QOPT: An Experiment-Oriented Software Package for Qubit Simulation and Quantum Optimal Control

Julian David Teske,* Pascal Cerfontaine, and Hendrik Bluhm†
JARA-FIT Institute for Quantum Information, Forschungszentrum Jülich GmbH and RWTH Aachen University, Aachen 52074, Germany

(Received 13 October 2021; revised 17 January 2022; accepted 2 February 2022; published 14 March 2022)

Realistic modeling of qubit systems including noise and constraints imposed by control hardware is required for performance prediction and control optimization of quantum processors. We introduce QOPT, a software framework for simulating qubit dynamics and robust quantum optimal control considering common experimental situations. To this end, we model open and closed qubit systems with a focus on the simulation of realistic noise characteristics and experimental constraints. Specifically, the influence of noise can be calculated using Monte Carlo methods, effective master equations or with the efficient filter function formalism, which enables the investigation and mitigation of autocorrelated noise. In addition, limitations of control electronics, including finite bandwidth effects as well as nonlinear transfer functions and drive-dependent noise, can be considered. The calculation of gradients based on analytic results is implemented to facilitate the efficient optimization of control pulses. The software easily interfaces with QUTIP, is published under an open-source license, is well tested, and features a detailed documentation.

DOI: 10.1103/PhysRevApplied.17.034036

I. INTRODUCTION

A central challenge for the realization of a universal quantum computer is the loss of entanglement and coherence due to the detrimental effects of uncontrolled interactions between the system storing quantum information and its environment [1]. To understand the underlying error processes and predict the performance of quantum processor designs, a realistic model for the manipulation by control hardware and the noise introduced by the environment is required. Such a model is also essential for optimizing the performance, for example using quantum optimal control (QOC) techniques, which adapt control pulses such that a suitably chosen metric is maximized [2,3]. QOC methods will be essential to achieve the best possible gate accuracy for quantum devices, both for noisy intermediate-scale systems and future universal quantum computers [4–9]. For increasing the number of qubits in a quantum processing unit, QOC techniques will likely be needed to address effects like crosstalk of control fields, frequency crowding, and unintended coupling between qubits. Furthermore, a well-founded performance assessment for benchmarking, platform selection and system design is only possible if the best possible control approaches are considered.

In a physical system, the time-dependent control fields implementing a desired quantum gate, which we refer to as control pulses, are applied to a qubit using classical control hardware (such as arbitrary waveform generators or lasers). In numerical QOC, a qubit's evolution is simulated for a given control pulse and then optimized prior to the application to the experiment. To achieve the best possible qubit performance, this optimization requires a model of the whole system, including any effects that influence its evolution. These factors can include the intrinsic properties of the qubits themselves, their coupling, the unintended interaction with the environment [10], as well as their response to and properties of the control hardware. Incorporating the control hardware into the models is also essential for the design of tailored control electronics for quantum computation (e.g., cryolectronic systems operated in close vicinity of the qubits to achieve a high wiring density). While providing at least the minimally required control capabilities, additional constraints like a reduced heat dissipation compatible with cryogenic cooling need to be considered [11,12] for such systems.

Even if the models used for pulse optimization are not sufficiently accurate for the direct ("open-loop") experimental application of the results, they can be a useful starting point for further fine-tuning in a closed loop with feedback from an actual experiment [13–15]. Even then, they should behave as similarly as possible to the physical system and at least qualitatively capture all relevant effects. Simulation frameworks are also helpful to simulate and develop such experimental tuning procedures [15–17], or they can be combined with the characterization of a qubit in an optimization loop [18].

*j.teske@fz-juelich.de
†bluhm@physik.rwth-aachen.de

The desire to achieve realistic models leads to a number of required features. The signal transduction from the hardware to the qubit must be modeled including all relevant technological limitations, such as the finite bandwidth of arbitrary waveform generators [19] and signal pathways, that affect the implemented gate. These limitations can also cause gate bleedthrough [5] and crosstalk between different control signals. Furthermore, nonlinear effects may arise, either due to the control hardware itself, or because of a nonlinear relation between the physical control fields and the effective control Hamiltonian (e.g., due to a truncation of the Hilbert space involving a Schrieffer-Wolf transformation). Regarding decoherence effects, it is important to realistically consider all relevant noise sources and their properties. An essential aspect is correlated noise, which can be incorporated by nontrivial spectral noise densities [10]. Correlated noise is a limiting factor for high-fidelity gates in many systems [16,20,21], but correlations can also be exploited to reduce decoherence effects in dynamically corrected gates. The performance quantification of quantum operations can be extended to such a mitigation of noise effects, often termed robust optimal control [10]. In the case of nonlinear relations between control fields and qubit Hamiltonian, the effect of noise in the control field automatically depends on the control field, thus representing drive-dependent noise.

From a performance point of view, it is important to be able to use efficient algorithms that are well suited for the problem at hand. Many algorithms for the numerical open-loop optimization discretize control pulses in time to yield a finite-dimensional parameter space. The discrete elements of the control pulses can then be updated simultaneously using gradient ascent methods such as GRAPE [22] or subsequently with Krotov's method [23] as well as gradient-free methods [24]. More advanced gradient-based algorithms include the use of second-order derivatives [25], gradient optimization of analytical controls (GOAT) [26], and the application of the Kalman filter for the estimation of gradients [27]. For a closed-loop optimization that includes direct feedback from an experiment, an approach is to parameterize control pulses in terms of a randomly chosen subspace using CRAB [28] or the remote version, RedCRAB [29]. A methodology combining open- and closed-loop optimization is the C3 tool set for integrated control, calibration, and characterization [18].

To ease the application of advanced models and algorithms, general-purpose, flexible and easily usable software implementations are highly advantageous. An early example is the unifying algorithmic framework DYNAMO [30], which implements GRAPE and Krotov's method in MATLAB and inspired the implementation of an optimal control package in the Quantum Toolbox in PYTHON (QUTIP) [31], an open-source program for the simulation of open quantum systems. QUTIP's extension by the subpackage QUTIP-QIP for quantum information processing introduces the capability to compile circuits into control pulses and to simulate them on the pulse level with the option to include noise models [32]. QUTIP's subpackages for optimal control and quantum information processing can also be combined to employ optimized pulses in the circuit simulation. An additional package (KROTOV) builds Krotov's method on top of QUTIP [33]. There are also special-purpose optimization frameworks like QEngine [34] for ultracold atoms or Pulser [35] for neutral-atom arrays. Some of these implementations can be generalized to noisy systems if an open system description based on master equations is adopted [36], thus readily treating Markovian noise. One possibility do deal with non-Markovian noise is the use of ancillary qubits [10,37], which, however, is computationally very costly as it substantially increases the Hilbert space dimension.

While these simulation frameworks are widely and successfully used in their respective domains, we found them to be less suited and difficult to extend to address the above requirements for the realistic, hardware aware simulation. We thus implemented the PYTHON package QOPT [38] featuring transfer functions and realistic noise models to describe noise of arbitrary correlations and spectral noise densities. The QOPT package offers a compelling advantage over other packages if a simulation or a pulse optimization is required to include hardware constraints or noise of arbitrary spectra.

The development was in many ways inspired by QUTIP's optimal control subpackage, but has in some aspects a different structure. Specifically, the performance of sequential optimization algorithms like Krotov's method is based on the possibility of efficiently updating pulses independently in each time-step, which is incompatible with generally parameterized pulses or transfer functions because these introduce relations of pulse values at different time-steps.

Concurrent with our work, the startup Q-CTRL developed software with similar methods but pursued a different strategy and targeted a commercial audience [39]. As for QOPT, these methods include generalized filter functions and the simulation of noise by explicitly sampling noise distributions as Monte Carlo simulations. The imperfections of control hardware are modeled by transfer functions. Additionally, Q-CTRL provides methods for the noise characterization of a given system. While one may expect that the commercial multipurpose software of Q-CTRL leads to a feature-rich and easy to use solution, the closed-source approach reduces the transparency and flexibility, which is often important for research use. As a GPL3-licensed open-source package, QOPT complements this approach, targeting mainly the scientific community in academia and industry. Besides low barriers to entry, the modular structure and complete API documentation [40] provide full flexibility in the implementation of techniques that expand the application to unsolved problems. The user

can supply their own optimization algorithm or replace any other relevant part. Multiprocessing is also supported.

This paper gives an overview of QOPT's capabilities, while full documentation and numerous introductory examples can be found online on readthedocs [40]. Section II describes the mathematical formulation used for the experimentally oriented simulation of qubits and the application in QOC. The actual implementation of the QOPT package is described in Sec. III, including a practical example. Finally, an outlook is given in Sec. IV.

II. PROBLEM FORMULATION AND SIMULATION METHODS

A. Problem formulation

A rather general model for a driven qubit system subject to (classical) noise can be described by a Hamiltonian of the form

$$H(t) = H_c(t) + H_d(t) + H_n(t), \tag{1}$$

$$H_c(t) = \sum_k u_k(t) B_k, \tag{2}$$

$$H_n(t) = \sum_k b_k(t) s_k(t) C_k, \tag{3}$$

with the control Hamiltonian H_c, the drift Hamiltonian H_d, and the noise Hamiltonian H_n. The control Hamiltonian models the manipulation of the system with time-dependent control amplitudes u_k and Hermitian operators B_k and C_k. The drift Hamiltonian H_d incorporates any effects that cannot be freely controlled but still affect the dynamics. It describes the natural evolution of the system in absence of any control (e.g., due to a fixed energy splitting). The noise Hamiltonian H_n models unintentional interactions of the system with the control hardware or the environment, like electrical noise on the control amplitudes or the interaction with electromagnetic fields from the host material. The noise amplitudes b_k describe the strength of the perturbation while the noise susceptibilities s_k describe the coupling strength to the noise source. Unlike b_k, s_k can depend on the control amplitudes u_k. This means we can describe the noise by control-independent stochastic variables while still allowing for control-dependent coupling strengths to the qubit system. We treat the interaction of classical noise sources with the qubit system in a semiclassical description, where the noise enters Schrödinger's equation as stochastic variable $b_k(t)$. We assume stochastic variables with zero mean and (wide-sense) stationary distributions. Quantum noise is defined as a noise source with quantized degrees of freedom and can thus be integrated by appropriate choice of the Hamiltonian. Auto'correlated classical noise is characterized (up to higher-order correlations for non-Gaussian processes) by its spectral density $S_k(\omega)$ defined as the Fourier transform of the correlator $\langle b_k(t_1) b_k(t_2)\rangle = (1/2\pi)\int_{-\infty}^{\infty} S_k(\omega) e^{-i\omega(t_1-t_2)} d\omega$.

In many experiments the control amplitudes $u_k(t)$ in the Hamiltonian from Eq. (2) are not directly controllable by the experimenter. Rather, they are functions of physical control fields $v_i(t)$. We call the mapping of v_i to u_k the *amplitude function* (see Fig. 1). An example is the control by Rabi driving where the control amplitude $v_1(t) = A(t)$ and phase offset $v_2(t) = \delta(t)$ of a control signal appear in the Hamiltonian as $u_1(t) = A(t)\sin[\omega t + \delta(t)]$. Nonlinear amplitude functions can also arise from the truncation of Hilbert space. For example, the exchange interaction of two electron spins depends nonlinearly on the detuning between different orbital states of the electrons, when the orbital states are truncated.

Furthermore, imperfections of the control electronics can be modeled by the use of linear transfer functions [19] acting on the controllable optimization parameters v_k. This can be done by oversampling and smoothing the control pulse for example, by convolution

$$v(t) = \int_{-t_0}^{t_0} \tilde{v}(t-\tau) K(\tau) d\tau, \tag{4}$$

$$K(\tau) = \frac{\Theta(\tau)}{\tau_0} e^{-\tau/\tau_0}, \tag{5}$$

with an RC filter kernel K with time constant $\tau_0 = RC$. Θ denotes the Heaviside step function and the boundaries in the convolution truncate the kernel to the domain $[-t_0, t_0]$. Alternatively, kernels corresponding to the dispersion of coaxial cables, a Gaussian filter or a realistic transfer function, measured for a specific setup, can be used. Our implementation also allows the user to add boundary conditions by padding the beginning and end of each pulse with appropriate values. A common use case is an additional idle time at the end of each pulse in order to avoid transients across pulses [16]. Note that the amplitude and transfer functions have similar roles, but different constraints. The amplitude function can be nonlinear but must be local in time, whereas the transfer function must be linear but can be nonlocal in time. The transfer function is applied before the amplitude function.

B. Noise simulation

To solve the model formulated in the previous section, the system dynamics in the presence of noise must be computed. Depending on the properties of the noise spectrum, in particular its frequency dependence and strength, different methods with different computation costs can be used. This subsection presents the mathematical formulation of the methods provided by QOPT.

The simulation methods currently implemented in QOPT fall into three categories. The simulation can be based on the explicit generation of noise traces for Monte Carlo

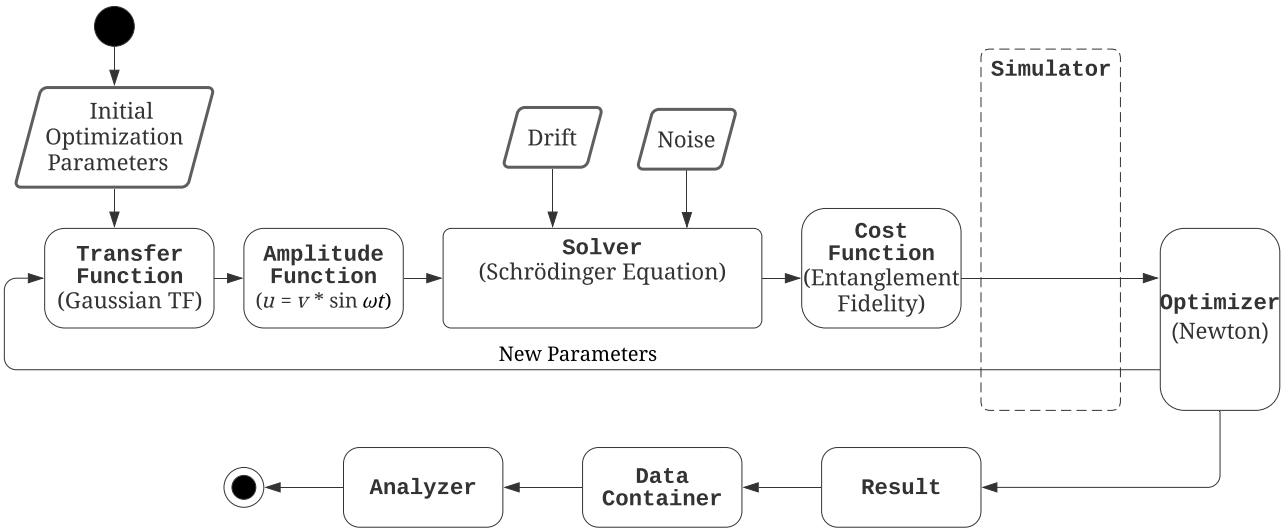

FIG. 1. Activity diagram of a pulse optimization with QOPT, where activities are identified by the active classes. The solid and encircled dots show the start and end point, respectively. The arrows mark the flow of data and the rounded rectangles the classes of the QOPT package with base class names written in bold, while the parallelograms contain direct input from the model. Concrete examples or explanations are given in the brackets below the class name (i.e., the example for a transfer function is the convolution with a Gaussian kernel). The box of the simulator is dashed because it defines an interface between the simulation and the optimization. The Optimizer obtains the cost function evaluations, and in the case of gradient-based optimization also the derivatives of the cost functions from the simulator, which sets the control parameters and initiates the simulation and subsequent evaluation of the cost functions.

methods, using a master equation to describe the qubits as part of an open quantum system, or the filter function formalism can be applied.

1. Monte Carlo

Explicitly generating numerous noise traces $b_k(t)$ whose Fourier transform converges (on average) to the noise spectral density S is one of the simplest methods but does not require any assumptions about the noise spectrum. If the main noise contribution occurs at frequencies much lower than the simulation dynamics, it is sufficient to consider the noise amplitude to be static during the pulse. For few quasistatic noise sources, explicit numerical integration of the (typically Gaussian) probability distribution of these noise values is more efficient than Monte Carlo sampling in small dimensions [41]. The user can choose between both methods in QOPT.

In this approach the highest relevant noise frequency sets the required time-step for the numerical integration, so that additional oversampling is required if it exceeds the bandwidth of the pulse itself. Since the numerical complexity of the simulation grows proportionally with oversampling, such a Monte Carlo approach becomes computationally inefficient if the noise spectral density cannot be neglected at frequencies much higher than the bandwidth frequency of the control electronics. Even if this is not an issue, many repetitions are required to gather statistics.

2. Lindblad master equation

If high noise frequencies are relevant, more efficient methods than Monte Carlo are available. In the special case of Markovian (uncorrelated) noise with a constant spectral density in the region of interest, the influence on the qubit system can be described by an effective master equation in Lindblad form [42]. In this master equation, the von Neumann equation is complemented by a dissipation term leading to nonunitary dynamics. The Lindblad form is

$$\dot{\rho} = -\frac{i}{\hbar}[H,\rho] + \sum_n \gamma_n \left(L_n \rho L_n^\dagger - \frac{1}{2}\{L_n^\dagger L_n, \rho\} \right), \quad (6)$$

where the Lindblad operators L_n describe dissipation effects and can themselves depend on the control amplitudes. The coefficients γ_n can be chosen to depend on the control parameters. If, for example, the Hamiltonian is of the form

$$H = u[v(t)]B + \delta v \frac{\partial u}{\partial v} B \quad (7)$$

with some controllable parameter v (e.g., a voltage) and perturbations $b = \delta v$ to this parameter, then the susceptibility to perturbations is the derivative $s = \partial u/\partial v$ and the coupling Hermitian operator B is the same for both terms. If the fluctuations δv are Markovian then we can translate this description directly into a master equation in Lindblad form with coefficient $\gamma = S_v \|\partial u/\partial v\|^2$ with the power spectral density S_v and Lindblad operator $L = B$ [43].

The master equation can be reformulated as a linear system of equations with the Kronecker matrix product $\otimes$ and calculated as a matrix exponential:

$$\mathrm{vec}(\rho)(t) = \exp[(-i\mathcal{H} + \mathcal{G})t]\ \mathrm{vec}(\rho)(0), \tag{8}$$

$$\mathcal{H} = I \otimes H - H^T \otimes I, \tag{9}$$

$$\mathcal{G} = \sum_{k=0}^{K} \mathcal{D}(L_k), \tag{10}$$

$$\mathcal{D}(L) = L^* \otimes L - \frac{1}{2} I \otimes (L^\dagger L) - \frac{1}{2}(L^T L^*) \otimes I, \tag{11}$$

where $\mathrm{vec}(\rho)$ denotes the density matrix written as a vector in columnwise order.

The derivation of a master equation in Lindblad form requires the assumption of Markovian (uncorrelated) noise. This approximation does not hold for many experimentally relevant noise sources such as flux noise in superconducting qubits and charge noise in many types of solid state qubits, which typically have a $1/f$ -like spectrum.

3. Filter function formalism

The filter function formalism provides a mathematical tool which can (perturbatively) model the decoherence caused by arbitrary classical noise [44]. The derivation of the filter function formalism only makes the assumption of classical noise with small amplitudes. Both master equation and filter functions have already been employed for numerical pulse optimization [17,45].

In the filter function formalism, a so-called filter function F_α can be calculated for each noise source α given the evolution of the system. For a given control pulse, F_α captures the susceptibility of the resulting quantum channel to the noisy quantity as a function of frequency. The noise contribution to the entanglement infidelity or other figures of merit can be calculated as the integral of a filter function and the spectral noise density,

$$\mathcal{I}_{\mathrm{FF}} = \int_{-\infty}^{\infty} \frac{d\omega}{2\pi} S_\alpha(\omega) F_\alpha(\omega). \tag{12}$$

This expression can be used as fidelity measure as explained in Sec. II C. Other figures of merit like the contribution to leakage can also be calculated within this formalism [46].

The numerical routines for the calculation of filter functions and their derivatives with respect to the control amplitudes $[\partial F_\alpha / \partial u_k(t)]$ are provided by the software package FILTER_FUNCTIONS [47]. The calculations within the filter function formalism outperform Monte Carlo simulations for small systems consisting of a few qubits, while Monte Carlo methods scale better with an increasing number of qubits [46,48,49] and are usually more efficient in the treatment of quasistatic noise. For the description of Markovian noise, a master equation is typically more efficient.

C. Fidelity measures

Fidelity measures are important as figures of merit for qubit performance as well as optimization targets for QOC procedures aiming to maximize performance, as discussed in II D. QOPT implements various fidelity measures to quantify the accuracy of quantum operations adapted to specific simulated experiments and the noise simulation methods applied.

For state transfers, the state fidelity between the initial and final quantum states described by the density matrices ρ_1 and ρ_2 can be used, which is defined as $\mathcal{F}_{\mathrm{st}}(\rho_1, \rho_2) = \left[\mathrm{tr}\left(\sqrt{\sqrt{\rho_1}\rho_2\sqrt{\rho_1}}\right)\right]^2$. One commonly used measure for the closeness of two quantum gates is the entanglement infidelity $\mathcal{I}_e(V, U)$, which is equivalent to the average gate fidelity [50]. If both the simulated propagator U and the target gate V are unitary, the entanglement fidelity $\mathcal{F}_e = 1 - \mathcal{I}_e$ is given by the Hilbert-Schmidt norm as

$$\mathcal{F}_e(V, U) = \frac{1}{d^2} |\mathrm{tr}(V^\dagger U)|^2. \tag{13}$$

The entanglement fidelity can also be generalized for open quantum systems [51] and we calculate it as

$$\mathcal{F}_o(V, P_s) = \frac{1}{d^2} \mathrm{tr}\left[\left(V^T \otimes V^\dagger\right) P_s\right], \tag{14}$$

where P_s is the simulated propagator of the noisy quantum process given by the matrix exponential of Eq. (8).

Leakage occurs in a quantum processor if states outside the computational subspace are populated. To simulate this error source, we extend the Hilbert space of computational states $\mathcal{H}_c$ as vector space sum $\mathcal{H} = \mathcal{H}_c \oplus \mathcal{H}_l$ by the space $\mathcal{H}_l$ spanned by the leakage states. We quantify the amount of information lost into the leakage subspace by cropping the unitary evolution on the entire system to the computational states and calculating the distance to unitarity of the projected propagator U_c as

$$\mathcal{L} = 1 - \mathrm{tr}(U_c^\dagger U_c)/d. \tag{15}$$

In order to investigate the amount of incoherent leakage (i.e., caused by noise), this cost function can be combined with a noise simulation. The calculation of leakage was also generalized for open quantum systems [52].

D. Optimization procedures

The minimization of such a fidelity over the optimization space spanned by all possible control pulses $u(t)$

formally defines QOC as the minimization problem

$$\min_{u(t)} \mathcal{I}[u(t)] = \mathcal{I}[u^*(t)]. \tag{16}$$

An introduction to QOC was written by D'Alessandro [53].

Although QOPT provides a general interface for cost functions that can be used with any optimization algorithm, we set a special focus on the use of gradient-based optimization algorithms. These are widely used and also applicable in QOC [22]. For this purpose we implement the analytic calculation of exact gradients, which do not require any assumption about the time discretization or about the control strength. An alternative method for the calculation of gradients is the use of automatic differentiation [18,54].

By default, QOPT assumes a fixed duration of a control pulse. If an optimization of the pulse duration is of interest, it can be achieved by a rescaling of the problem. If we have, for example, a Hamiltonian model of the form $H = uB$ with a single control amplitude $u = v$ then the solution to the corresponding Schrödinger's equation is calculated as

$$U = e^{-iuBt} = e^{-i(ut)B}. \tag{17}$$

For ease of notation, we consider only constant control, but the concept can readily be generalized. Thus, we can redefine our optimization by setting the pulse length to 1 and defining our effective Hamiltonian as $H_{\text{eff}} = u_1 B$ with $u_1 = v_1 v_2$ and the control parameter $v_1 = v$ and $v_2 = t$. It should be noticed that noise spectra are not automatically adapted with this rescaling.

When multiple cost functions are evaluated, the software supplies them as a vector to the optimization algorithm. This leaves more options for the optimization and allows for an individual weight for each cost function.

III. IMPLEMENTATION

In this section we present QOPT's object-oriented implementation by first discussing the structure of an optimal control setup with QOPT, and then outlining the optimization of an $X_{\pi/2}$ gate for a single qubit controlled by Rabi driving as a simple illustrative example.

A. Program flow

The intended setup is plotted as an activity diagram in Fig. 1 giving a full picture, although not every feature and every class need be used in practice. Only the solver class is central to the simulation. The diagram shows the modular software structure and the flow of information between the classes.

The optimization commences with a set of initial optimization parameters v_k, which can be chosen randomly or based on some insight. The package also features convenience functions to run the simulation with many different initial conditions in parallel to exploit the problem structure for trivial full parallelization.

First, the ideal pulse parameters are mapped to the actual pulse seen by the qubits as defined by a transfer function class. Then the control amplitudes u_k are calculated by the amplitude function class. The control amplitudes enter the Schrödinger or master equation in Lindblad form together with the drift dynamics and the noise. The appropriate differential equation to describe the system is chosen by selecting the solver algorithm class. For the numerical solution of the Schrödinger equation, we assume piecewise constant control during n_t time-steps of length $(\Delta t_1, \ldots, \Delta t_{n_t})$. The total unitary propagator of an evolution is calculated as a product of matrix exponentials of the time-independent Hamiltonians

$$U = e^{-iH(t_{n_t})\Delta t_{n_t}} \ldots e^{-iH(t_2)\Delta t_2} \times e^{-iH(t_1)\Delta t_1}, \tag{18}$$

using the convention $\hbar = 1$. The solution of the differential equation is subsequently passed to the cost function class, to calculate the figure of merit for the optimization.

The simulator class encircles other classes in Fig. 1 because it provides the interface between the simulation and the optimization algorithm. Furthermore, it gathers run- time statistics like the time spent in each cost function and on the calculation of the gradient. The optimizer class uses this interface to run the simulation in a loop until the internal termination criteria are met. Then it saves the final state in a result class and passes it to the data container class. The analyzer class can be used to visualize the results of several optimization runs stored in the data container class.

The object-oriented modular implementation of the code allows the user to easily replace single parts of the optimization framework. Among other things, this allows the user to make changes to the cost function (i.e., to use a different fidelity metric) or use a specific transfer or amplitude function. With the interface provided by the simulation class it is possible to use most standard optimization algorithms. Currently, the "minimize" and "least squares" functions of scipy's optimization subpackage are supported [55].

Numeric operations are encapsulated in an operator class. The computationally most expensive single operation during the simulation is the calculation of a matrix exponential as required for the numeric solution of the differential equations. The encapsulation allows the exchange of algorithms for the calculation of this matrix exponential (see [56] for examples). The Qobj class from

QUTIP can be converted automatically into the QOPT operator class to improve the compatibility between both packages. This makes it easy to transfer simulations to QOPT.

More information about QOPT can be found in the online documentation [40]. This features a complete API reference documentation and two series of IPython notebooks. The former explains each feature of QOPT in detail while the latter discusses practical examples including some information about the physics and numerics of qubit simulation, how noise sources can be characterized, which type of noise simulation is the most efficient in each case and which effects noise will have on the system. These notebooks also serve as integration tests by demonstrating the consistency of different methods and the comparison with analytic calculations. Together with various unit tests for the critical parts of the implementation, they ensure the reliability of QOPT.

B. Example

We illustrate the usage of QOPT in the pulse optimization with the example of an $X_{\pi/2}$ single-qubit gate and optimize the pulse separately for quasistatic and autocorrelated fast noise, so that we can demonstrate how different noise models are implemented in QOPT's API.

The Hamiltonian of the example will be a single qubit manipulated by Rabi driving in the rotating frame, which can be denoted by $H = u_x(t)\sigma_x + u_y(t)\sigma_y + \delta_\omega(t)\sigma_z$, where $u_x(t)$ and $u_y(t)$ are the control amplitudes (corresponding to the in-phase amplitude I and the quadrature amplitude Q of quadrature amplitude control), $\delta_\omega(t)$ is the deviation of the driving frequency from the resonance frequency, and σ_i, $i \in x, y, z$, is the Pauli matrix. This Hamiltonian can be identified with the general description in Eqs. (1)–(3). The first two terms represent the control Hamiltonian and the third term the noise Hamiltonian with the noise amplitude $b(t) = \delta\omega(t)$, the noise susceptibility being 1 and the Hermitian operator $C = \sigma_z$.

The detuning $\delta_\omega(t)$ enters Schrödinger's equation as a stochastic variable. In a first optimization we assume that the resonance frequency changes much more slowly than our gate time and is thus assumed to be constant during the pulse [$\delta_\omega(t) = \delta_\omega$]. We therefore integrate δ_ω over a Gaussian distribution and calculate the corresponding infidelity $\mathcal{I}_{\text{qs}}$ using a Monte Carlo method.

In the second optimization we will use the final parameters of the first optimization as the initial pulse and assume that the resonance frequency is subject to pink noise and therefore the power spectral density of $\delta_\omega(t)$ has the form $S(f) = S_0/f$. In this case we calculate the entanglement infidelity caused by systematic deviations $\mathcal{I}_e$ separately from that caused by noise, which we calculate with filter functions $\mathcal{I}_{\text{FF}}$ as in Eq. (12).

We optimize a pulse with 20 time-steps of equal length:

```
import qopt as qo
import numpy as np
import matplotlib.pyplot as plt

n_time_steps = 20
delta_t = .5 * np.pi
```

We start setting up the first simulation with quasistatic noise. The noise trace generator (NTG) provides the noise samples for the solver algorithm:

```
noise_gen = qo.NTGQuasiStatic(
    n_samples_per_trace=n_time_steps,
    n_traces=10,
    standard_deviation=[.05, ]
)
```

The solver class holds the information about the Hamiltonian, including the corresponding noise trace generator. The quantum operators are represented by the dense operator class that is based on a numpy array. We can also decide the exponential method, which is the algorithm used to calculate the matrix exponential and on demand its derivative (usually in combination). QOPT implements, among other methods, a spectral decomposition or, as in this example, the default scipy method for calculation of the Fréchet derivative of the matrix exponential. The solver class also interfaces to the FILTER_FUNCTIONS package [47].

```
solver = qo.SchroedingerSMonteCarlo(
    h_ctrl=[.5 * qo.DenseOperator.pauli_x(),
            .5 * qo.DenseOperator.pauli_y()],
    h_drift=[0 * qo.DenseOperator.pauli_x()],
    tau=delta_t * np.ones(n_time_steps),
    exponential_method='Frechet',
    h_noise=[.5 * qo.DenseOperator.pauli_z()],
    noise_trace_generator=noise_gen,
    filter_function_h_n=[
        [qo.DenseOperator.pauli_z().data,
        np.ones(n_time_steps)]
    ]
)
```

The target operation is an $X_{\pi/2}$-gate and the cost function for the first simulation evaluates the mean deviation between the simulated propagator and the target gate using the entanglement infidelity as given in Eq. (13). We can also choose to neglect the systematic errors to calculate the entanglement fidelity on average between the unperturbed simulation and the simulation including noise. The optimization algorithm is by default the gradient-based L-BFGS-B algorithm implemented in scipy and the bounds restrict the search space.

```
target = (
        qo.DenseOperator.pauli_x()
    ).exp(.25j * np.pi)

cost_func_qs = qo.OperationNoiseInfidelity(
    solver=solver,
    target=target,
    neglect_systematic_errors=False,
    label=[r'$\mathcal{I}_{\mathrm{qs}}$', ]
)
```

```
optimizer_qs = qo.ScalarMinimizingOptimizer(
    system_simulator=qo.Simulator(
        solvers=[solver, ],
        cost_funcs=[cost_func_qs, ]
    ),
    bounds=[[-1, 1], ] * 2 * n_time_steps
)
```

In the second simulation we use one cost function for the systematic deviations calculated by the standard entanglement fidelity and a second cost function calculating the infidelity caused by pink noise based on filter functions. The sampling frequencies for the integral in equation Eq. (12) are set at the keyword argument `omega`.

```
cost_func_plain = qo.OperationInfidelity(
    solver=solver,
    target=target,
    label=[r'$\mathcal{I}_{\mathrm{e}}$']
)
def noise_psd(f):
    return 1e-3 / f

total_time = n_time_steps * delta_t
start = np.log10(1 / total_time)
end = np.log10(1 / delta_t)

cost_func_ff = \
    qo.OperatorFilterFunctionInfidelity(
        solver=solver,
        noise_power_spec_density=noise_psd,
        omega=np.logspace(start, end, 200),
        label=[r'$\mathcal{I}_{\mathrm{ff}}$']
    )

optimizer_ff = qo.ScalarMinimizingOptimizer(
    system_simulator=qo.Simulator(
        solvers=[solver, ],
        cost_funcs=[cost_func_plain,
                    cost_func_ff]
    ),
    bounds=[[-1, 1], ] * 2 * n_time_steps
)
```

The simulation is executed with the following commands:

```
np.random.seed(0)
result = optimizer_qs.run_optimization(
    initial_control_amplitudes=
    np.random.rand(20, 2))
result_ff = optimizer_ff.run_optimization(
    initial_control_amplitudes=
    result.final_parameters)
```

The resulting data can be stored with the data container class and plotted with the analyzer class.

```
data_qs = qo.DataContainer()
data_qs.append_optim_result(result)
analyser_qs = qo.Analyser(data_qs)
data_ff = qo.DataContainer()
data_ff.append_optim_result(result_ff)
analyser_ff = qo.Analyser(data_ff)
```

Some cosmetics in plotting commands are shortened for the sake of brevity. We can plot a final pulse (compare to Fig. 3) with:

```
solver.transfer_function.plot_pulse(result.
    final_parameters)
```

And the infidelity during the optimization (compare to Fig. 2) with:

```
fig, axes = plt.subplots(2)
analyser_qs.plot_costs(ax=axes[0])
analyser_ff.plot_costs(ax=axes[1])
```

Each optimization took less than 10 s on a desktop PC. The decrease in the infidelities during the optimizations can be seen in Fig. 2 and the final pulses are plotted in Fig. 3. With the second optimization run, the infidelity as calculated in Eq. (12) is decreased by a factor of about 6, demonstrating the benefit of explicitly considering fast autocorrelated noise.

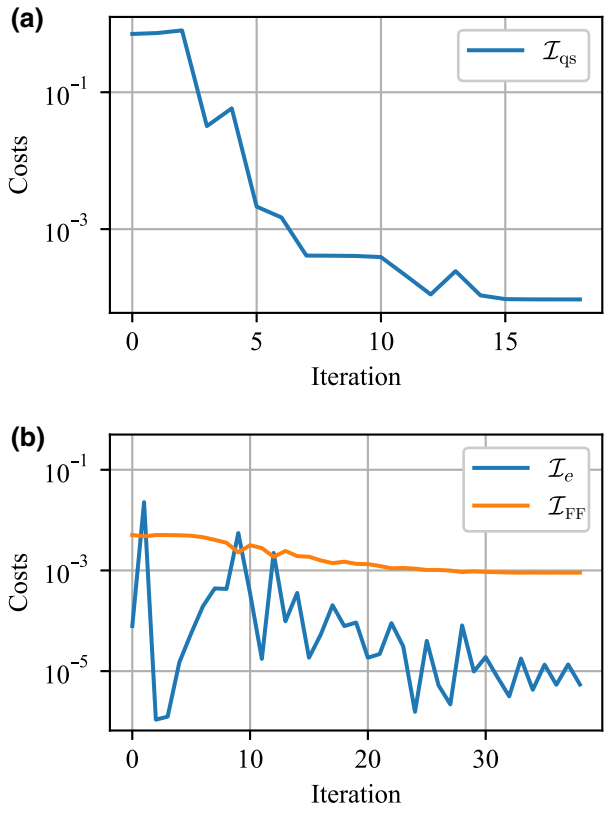

FIG. 2. Infidelities as cost functions of the iteration during the optimization. (a) During the first optimization, we minimize the infidelity in a Monte Carlo simulation of quasistatic noise $\mathcal{I}_{\mathrm{qs}}$ to find a pulse which is not susceptible to slow noise. (b) Subsequently, we use the final parameters of the first optimization to optimize the pulse for pink noise. We can see that the infidelity $\mathcal{I}_{\mathrm{FF}}$ of Eq. (12) decreases during the second optimization by a factor of about 6. Thus pulses that mitigate slow noise are far from perfect in mitigating pink noise.

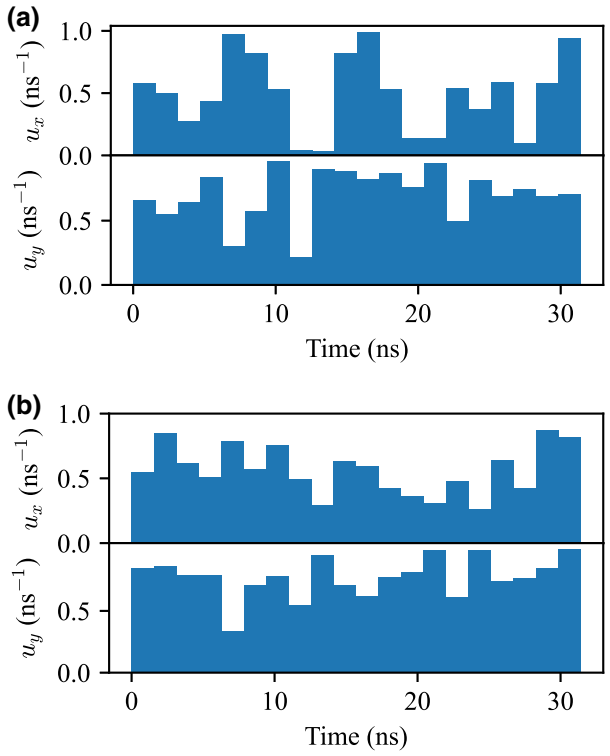

FIG. 3. Plots of the optimized control amplitudes as a function of time. (a) In the first optimization run, the control pulse is optimized for quasistatic noise. (b) The control amplitudes of the first run are used as a starting point for an optimization in the presence of pink noise.

IV. SUMMARY AND OUTLOOK

Our open-source software package QOPT [38] provides a general platform for the realistic simulation of qubit systems and QOC. We set the focus on the accurate and efficient simulation of arbitrary classical noise by including three noise simulation methods with distinct application areas. Quasistatic noise is efficiently simulated with Monte Carlo methods or numerical integration, Markovian noise is described best by a master equation in Lindblad form, while fast non-Markovian noise can be treated with filter functions. In the implementation of each method, the noise model can be drive-dependent. The limitations of control hardware are accounted for by the use of transfer functions. In addition to the example provided in this paper, the online documentation [40] contains a complete API documentation and numerous IPython notebooks discussing QOPT's features and an introduction to practical simulations.

We also provide an open-source repository of public simulation and optimal control projects called QOPT-APPLICATIONS [57]. This can serve as starting point for simulations and facilitates the transfer of knowledge and optimal control techniques.

QOC will continue to play a role in the search for the optimal qubit system for the construction of universal quantum computers. The increasing number of qubits in quantum processors leads to new challenges like the mitigation of crosstalk between adjacent qubits, robustness of quantum gates towards qubit inhomogeneities, or increased noise levels in quantum processors operated at higher temperatures. Since the improvements from adopting QOC can be dramatic, any assessment of a quantum computation platform should take the applicable control techniques into account. The clean interface between the simulation and the optimization algorithm makes QOPT ideal for the comparison of various optimization algorithms in the context of QOC. Optimization algorithms based on artificial intelligence are interesting candidates.

While we have found QOPT to be very useful in a number of applications, there is certainly much room for extensions. One such a feature could be the application of QOPT for spectral noise analysis. This could, for example, be achieved by the introduction of a cost function class measuring the sensitivity of a pulse towards noise of a specific frequency with the help of filter functions.

If performance becomes a bottleneck, QOPT could profit from a high-performance compilation with numba [58], the use of other algorithms for the calculation of the matrix exponential, or highly optimized implementations of performance critical functions in a compiled language.

For even more general modeling and pulse parameterization capabilities, the amplitude and the transfer function classes could be generalized, allowing, for example, the application of the amplitude function before the transfer function.

We published QOPT with the aim of establishing a community standard for qubit simulations. The application of a common simulation platform makes simulations less time-consuming and more reproducible compared to the use of special-purpose simulation code. Reproducibility increases the trust in simulation results and facilitates the transfer of simulation and optimal control techniques between different qubit systems. We thus encourage users to upload their simulation code to QOPT-APPLICATIONS [57] to increase their visibility and contribute to the advancement of the state of the art. We encourage the participation in the development and welcome feedback on which new features would be useful.

ACKNOWLEDGMENTS

We thank Alexander Pitchford, Eric Giguère, Neill Lambert, and Franco Nori for helpful discussions and advice in the design of QOPT. We also thank Alexander Willmes, Christian Gorjaew, Paul Surrey, Frederike Butt, and Jiaqi Ai for testing QOPT and providing feedback. We acknowledge support from the European Research Council (ERC) under the European Union's Horizon 2020

research and innovation program (Grant Agreement No. 679342), Impulse and Networking Fund of the Helmholtz Association.

[1] W. G. Unruh, Maintaining coherence in quantum computers, Phys. Rev. A **51**, 992 (1995).

[2] S. J. Glaser, U. Boscain, T. Calarco, C. P. Koch, W. Köckenberger, R. Kosloff, I. Kuprov, B. Luy, S. Schirmer, T. Schulte-Herbrüggen, D. Sugny, and F. K. Wilhelm, Training Schrödinger's cat: Quantum optimal control: Strategic report on current status, visions and goals for research in Europe, Eur. Phys. J. D **69**, 279 (2015).

[3] C. Brif, R. Chakrabarti, and H. Rabitz, Control of quantum phenomena: Past, present and future, New J. Phys. **12**, 075008 (2010).

[4] J. M. Chow, L. DiCarlo, J. M. Gambetta, F. Motzoi, L. Frunzio, S. M. Girvin, and R. J. Schoelkopf, Optimized driving of superconducting artificial atoms for improved single-qubit gates, Phys. Rev. A **82**, 040305 (2010).

[5] J. Kelly *et al.*, Optimal Quantum Control Using Randomized Benchmarking, Phys. Rev. Lett. **112**, 240504 (2014).

[6] J. Preskill, Quantum computing in the NISQ era and beyond, Quantum **2**, 79 (2018).

[7] X. Wang, L. S. Bishop, E. Barnes, J. P. Kestner, and S. D. Sarma, Robust quantum gates for singlet-triplet spin qubits using composite pulses, Phys. Rev. A **89**, 022310 (2014).

[8] X. C. Yang and X. Wang, Noise filtering of composite pulses for singlet-triplet qubits, Sci. Rep. **6**, 1 (2016).

[9] B. E. Anderson, H. Sosa-Martinez, C. A. Riofrío, I. H. Deutsch, and P. S. Jessen, Accurate and Robust Unitary Transformations of a High-Dimensional Quantum System, Phys. Rev. Lett. **114**, 240401 (2015).

[10] F. F. Floether, P. de Fouquieres, and S. G. Schirmer, Robust quantum gates for open systems via optimal control: Markovian versus non-Markovian dynamics, New J. Phys. **14**, 073023 (2012).

[11] L. Geck, A. Kruth, H. Bluhm, S. van Waasen, and S. Heinen, Control electronics for semiconductor spin qubits, Quantum Sci. Technol. **5**, 015004 (2019).

[12] E. Charbon, F. Sebastiano, A. Vladimirescu, H. Homulle, S. Visser, L. Song, and R. M. Incandela, in *2016 IEEE International Electron Devices Meeting (IEDM)* (2016), p. 13.5.1.

[13] P. Cerfontaine, T. Botzem, J. Ritzmann, S. S. Humpohl, A. Ludwig, D. Schuh, D. Bougeard, A. D. Wieck, and H. Bluhm, Closed-loop control of a gaas-based singlet-triplet spin qubit with 99.5% gate fidelity and low leakage, Nat. Commun. **11**, 4144 (2020).

[14] P. Cerfontaine, R. Otten, and H. Bluhm, Self-Consistent Calibration of Quantum-Gate Sets, Phys. Rev. Appl. **13**, 044071 (2020).

[15] D. J. Egger and F. K. Wilhelm, Adaptive Hybrid Optimal Quantum Control for Imprecisely Characterized Systems, Phys. Rev. Lett. **112**, 240503 (2014).

[16] P. Cerfontaine, R. Otten, M. A. Wolfe, P. Bethke, and H. Bluhm, High-fidelity gate set for exchange-coupled singlet-triplet qubits, Phys. Rev. B **101**, 155311 (2020).

[17] P. Cerfontaine, T. Botzem, D. P. DiVincenzo, and H. Bluhm, High-Fidelity Single-Qubit Gates for Two-Electron Spin Qubits in Gaas, Phys. Rev. Lett. **113**, 150501 (2014).

[18] N. Wittler, F. Roy, K. Pack, M. Werninghaus, A. S. Roy, D. J. Egger, S. Filipp, F. K. Wilhelm, and S. Machnes, Integrated Tool set for Control, Calibration, and Characterization of Quantum Devices Applied to Superconducting Qubits, Phys. Rev. Appl. **15**, 034080 (2021).

[19] F. Motzoi, J. M. Gambetta, S. T. Merkel, and F. K. Wilhelm, Optimal control methods for rapidly time-varying hamiltonians, Phys. Rev. A **84**, 022307 (2011).

[20] O. E. Dial, M. D. Shulman, S. P. Harvey, H. Bluhm, V. Umansky, and A. Yacoby, Charge Noise Spectroscopy Using Coherent Exchange Oscillations in a Singlet-Triplet Qubit, Phys. Rev. Lett. **110**, 146804 (2013).

[21] J. Yoneda, K. Takeda, T. Otsuka, T. Nakajima, M. R. Delbecq, G. Allison, T. Honda, T. Kodera, S. Oda, Y. Hoshi, N. Usami, K. M. Itoh, and S. Tarucha, A quantum-dot spin qubit with coherence limited by charge noise and fidelity higher than 99.9%, Nat. Nanotechnol. **13**, 102 (2018).

[22] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, Optimal control of coupled spin dynamics: Design of nmr pulse sequences by gradient ascent algorithms, J. Magn. Reson. **172**, 296 (2005).

[23] S. G. Schirmer and P. de Fouquieres, Efficient algorithms for optimal control of quantum dynamics: The krotov method unencumbered, New J. Phys. **13**, 073029 (2011).

[24] C.-H. Huang and H.-S. Goan, Robust quantum gates for stochastic time-varying noise, Phys. Rev. A **95**, 062325 (2017).

[25] P. de Fouquieres, S. Schirmer, S. Glaser, and I. Kuprov, Second order gradient ascent pulse engineering, J. Magn. Reson. **212**, 412 (2011).

[26] S. Machnes, E. Assémat, D. Tannor, and F. K. Wilhelm, Tunable, Flexible, and Efficient Optimization of Control Pulses for Practical Qubits, Phys. Rev. Lett. **120**, 150401 (2018).

[27] J. D. Teske, S. S. Humpohl, R. Otten, P. Bethke, P. Cerfontaine, J. Dedden, A. Ludwig, A. D. Wieck, and H. Bluhm, A machine learning approach for automated fine-tuning of semiconductor spin qubits, Appl. Phys. Lett. **114**, 133102 (2019).

[28] T. Caneva, T. Calarco, and S. Montangero, Chopped random-basis quantum optimization, Phys. Rev. A **84**, 022326 (2011).

[29] R. Heck, O. Vuculescu, J. J. Sørensen, J. Zoller, M. G. Andreasen, M. G. Bason, P. Ejlertsen, O. Elíasson, P. Haikka, J. S. Laustsen, L. L. Nielsen, A. Mao, R. Müller, M. Napolitano, M. K. Pedersen, A. R. Thorsen, C. Bergenholtz, T. Calarco, S. Montangero, and J. F. Sherson, Remote optimization of an ultracold atoms experiment by experts and citizen scientists, Proc. Natl. Acad. Sci. **115**, E11231 (2018).

[30] S. Machnes, U. Sander, S. J. Glaser, P. de Fouquières, A. Gruslys, S. Schirmer, and T. Schulte-Herbrüggen, Comparing, optimizing, and benchmarking quantum-control algorithms in a unifying programming framework, Phys. Rev. A **84**, 022305 (2011).

[31] J. Johansson, P. Nation, and F. Nori, Qutip 2: A python framework for the dynamics of open quantum systems, Comput. Phys. Commun. **184**, 1234 (2013).

[32] B. Li, S. Ahmed, S. Saraogi, N. Lambert, F. Nori, A. Pitchford, and N. Shammah, Pulse-level noisy quantum circuits with QuTiP, Quantum **6**, 630 (2022).
[33] M. H. Goerz, D. Basilewitsch, F. Gago-Encinas, M. G. Krauss, K. P. Horn, D. M. Reich, and C. P. Koch, Krotov: A python implementation of krotov's method for quantum optimal control, SciPost Phys. **7**, 80 (2019).
[34] J. Sørensen, J. Jensen, T. Heinzel, and J. Sherson, Qengine: A c++ library for quantum optimal control of ultracold atoms, Comput. Phys. Commun. **243**, 135 (2019).
[35] H. Silvério, S. Grijalva, C. Dalyac, L. Leclerc, P. J. Karalekas, N. Shammah, M. Beji, L.-P. Henry, and L. Henriet, Pulser: An open-source package for the design of pulse sequences in programmable neutral-atom arrays, Quantum **6**, 629 (2022).
[36] C. P. Koch, Controlling open quantum systems: Tools, achievements, and limitations, J. Phys.: Condens. Matter **28**, 213001 (2016).
[37] L. Pawela and P. Sadowski, Various methods of optimizing control pulses for quantum systems with decoherence, Quantum Inf. Process. **15**, 1937 (2016).
[38] J. Teske, qopt: A simulation and quantum optimal control package, https://https://github.com/qutech/qopt (2020).
[39] H. Ball, M. J. Biercuk, A. R. R. Carvalho, J. Chen, M. Hush, L. A. D. Castro, L. Li, P. J. Liebermann, H. J. Slatyer, C. Edmunds, V. Frey, C. Hempel, and A. Milne, Software tools for quantum control: Improving quantum computer performance through noise and error suppression, Quantum Sci. Technol. **6**, 044011 (2021).
[40] J. Teske, qopt: Api documentation and introduction notebooks, https://qopt.readthedocs.io/en/latest/index.html (2020).
[41] R. Caflish, Monte carlo and quasi-monte carlo methods, Acta Numerica **7**, 1 (1998).
[42] T. F. Havel, Robust procedures for converting among lindblad, kraus and matrix representations of quantum dynamical semigroups, J. Math. Phys. **44**, 534 (2003).
[43] D. Manzano, A short introduction to the lindblad master equation, AIP Adv. **10**, 025106 (2020).
[44] T. J. Green, J. Sastrawan, H. Uys, and M. J. Biercuk, Arbitrary quantum control of qubits in the presence of universal noise, New J. Phys. **15**, 095004 (2013).
[45] C.-H. Huang, C.-H. Yang, C.-C. Chen, A. S. Dzurak, and H.-S. Goan, High-fidelity and robust two-qubit gates for quantum-dot spin qubits in silicon, Phys. Rev. A **99**, 042310 (2019).
[46] T. Hangleiter, P. Cerfontaine, and H. Bluhm, Filter-function formalism and software package to compute quantum processes of gate sequences for classical non-Markovian noise, Phys. Rev. Res. **3**, 043047 (2021).
[47] T. Hangleiter, filter_functions: A package for efficient numerical calculation of generalized filter functions, https://github.com/qutech/filter_functions (2019).
[48] P. Cerfontaine, T. Hangleiter, and H. Bluhm, Filter Functions for Quantum Processes under Correlated Noise, Phys. Rev. Lett. **127**, 170403 (2021).
[49] I. N. M. Le, J. D. Teske, T. Hangleiter, P. Cerfontaine, and H. Bluhm, Analytic Filter-Function Derivatives for Quantum Optimal Control, Phys. Rev. Appl. **17**, 024006 (2022).
[50] M. A. Nielsen, A simple formula for the average gate fidelity of a quantum dynamical operation, Phys. Lett. A **303**, 249 (2002).
[51] M. D. Grace, J. Dominy, R. L. Kosut, C. Brif, and H. Rabitz, Environment-invariant measure of distance between evolutions of an open quantum system, New J. Phys. **12**, 015001 (2010).
[52] C. J. Wood and J. M. Gambetta, Quantification and characterization of leakage errors, Phys. Rev. A **97**, 032306 (2018).
[53] D. D'Alessandro, *Introduction to Quantum Control and Dynamics* (Chapman and Hall/CRC, Ames, USA, 2007), 1st ed.
[54] M. Abdelhafez, D. I. Schuster, and J. Koch, Gradient-based optimal control of open quantum systems using quantum trajectories and automatic differentiation, Phys. Rev. A **99**, 052327 (2019).
[55] P. Virtanen *et al.*, SciPy 1.0: Fundamental algorithms for scientific computing in python, Nat. Methods **17**, 261 (2020).
[56] C. Moler and C. V. Loan, Nineteen dubious ways to compute the exponential of a matrix, SIAM Review **20**, 801 (1978).
[57] J. Teske, qopt-applications: Simulations and optimal control implemented with qopt, https://github.com/qutech/qopt-applications (2020).
[58] S. K. Lam, A. Pitrou, and S. Seibert, in *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15 (Association for Computing Machinery, New York, NY, USA, 2015).