axis) for the voltage controller optimization with variable $K_{\mathrm{p,v}}$ and $K_{\mathrm{i,v}}$.

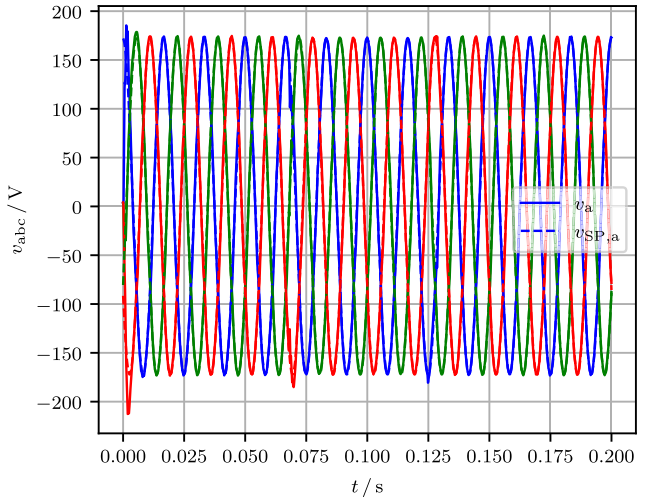

FIGURE 19. Measured voltage waveforms using gain values $K_{\mathrm{p,v}} = 0.012\,\mathrm{A/V}$ and $K_{\mathrm{i,v}} = 183\,\mathrm{A/(Vs)}$.

less overshoot during the loadsteps can be detected. Nevertheless, the control performance increase due to the application of *SafeOpt* is significant.

D. MUTUAL CURRENT AND VOLTAGE CONTROLLER INVESTIGATION

At least the in Sec.IV-F described optimization of the current and voltage controller parameters at the same time is run on the test bench using same lengthscales and boundaries for the Gaussian process and the same reward parameter from Tab. 4. As initial parameters for the voltage controller (13) and for the current controller (10) were taken. The same setting as described in Sec. IV-F was used and the best tune is listed in Tab. 7 in the left colon, reaching a performance of $J = 1.26$.

The current controller proportional gain is higher and the integral gain is lower compared to Sec. V-B. That can be explained by the modified use case. The controller parameters for the voltage controller are in the range of the in

TABLE 7. Resulting best controller parameters and corresponding performances from real-world optimization.

	Results Sec. V-B	Results Sec. V-C	Results Sec. V-D
$K_{\mathrm{p,c}}$	$0.042\,\mathrm{V/A}$	$0.04\,\mathrm{V/A}$	$0.039\,\mathrm{V/A}$
$K_{\mathrm{i,c}}$	$24\,\mathrm{V/(As)}$	$12\,\mathrm{V/(As)}$	$2.3\,\mathrm{V/(As)}$
$K_{\mathrm{p,v}}$	-	$0.012\,\mathrm{A/V}$	$0.018\,\mathrm{A/V}$
$K_{\mathrm{i,v}}$	-	$183\,\mathrm{A/(Vs)}$	$92\,\mathrm{A/(Vs)}$
ΔJ	12 %	258 %	26 %

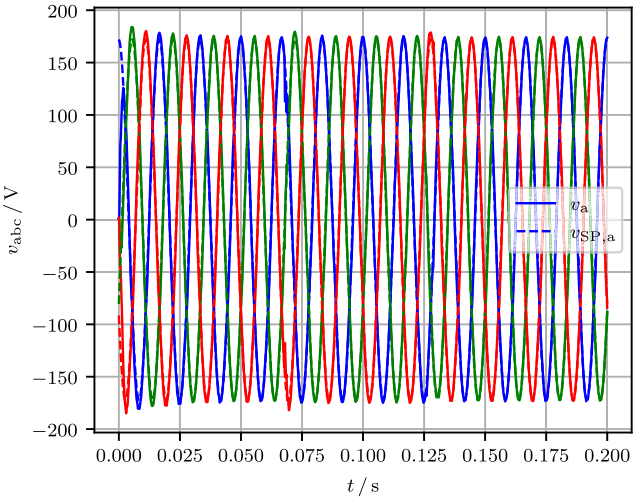

FIGURE 20. Measured voltage waveforms in abc frame using gain values $K_{\mathrm{p,c}} = 0.039\,\mathrm{V/A}$, $K_{\mathrm{i,c}} = 2.3\,\mathrm{/(As)}$, $K_{\mathrm{p,v}} = 0.018\,\mathrm{A/V}$ and $K_{\mathrm{i,v}} = 92,\,\mathrm{A/(Vs)}$.

Sec. V-C found plateau. Small differences in the optimal found parameters can be caused by different used current controller parameters or less exploration. The resulting voltage waveforms for the upper mentioned best controller tunes are shown in Fig. 20. Especially while comparing Fig. 20 with Fig. 19 – where only the voltage controller parameters are optimized – it can be seen, that the overshoot during blackstart was reduced. So, optimizing all used controller parameters at once can increase performance but is difficult due to the complex reward and optimization functions.

VI. CONCLUSION AND OUTLOOK

With the OMG toolbox, a fully open-source, scalable and flexible platform for simulation and testing of intelligent microgrid control is proposed. The toolbox fills a gap in the area of dynamic system and control analysis for inverter-driven microgrids. The core feature is a customizable interface between OpenModelica for plug and play-like system modeling and Python for the integration of arbitrary control algorithms. OMG already offers some standard controllers as well as auxiliary tools (e.g. phase-looked loops) to speed up the overall simulation and control design process for the user. In addition, the integrated OpenAI Gym interface offers a wide range of options for training and evaluating data-driven controllers from the field of reinforcement learning.

The importance of safety has already been highlighted by the data-driven optimization case study of a linear feedback controller. Although the standard voltage source inverter

control framework is heavily based on expert knowledge, its data-driven optimization is associated with the risk of creating unsafe system states, potentially leading to system malfunctions or damages. Safe Bayesian optimization could only prevent this to a limited extent, because its abstract uncertainty evaluation based on Gaussian processes cannot provide a fully reliable safety prediction. It also comes with several important but unintuitive hyperparameters which configuration has major impact on the exploration behavior. Only extensive, simulative pre-investigations using OMG and an accompanying empirical optimization of the hyperparameters could enable a safe transfer to real-world experiments.

Safe, data-driven control of microgrids remains an exciting research challenge for future work. The integration of a priori expert knowledge for the evaluation of safe control methods appears to be promising, e.g. to monitor and guide the training of exploring, self-adapting control methods like from the field of reinforcement learning. In addition, OMG's functionality is continuously being extended, with a current focus on the development of abstracted, dynamic models (e.g. swing equation approach). This will make it possible to investigate larger MSG topologies as well as simulate longer time spans with acceptable numerical effort, which is necessary for evaluating higher level MSG control strategies.

ACKNOWLEDGMENT

The authors would like to thank Andreas Heuermann from Linköping University for support on OpenModelica and FMU integration issues.

REFERENCES

[1] United Nations. *Sustainable Development Goals*. Accessed: 2020. [Online]. Available: https://sustainabledevelopment.un.org/?menu=1300

[2] H. Lund, P. A. Østergaard, D. Connolly, and B. V. Mathiesen, "Smart energy and smart energy systems," *Energy*, vol. 137, pp. 556–565, Oct. 2017.

[3] N. Hatziargyriou, H. Asano, R. Iravani, and C. Marnay, "Microgrids," *IEEE Power Energy Mag.*, vol. 5, no. 4, pp. 78–94, Jul./Aug. 2007.

[4] B. Kroposki, R. Lasseter, T. Ise, S. Morozumi, S. Papathanassiou, and N. Hatziargyriou, "Making microgrids work," *IEEE Power Energy Mag.*, vol. 6, no. 3, pp. 40–53, May 2008.

[5] F. Katiraei and M. R. Iravani, "Power management strategies for a microgrid with multiple distributed generation units," *IEEE Trans. Power Syst.*, vol. 21, no. 4, pp. 1821–1831, Nov. 2006.

[6] J. M. Guerrero, J. C. Vasquez, J. Matas, L. García de Vicuna, and M. Castilla, "Hierarchical control of droop-controlled AC and DC microgrids—A general approach toward standardization," *IEEE Trans. Ind. Electron.*, vol. 58, no. 1, pp. 158–172, Jan. 2011.

[7] E. Hossain, E. Kabalci, R. Bayindir, and R. Perez, "Microgrid testbeds around the world: State of art," *Energy Convers. Manage.*, vol. 86, pp. 132–153, Oct. 2014.

[8] N. W. A. Lidula and A. D. Rajapakse, "Microgrids research: A review of experimental microgrids and test systems," *Renew. Sustain. Energy Rev.*, vol. 15, no. 1, pp. 186–202, Jan. 2011.

[9] J. M. Guerrero, M. Chandorkar, T.-L. Lee, and P. C. Loh, "Advanced control architectures for intelligent microgrids—Part I: Decentralized and hierarchical control," *IEEE Trans. Ind. Electron.*, vol. 60, no. 4, pp. 1254–1262, Apr. 2013.

[10] J. M. Guerrero, P. C. Loh, T.-L. Lee, and M. Chandorkar, "Advanced control architectures for intelligent microgrids—Part II: Power quality, energy storage, and AC/DC microgrids," *IEEE Trans. Ind. Electron.*, vol. 60, no. 4, pp. 1263–1270, Apr. 2013.

[11] D. E. Olivares, A. Mehrizi-Sani, A. H. Etemadi, C. A. Cañizares, R. Iravani, M. Kazerani, A. H. Hajimiragha, O. Gomis-Bellmunt, M. Saeedifard, R. Palma-Behnke, G. A. Jiménez-Estévez, and N. D. Hatziargyriou, "Trends in microgrid control," *IEEE Trans. Smart Grid*, vol. 5, no. 4, pp. 1905–1919, Jul. 2014.

[12] Y. A.-R. I. Mohamed and E. F. El-Saadany, "Adaptive decentralized droop controller to preserve power sharing stability of paralleled inverters in distributed generation microgrids," *IEEE Trans. Power Electron.*, vol. 23, no. 6, pp. 2806–2816, Nov. 2008.

[13] M. Armin, P. N. Roy, S. K. Sarkar, and S. K. Das, "LMI-based robust PID controller design for voltage control of islanded microgrid," *Asian J. Control*, vol. 20, no. 5, pp. 2014–2025, Sep. 2018.

[14] N. Liu, Q. Chen, J. Liu, X. Lu, P. Li, J. Lei, and J. Zhang, "A heuristic operation strategy for commercial building microgrids containing EVs and PV system," *IEEE Trans. Ind. Electron.*, vol. 62, no. 4, pp. 2560–2570, Apr. 2015.

[15] B. Khan and P. Singh, "Selecting a meta-heuristic technique for smart micro-grid optimization problem: A comprehensive analysis," *IEEE Access*, vol. 5, pp. 13951–13977, 2017.

[16] I. Prodan and E. Zio, "A model predictive control framework for reliable microgrid energy management," *Int. J. Electr. Power Energy Syst.*, vol. 61, pp. 399–409, Oct. 2014.

[17] A. Parisio, E. Rikos, and L. Glielmo, "A model predictive control approach to microgrid operation optimization," *IEEE Trans. Control Syst. Technol.*, vol. 22, no. 5, pp. 1813–1827, Sep. 2014.

[18] F.-D. Li, M. Wu, Y. He, and X. Chen, "Optimal control in microgrid using multi-agent reinforcement learning," *ISA Trans.*, vol. 51, no. 6, pp. 743–751, Nov. 2012.

[19] M. Adibi and J. V. D. Woude, "A reinforcement learning approach for frequency control of inverted-based microgrids," *IFAC-PapersOnLine*, vol. 52, no. 4, pp. 111–116, 2019.

[20] H. Bode, S. Heid, D. Weber, and O. Wallscheid. *OpenModelica Microgrid Gym (OMG)*. Accessed: 2020. [Online]. Available: https://github.com/upb-lea/openmodelica-microgrid-gym

[21] 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," *J. Open Source Softw.*, vol. 5, no. 54, p. 2435, Oct. 2020.

[22] Open Source Modelica Consortium (OSMC). *OpenModelica*. Accessed: 2020. [Online]. Available: https://www.openmodelica.org/

[23] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," 2016, *arXiv:1606.01540*. [Online]. Available: http://arxiv.org/abs/1606.01540

[24] F. Guo, C. Wen, J. Mao, and Y.-D. Song, "Distributed secondary voltage and frequency restoration control of droop-controlled inverter-based microgrids," *IEEE Trans. Ind. Electron.*, vol. 62, no. 7, pp. 4355–4364, Jul. 2015.

[25] N. Soni, S. Doolla, and M. C. Chandorkar, "Inertia design methods for islanded microgrids having static and rotating energy sources," *IEEE Trans. Ind. Appl.*, vol. 52, no. 6, pp. 5165–5174, Dec. 2016.

[26] F. Berkenkamp. *SafeOpt: Safe Bayesian Optimization*. Accessed: 2020. [Online]. Available: https://github.com/befelix/SafeOpt

[27] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, Feb. 2011.

[28] L. Thurner, A. Scheidler, F. Schäfer, J.-H. Menke, J. Dollichon, F. Meier, S. Meinecke, and M. Braun, "Pandapower—An open-source Python tool for convenient modeling, analysis, and optimization of electric power systems," *IEEE Trans. Power Syst.*, vol. 33, no. 6, pp. 6510–6521, Nov. 2018.

[29] T. Brown, J. Hörsch, and D. Schlachtberger, "PyPSA: Python for power system analysis," *J. Open Res. Softw.*, vol. 6, no. 4, pp. 1–10, Jan. 2018.

[30] F. Milano, L. Vanfretti, and J. C. Morataya, "An open source power system virtual laboratory: The PSAT case and experience," *IEEE Trans. Educ.*, vol. 51, no. 1, pp. 17–23, Feb. 2008.

[31] S. Balderrana and S. Quoilin. *Micro-Grids*. Accessed: 2020. [Online]. Available: https://github.com/squoilin/MicroGrids

[32] Mathworks. *Simscape*. Accessed: 2020. [Online]. Available: https://www.mathworks.com/products/simscape.html

[33] Mathworks. *MATLAB Engine API for Python*. Accessed: 2020. [Online]. Available: https://www.mathworks.com/help/matlab/matlab-engine-for-python.html

[34] A. Kuhnle, M. Schaarschmidt, and K. Fricke. *TensorForce: A TensorFlow Library for Applied Reinforcement Learning*. Accessed: 2017. [Online]. Available: https://github.com/tensorforce/tensorforce
[35] Modelica Association. *Functional Mock-Up Interface Standard*. Accessed: Feb. 24, 2021. [Online]. Available: https://fmi-standard.org/
[36] The SciPy Community. *Scipy.Integrate.Solve_ivp*. Accessed: Feb. 24, 2021. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/tutorial/integrate.html
[37] Modelon. *Pyfmi*. Accessed: Feb. 24, 2021. [Online]. Available: https://pypi.org/project/PyFMI/
[38] H. Bode, S. Heid, D. Weber, and O. Wallscheid. *OpenModelica Microgrids*. Accessed: 2020. [Online]. Available: https://github.com/upb-lea/OpenModelica_Microgrids
[39] S. Buso and P. Mattavelli, *Digital Control in Power Electronics*. San Rafael, CA, USA: Morgan & Claypool, 2006.
[40] A. Keyhani, M. Marwali, and M. Dai, *Integration of Green and Renewable Energy in Electric Power Systems*. Hoboken, NJ, USA: Wiley, 2009.
[41] M. S. Sadabadi, Q. Shafiee, and A. Karimi, "Plug-and-play voltage stabilization in inverter-interfaced microgrids via a robust control strategy," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 3, pp. 781–791, May 2017.
[42] S. Riverso, F. Sarzo, and G. Ferrari-Trecate, "Plug-and-play voltage and frequency control of islanded microgrids with meshed topology," *IEEE Trans. Smart Grid*, vol. 6, no. 3, pp. 1176–1184, May 2015.
[43] J. W. 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?" *IEEE Trans. Ind. Appl.*, vol. 26, no. 3, pp. 489–497, May 1990.
[44] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proc. IEEE*, vol. 104, no. 1, pp. 148–175, Jan. 2016.
[45] P. Auer, "Using confidence bounds for exploitation-exploration tradeoffs," *J. Mach. Learn. Res.*, vol. 3, pp. 397–422, Nov. 2003.
[46] C. Rasmussen, *Gaussian Processes for Machine Learning*. Berlin, Germany: Springer, 2009.
[47] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, and R. Kern, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020, doi: 10.1038/s41586-020-2649-2.
[48] G. Ellis, *Observers in Control Systems: A Practical Guide*. Amsterdam, The Netherlands: Elsevier, 2002.

DANIEL WEBER was born in Berlin, Germany, in 1988. He received the B.Sc. and M.Sc. degrees in electrical engineering from Paderborn University, Germany, in 2013 and 2016, respectively.

Since 2016, he has been a Research Assistant with the Chair of Power Electronics and Electrical Drives, Paderborn University. His research interests include the data-driven control approaches in the context of microgrids and grid control.

STEFAN HEID was born in Hammelburg, Germany, in 1992. He received the B.Sc. and M.Sc. degrees in computer science from Paderborn University, Germany, in 2016 and 2020, respectively.

Since 2020, he has been a Research Assistant with the Chair of Intelligent Systems and Machine Learning, Paderborn University. His research interest includes reinforcement learning with a special focus on safe control.

HENRIK BODE was born in Herford, Germany, in 1993. He received the B.Sc. and M.Sc. degrees in industrial and electrical engineering from Paderborn University, Germany, in 2018.

From 2019 to 2020, he was a Scientific Assistant with the Chair of Power Electronics and Electrical Drives, Paderborn University. Since 2020, he has been a Research Assistant with the Chair of Power Electronics and Electrical Drives and the Chair of Teaching Technics, Paderborn University. His research interests include the simulation efficient and robust microgrids as well as didactical aspects to teach students and pupil the technical background behind energy grids.

JARREN H. LANGE was born in Johannesburg, Gauteng, South Africa, in 1992. He received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from the University of the Witwatersrand, Johannesburg, in 2015 and 2019, respectively.

Since 2020, he has been a Research Associate with the Chair of Power Electronics and Electrical Drives, Paderborn University, Germany. His research interests include the development of microgrids with the focus of decentralized control techniques and renewable power system technologies and techniques.

EYKE HÜLLERMEIER (Senior Member, IEEE) received the Ph.D. degree in 1997 and the Habilitation degree in 2002. He is currently a Professor with the Department of Computer Science, Paderborn University, where he heads the Intelligent Systems and Machine Learning Group, a member of the Heinz Nixdorf Institute, and a Director of the Software Innovation Campus Paderborn. Prior to joining Paderborn University in 2014, he held professorships at the Universities of Dortmund, Magdeburg and Marburg.

OLIVER WALLSCHEID (Member, IEEE) received the bachelor's and master's degrees (Hons.) in industrial engineering and the Ph.D. degree (Hons.) in electrical engineering from Paderborn University, Germany, in 2010, 2012, and 2017, respectively. Since then, he has been worked as a Senior Research Fellow with the Department of Power Electronics and Electrical Drives, Paderborn University. He is currently serving as an Acting Professor for the Automatic Control Department, Paderborn University. His research interests include system identification and intelligent control methods for technical systems in the areas of power electronics, drives, and decentralized grids. Recent work focuses on hybrid methods utilizing machine learning (black box approaches) and classical engineering techniques (white box approaches).