Gradient Ascent Pulse Engineering with Feedback

Riccardo Porotti,[1, 2, *] Vittorio Peano,[1] and Florian Marquardt[1, 2]
[1] *Max Planck Institute for the Science of Light, Erlangen, Germany*
[2] *Department of Physics, Friedrich-Alexander Universität Erlangen-Nürnberg, Germany*
(Dated: March 9, 2022)

Efficient approaches to quantum control and feedback are essential for quantum technologies, from sensing to quantum computation. Pure control tasks have been successfully solved using optimization techniques, including methods like gradient-ascent pulse engineering (GRAPE) , relying on a differentiable model of the quantum dynamics. For feedback tasks, such methods are not directly applicable, since the aim is to discover strategies conditioned on measurement outcomes. There, model-free reinforcement learning (RL) has recently proven a powerful new ansatz. What is missing is a way to combine the best of both approaches for scenarios that go beyond weak measurements. In this work, we introduce feedback-GRAPE, which borrows concepts from model-free RL to incorporate the response to strong stochastic (discrete or continuous) measurements, while still performing direct gradient ascent through the quantum dynamics. We illustrate its power on a Jaynes-Cummings model with feedback, where it yields interpretable feedback strategies for state preparation and stabilization in the presence of noise. This approach could be employed for discovering strategies in a wide range of feedback tasks, from calibration of multi-qubit devices to linear-optics quantum computation strategies, quantum-enhanced sensing with adaptive measurements, and quantum error correction.

I. INTRODUCTION

The application of optimal-control techniques to quantum systems [1, 2] forms a cornerstone of modern quantum technologies, ranging from the tailoring of laser pulses acting on molecules to the synthesis of unitaries in multi-qubit systems as part of the "compilation" of quantum algorithms for specific hardware platforms. Since the equations of quantum dynamics are explicitly known and even differentiable, one can exploit this knowledge and specifically make use of powerful gradient-based techniques. The most prominent approach is "gradient-ascent pulse engineering" (GRAPE)[3, 4], with its efficient evaluation of gradients, together with its variants. GRAPE has been employed to find optimal control sequences for spin systems [3, 5, 6], coupled qubits [7, 8], an implementation of the Jaynes-Cummings model [9], and qubit-cavity lattices [10], among many other examples. It has also been used to optimize open dynamics [11, 12], has been turned into an adaptive approach to cope with parameter uncertainties [13], and has been extended to second-order optimization techniques [14]. Other efficient gradient-based optimal control approaches have also been presented recently (e.g. [15]).

However, there is one crucial extension that is not easily addressed by such gradient-based techniques: feedback. Conditioning the control sequence based on the stochastic outcomes of quantum measurements is an important component of many more challenging tasks [16]. It allows to remove entropy from the system and is therefore essential in applications like state preparation and stabilization in the presence of noise [17–21], adaptive measurements [22], or quantum error correction with its syndrome extraction (e.g. [23–26]). Feedback strategies live in a space that is combinatorially larger than that of pure control strategies, since every sequence of measurement outcomes may require a different response.

Feedback is a natural ingredient of reinforcement learning (RL)[27], a set of methods from the domain of machine learning. These methods consider the interaction of an agent (a controller) with an environment (a system). In particular, in the very powerful and flexible so-called model-free RL approaches, the internal dynamics of the environment need not be known but is rather treated as a black box. During the last few years, a number of groups have demonstrated numerically the promise of model-free RL for quantum physics. This included both pure control tasks (e.g. [28–31], even in an experiment [32]) but in particular also the more challenging quantum real-time feedback tasks that rely on adaptive responses to measurement outcomes [33–36].

As mentioned above, model-free RL approaches treat the quantum system as a "black box". On the one hand, this can be an advantage in applying it to experimental setups whose parameters are partially unknown (as emphasized e.g. in [32, 35]). On the other hand, much of the training time is spent in learning (implicitly) a model of the dynamics while simultaneously attempting to find good feedback strategies. This can make learning relatively inefficient.

It would therefore seem desirable to combine the best aspects of direct gradient-based approaches (making use of our knowledge of the differentiable quantum dynamics) and model-free RL (with its natural incorporation of feedback to stochastic measurements). In this work, we present such a technique, which we refer to as 'feedback-GRAPE'. It keeps the ability to exploit gradients through the quantum dynamics, while also allowing for feedback

[*] riccardo.porotti@mpl.mpg.de

to stochastic quantum measurements. It borrows a certain idea from policy gradient, a subclass of model-free RL, but instead of applying gradients to a policy (probabilistic choice of actions), we apply gradients to the probability of measurement outcomes, in addition to the continuous deterministic dynamics that would be the domain of GRAPE and similar methods.

The procedure to be presented enables us in particular to deal with the outcomes of strong stochastic measurements, both discrete and continuous, with arbitrary probability distributions. The special limiting case of weak Gaussian-distributed measurements, which does not yet require the mathematical treatment that we will introduce, has recently been considered by Schäfer et al. [37], which can thus be considered an important first step towards the general method we are going to discuss here. Another aspect of our approach is that it is well-suited for the application of automatic differentiation techniques. Such techniques have recently been suggested as a convenient tool for optimal quantum control in a number of works, namely [38] and subsequent articles [39–42], which also include open-systems dynamics treated by quantum jump trajectories [43].

Overall, the technique we introduce here, feedback-GRAPE, is conceptually simple: GRAPE-type gradient ascent for the continuous control parts (possibly implemented using automatic differentiation for convenience; and in any case exploiting modern gradient optimizers), supplemented with stochastic sampling of measurement outcomes, plus the addition of an important required 'correction term' to the overall cost/reward function (for discrete measurement outcomes). Although the method is general, we find that it works particularly well for feedback sequences with a modular structure, i.e. where building blocks like unitaries and measurements are combined in discrete time steps. These are useful scenarios, since it may become easier to interpret the resulting strategies. We illustrate the power of feedback-GRAPE in a series of different tasks. All these example tasks are based on the physical scenario of a Jaynes-Cummings model (coupled qubit-cavity system) supplemented with feedback. This scenario is closely related to modern quantum computing experimental platforms.

When viewed from the general perspective of reinforcement learning, the feedback-GRAPE approach to be presented here can be classified as a model-based RL technique, although this categorization includes a wide range of approaches (sometimes even including model-free RL applied to model-based simulations).

In the following, we will first present the general method, then provide illustrative numerical examples, and finally discuss further extensions.

II. FEEDBACK-GRAPE METHOD

We consider a general dissipative quantum system with feedback (for an overview of the scheme, see Fig. 1). Sup-

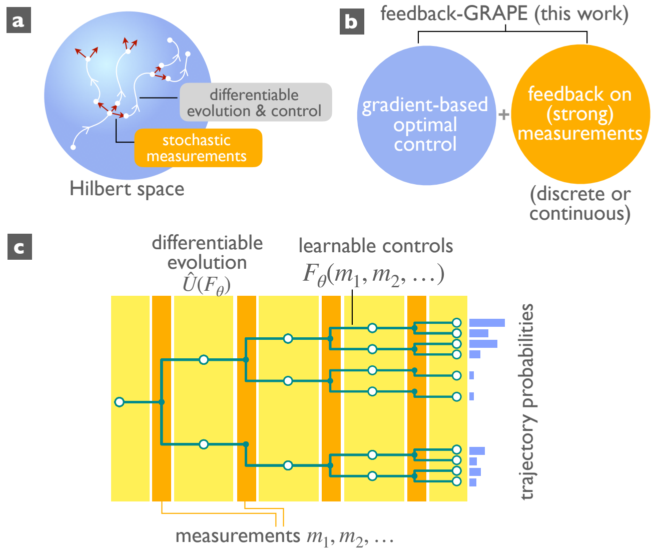
a
differentiable evolution & control
stochastic measurements
Hilbert space
b
feedback-GRAPE (this work)
gradient-based optimal control
+
feedback on (strong) measurements
(discrete or continuous)
c
differentiable evolution
$\hat{U}(F_\theta)$
learnable controls
$F_\theta(m_1, m_2, \dots)$
trajectory probabilities
measurements $m_1, m_2, \dots$

FIG. 1. Quantum feedback control with strong measurements. (a) Such feedback tasks combine smooth, differentiable dynamics in Hilbert space with measurement-induced jumps. (b) This work integrates model-based techniques relying on gradients and feedback to strong stochastic measurements. (c) Schematic decision-tree representation of a feedback strategy for discrete measurement outcomes. Intervals of differentiable evolution with optimizable control functions F_θ^j depend on the sequence of outcomes $m_1, \dots, m_j$. In general, the evolution can be dissipative.

pose measurements are performed at times $t_1, t_2, \dots, t_N$, and the evolution is controlled – in a manner to be optimized – based on the corresponding measurement outcomes m_j. Specifically, the control parameter (which might be a vector) applied during the time interval $[t_j, t_{j+1}]$ can be written as some function of all previous measurement results,

$$F_\theta^j(m_j, m_{j-1}, \dots, m_1). \tag{1}$$

Here we anticipated that the feedback-control functions F_θ^j are parametrized, depending on parameters θ that will be optimized via gradient ascent (θ is typically a high-dimensional vector). We assume F_θ^j to be differentiable with respect to θ, though not necessarily with respect to the measurement results, which may or may not be discrete (more on that below). For brevity, we will sometimes write $F_\theta^j(m)$, it being understood that F_θ^j can only depend on outcomes up to and including m_j. Ultimately, the value of F_θ^j will be provided by a neural network, or, alternatively, a lookup table: we comment on these different approaches further below, but the present considerations are independent of this aspect. In practice, the control vector F_θ^j might enter a Hamiltonian or directly a parametrized unitary gate. On a minor note, in some scenarios, during the first time interval $[0, t_1]$, one might apply a control F^0 that does not depend on any previous measurement outcomes but can still be optimized.

With this notation in place, the time evolution of the system's density matrix, for a particular measurement sequence $m = m_1, m_2, \ldots$, can be written in the general form

$$\hat{\rho}(T|m) = \Phi^{(m)}(F_\theta^N, F_\theta^{N-1}, \ldots, F_\theta^0)[\hat{\rho}(0)] \quad (2)$$

where $\Phi^{(m)}$ is the map that depends on the control parameters and implements the quantum-dissipative time evolution throughout the whole time interval $[0,T]$, *conditioned on the given fixed sequence m of measurement outcomes.* Note that our definition implies that $\Phi^{(m)}$ itself is not a completely positive (CP) map, because it contains the renormalization of the quantum state required after each measurement (it implements a "quantum instrument"), which introduces a nonlinear dependence on the initial state. To obtain the unconditional average quantum state, the average $\langle \ldots \rangle_m$ of this expression may be taken over all possible measurement sequences, weighted with their respective probabilities.

Eq. (2) is valid formally even if the overall evolution is non-Markovian. It can be simplified in the important Markovian case. Then, evolution proceeds step-wise. Let us denote by Φ_j the CP map for the continuous evolution during the time interval $[t_j^+, t_{j+1}^-]$, where t^- is shorthand for a time point just prior to the measurement at t, and correspondingly t^+ is right after the measurement. Then we have $\hat{\rho}(t_{j+1}^-) = \Phi_j(F_\theta^j(m))[\hat{\rho}(t_j^+)]$. In the special case of unitary dynamics, the evolution itself simplifies further to $\hat{\rho}(t_{j+1}^-) = \hat{U}_j(F_\theta^j(m))\hat{\rho}(t_j^+)\hat{U}_j(F_\theta^j(m))^\dagger$. Here $\hat{\rho}(t_{j+1}^-)$ is understood to be the quantum state at time t_{j+1} for a fixed sequence $m_1, \ldots, m_j$ of previous measurement outcomes, just prior to the next positive-operator-valued measure (POVM) measurement implemented at t_{j+1}.

This measurement is described by some POVM element that can be written in the form $\hat{M}(m')^\dagger \hat{M}(m')$, with the POVM normalization condition $\sum_{m'} \hat{M}(m')^\dagger \hat{M}(m') = 1$ and $\hat{M} \equiv \hat{M}_{j+1}$ depending on the physics of the measurement. It will yield a particular outcome $m_{j+1} \equiv m'$ with probability $P(m') = \mathrm{tr}[\hat{M}(m')^\dagger \hat{M}(m')\hat{\rho}(t_{j+1}^-)]$ and an updated state $\hat{\rho}(t_{j+1}^+) = \hat{M}(m')\hat{\rho}(t_{j+1}^-)\hat{M}(m')^\dagger / P(m')$.

Our goal is to maximize some overall cumulative reward $\mathcal{R}$, which is called "return" in the nomenclature of reinforcement learning. For example, in a state-preparation task this might be the final fidelity with respect to some target state $\hat{\sigma}$. For a given sequence m of outcomes, we would define $\mathcal{R}(m) = \left(\mathrm{tr}\sqrt{\sqrt{\hat{\sigma}}\hat{\rho}(T|m)\sqrt{\hat{\sigma}}}\right)^2$. This would be averaged eventually over all possible measurement outcome sequences to yield $\bar{\mathcal{R}} = \langle \mathcal{R}(m) \rangle_m$. The return $\mathcal{R}$ could also involve penalties for suppressing larger control amplitudes etc. These additional contributions depend on the specific sequence m as well, via the controls $F_\theta^j(m)$.

It might now seem straightforward to employ automatic differentiation for optimizing $\bar{\mathcal{R}}$ via gradient ascent, updating $\delta\theta = \eta \frac{\partial \bar{\mathcal{R}}}{\partial \theta}$, with some learning rate η and with all parameters combined in a vector θ.

The crucial observation to be made at this stage is that the introduction of stochastic measurement results into this scheme requires some extra care. The following considerations constitute the main conceptual steps needed to enable the discovery of feedback-based quantum control strategies based on gradient ascent.

We have to distinguish between discrete and continuous measurement outcomes, which require substantially different treatment.

For the particularly interesting discrete case (e.g. strong projective qubit measurements), the essential insight is that the probabilities P for obtaining the different measurement outcomes themselves depend on all the controls F_θ^j applied during previous time intervals, simply because the quantum state itself carries this dependence. This has to be taken care of during the evaluation of gradients with respect to θ. Illustrating this in the case of a single measurement at time $t_1 \in [0,T]$, we have

$$\langle \mathcal{R} \rangle_m = \sum_m P(m|\hat{\rho}(t_1^-))\mathcal{R}(\Phi_1(F_\theta^1(m))[\hat{\rho}(t_1^+)]) \quad (3)$$

Here $P(m|\hat{\rho})$ is the probability for measurement outcome m given state $\hat{\rho}$. As we take the gradient with respect to the parameters θ, we observe that the derivative acts not only on the return $\mathcal{R}$ based on the time-evolved state (the second factor inside the sum) but also on the probability $P(m)$ itself, due to its dependence on the initial control, $\hat{\rho}(t_1^-) = \Phi_0(F_\theta^0)[\hat{\rho}(0)]$.

Generalizing this observation, we cannot simply implement gradients of the measurement-averaged return $\bar{\mathcal{R}} = \langle \mathcal{R}(m) \rangle_m$ by averaging the gradient of the sequence-specific return, $\langle \partial \mathcal{R}(m)/\partial\theta \rangle_m$. Rather, observe $\langle \mathcal{R}(m) \rangle_m = \sum_m P(m)\mathcal{R}(m)$. Thus, when evaluating $\partial \langle \mathcal{R}(m) \rangle_m / \partial\theta$, we will get two contributions: $\partial[\mathcal{R}(m)P(m)]/\partial\theta = P(m)\partial\mathcal{R}(m)/\partial\theta + \mathcal{R}(m)\partial P(m)/\partial\theta$. To enable stochastic sampling of the second term, we rewrite it using $\partial P(m)/\partial\theta = P(m)\partial \ln P(m)/\partial\theta$. This then leads to:

$$\frac{\partial \langle \mathcal{R}(m) \rangle_m}{\partial\theta} = \left\langle \frac{\partial \mathcal{R}(m)}{\partial\theta} \right\rangle_m + \left\langle \mathcal{R}(m) \frac{\partial \ln P_\theta(m)}{\partial\theta} \right\rangle_m . \quad (4)$$

Here we displayed explicitly the parameter-dependence of $P_\theta(m)$, which represents the probability of the full sequence of outcomes $m = (m_1, m_2, \ldots)$, given the parameters θ that determined the shape of the control functions F_θ^j.

The mathematics for the extra term appearing here, with the gradient of the log-likelihood, is well known from policy-gradient-based approaches in *model-free* reinforcement learning. However, there this term appears for a different reason. It arises due to the deliberate

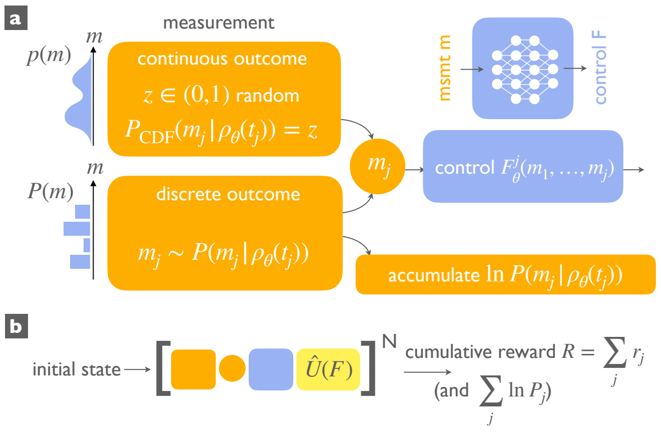
a
measurement
$p(m)$
m
continuous outcome
$z \in (0,1)$ random
$P_{\mathrm{CDF}}(m_j | \rho_\theta(t_j)) = z$
$P(m)$
m
discrete outcome
$m_j \sim P(m_j | \rho_\theta(t_j))$
m_j
msmt m
control F
control $F^j_\theta(m_1, \ldots, m_j)$
accumulate $\ln P(m_j | \rho_\theta(t_j))$
b
initial state
$\hat{U}(F)$
N
cumulative reward $R = \sum_j r_j$
(and $\sum_j \ln P_j$)

FIG. 2. Quantum feedback sequences considered within feedback-GRAPE, set up for automatic differentiation. (a) The measurement samples a stochastic outcome m_t, adopting a different method depending on whether the outcome is continuous or discrete. In both cases, the probability distribution depends (in a differentiable way) on the learnable parameters θ, via the preceding unitary controls that have generated the present quantum state ρ_θ. Depending on the measurement outcome, a learnable control F is applied that may be implemented either via a neural network or a lookup table. (b) Full sequence. This consists of repeated application of the blocks depicted in (a), plus subsequent implementation of unitary controls depending on F, potentially with decay and decoherence included in the model of the system's evolution. When optimizing the overall return $\mathcal{R}$ using gradient ascent, a correction term needs to be taken into account (in the case of discrete outcomes).

choice of implementing stochastic controls, in order to avoid any need to take gradients through the possibly unknown dynamics of the system to be controlled (and also in order to treat discrete actions without giving up gradients). In our case, by contrast, we do take gradients through the known dynamics and the controls themselves are deterministic when conditioned on a fixed sequence of measurements. The randomness enters via the stochastic measurement outcomes (these are observations of the "environment" in RL language).

Due to the sequential nature of the control procedure, the log-likelihood term can be rewritten as a sum of contributions, $\ln P_\theta(m) = \sum_j \ln P_\theta(m_j | m_{j-1}, \ldots, m_1)$. Thus, during the individual time evolution trajectory, this term may be easily accumulated step by step, since the conditional probabilities are known (these are just the POVM measurement probabilities). The gradients of Eq. (4) can then be taken for such an individual trajectory (or rather a batch), substituting stochastic sampling for an exact average over m. The whole approach, with its calculational pipeline, is schematically illustrated in Fig. 2.

We note in passing that there is a special case in which things can be simplified: if the return $\bar{\mathcal{R}}$ is defined only to depend on the average state $\langle \hat{\rho}(T) \rangle_m$ itself, we can exploit that this state can be obtained directly as a sum over measurement results, without any probability weights: in the example above, one finds $\langle \hat{\rho}(T|m) \rangle_m = \sum_m \Phi_1(F_1(m))[\hat{M}(m)\Phi_0(F^0_\theta)[\hat{\rho}(0)]\hat{M}(m)^\dagger]$. In this expression, formally no normalization factors are needed, and gradients could be taken directly. However, not only does this approach preclude Monte-Carlo evaluation of the sum via stochastic sampling (which is important for efficiency in the case of long measurement sequences or many outcomes), but this assumption about $\bar{\mathcal{R}}$ is also very restrictive. It excludes even some of the most important cases, like a return that tries to maximize the average fidelity instead of the fidelity of the average state. It also excludes many useful contributions to the return, such as those based on control amplitudes (that depend on m), as well as, for example, penalties for large fluctuations of the sequence-specific fidelity, or in general the return being an average of some nonlinear function of the density matrix, such as the purity $\langle \mathrm{tr}\hat{\rho}^2 \rangle_m$. We will therefore not consider this special case further.

The evaluation of the gradients of the return with respect to the parameters θ can proceed in two different ways, using either automatic differentiation (see below) or exploiting analytical approaches to obtain explicit expressions for the gradients that can then be evaluated numerically. In the latter case, one can either set up evolution equations for the parameter-gradient of the quantum state, $\partial_\theta \hat{\rho}$ or, in the suitable scenario, directly apply a modified version of the original GRAPE technique to efficiently evaluate the gradients. We describe both of these procedures in detail in appendix A. In the language of current machine learning concepts, taking the gradient through the continuous-evolution intervals would be generally speaking an example of the concept of neural ordinary differential equations, a rather recent development [44].

Alternatively, and sometimes more conveniently, the whole evolution pipeline described above can straightforwardly be implemented in an automatic differentiation framework, such as TensorFlow[45], PyTorch, JAX, or others. Gradients of the resulting overall return and of the log-probability can then be obtained using that framework without extra effort. The sequence of discrete measurement outcomes of a given trajectory is considered fixed when taking the gradient in this manner. The automatic-differentiation approach is particularly helpful and efficient in cases where the whole time evolution can be split into many building blocks (parametrized gates, i.e. unitaries, acting during fixed time intervals), as is common practice for many quantum control tasks in present quantum computing platforms. Whenever this latter situation is encountered, it also aids interpretability, as we will see in the numerical examples.

As we remarked at the beginning, the central quantity of our approach are the measurement-dependent controls $F^j_\theta(m_1, m_2, \ldots, m_j)$. For accessing those, one can simply adopt a lookup table, at least for the case of discrete measurements discussed up to now and when the total number of measurements during the full time evolution is not too large. The table for F_θ^j needs M^j entries, if there are M possible outcomes for each measurement, corresponding to the exponentially many possible sequences. In that case, the entries of this table would directly represent the parameters θ for which we perform gradient ascent. Alternatively, the controls $F_\theta^j(m)$ can be implemented via a neural network that takes measurement results as input and maps those to the current control vector. Since the number of available measurement results is different for each time step j, one may choose to set up a different network at each j. However, training efficiency and generalization ability can be improved by constructing a single recurrent network, i.e. a network with memory that is employed in sequence processing tasks [46]. It takes the temporal sequence of measurements as input, one step at a time, producing a control vector at each such time step. This approach can possibly generalize to infinitely long feedback control sequences, for example during state stabilization tasks. An alternative, also network-based approach replaces the measurement results by the full current quantum state, conditioned on these results. This carries the same information, and it might sometimes be easier for the network to learn. In the course of our numerical experiments, to be detailed later, we did not observe a single approach to outperform the others in all scenarios.

Continuous measurement outcomes can be treated in exactly the same way as discrete ones. However, for that scenario there also exists an alternative, which obviates the need for the logarithmic-likelihood correction term: we can adopt a general version of what is known as the 'reparametrization trick' in stochastic neural networks (e.g. in variational autoencoders). The idea is that we can generate a stochastic variable z according to some *fixed* probability density and then transform this into the required measurement probability density $p(m|\hat{\rho})$, which does depend on control parameters (via the state $\hat{\rho}$, as explained above) and must be subjected to gradients. This parameter-dependent transformation can be implemented in a differentiable way, as we now show. We first obtain the cumulative distribution function $f(m) = \int_{-\infty}^{m} p(m'|\hat{\rho})dm'$, by discretizing p as a vector on a lattice and using a cumulative sum for an Euler approximation of the integral (this operation exists in frameworks like TensorFlow). We then draw a random uniformly distributed $z \in [0,1]$ and *invert* $f(m)$. The last step also needs to be performed in a differentiable way. One option is to set $m = f^{-1}(z) \approx \sum_n \tilde{m}_n H(z - z_n)H(z_{n+1} - z)$. Here $z_n = f(m_n)$ defines the lattice version of f, H is the Heaviside step function, the sum ranges over the lattice points, and $\tilde{m}_n$ solves the piecewise linearized approximation of $m = f^{-1}(z)$ associated with the interval n: $\tilde{m}_n = (m_{n+1} - m_n)(z - z_n)/(z_{n+1} - z_n) + m_n$. The set of measure zero where the gradient is undefined can be ignored (as is common practice in using activation functions like rectified linear units in neural networks).

In this way, one can implement, within the automatic differentiation framework, for example measurements of discrete variables with continuous outcomes. A typical case would be a qubit measurement with $m = \sigma + \xi$, where $\sigma = \pm 1$ is the qubit state and ξ some measurement noise of density $q(\xi)$. Formally, $p(m|\hat{\rho}) = \sum_\sigma q(m - \sigma)\rho_{\sigma\sigma}$, and $\hat{M}(m) = \sum_\sigma \sqrt{q(m-\sigma)}\,|\sigma\rangle\langle\sigma|$. One can also perform measurements on continuous variables, e.g. a weak measurement of position, $p(m|\hat{\rho}) = \int dx q(m - x)\rho(x,x)$, with $\hat{M}(m) = \int dx \sqrt{q(m-x)}\,|x\rangle\langle x|$. The dependence of the probability density p in each case on the parameters determining the control functions at earlier times will be correctly taken into account, and one can now use the straightforward formula $\partial\bar{\mathcal{R}}/\partial\theta = \langle \partial\mathcal{R}(m)/\partial\theta \rangle_m$ for stochastic sampling of the gradient. Note that the discrete-outcome case (above) and the continuous-outcome case can also be easily combined in our approach.

So far, controls have been continuous and represented via functions (differentiable with respect to parameters) depending on previous measurement results. However, sometimes one might want to *also* take discrete actions, e.g. deciding whether some measurement should be performed at all or not, or whether some fixed qubit gate should be applied. This can be incorporated without any substantial changes to the approach discussed here, borrowing from policy-gradient model-free reinforcement learning, by introducing stochastic actions a, in contrast to the deterministic continuous actions discussed so far: use a network or a lookup-table to calculate the probability $P(a|m)$ of taking a discrete action a given the previous measurement record m and then sample from all actions accordingly. Now a new log-likelihood term $\ln P(a|m)$, stemming from these action probabilities, needs to be accumulated and treated in the same way in the overall gradient ascent as explained for stochastic discrete measurement outcomes above.

III. NUMERICAL EXAMPLES

We now turn to an illustration of the feedback-GRAPE method by solving several different quantum feedback control tasks in a paradigmatic scenario relevant for quantum technologies. For this purpose, we chose the Jaynes-Cummings model, i.e. a qubit coupled to a cavity. This is a well-known system that first emerged as the simplest light-matter coupling scenario in quantum optics [47, 48] but is nowadays of practical relevance for modern quantum-computing platforms [49]. In those, it is employed both for qubit-readout and for qubit-enabled nonlinear manipulation of cavity states.

The Jaynes-Cummings model has the additional advantage of featuring an exact solution for a specific quantum control task, still without feedback: State preparation of arbitrary cavity states with the help of the qubit, in the absence of noise. In a groundbreaking work [50], Law and Eberly showed that this can be achieved by employing a sequence of steps, each of which involves a rotation of the qubit by some angle, followed by a qubit-cavity

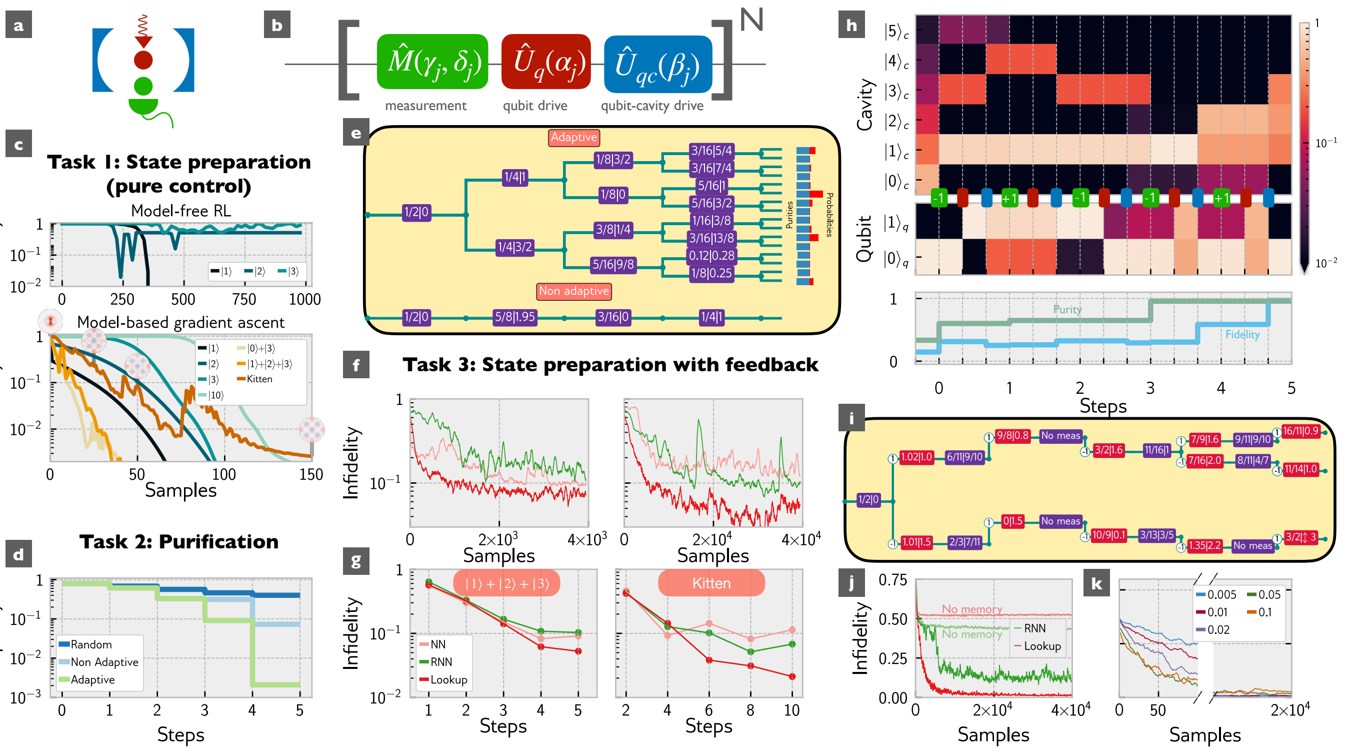

FIG. 3. Numerical examples: Feedback control tasks in a cavity-qubit system. a) Schematic of the Jaynes-Cummings system with an additional ancilla qubit used for the measurement. b) Sequence of parametrized controls inside one step, to be repeated N times. c) State preparation out of the ground state (pure control, no feedback), with target states indicated. Model-free RL performs very poorly, while the direct gradient-based approach used as the basis for our method converges well. "Samples" refers to the number of trajectories generated for gradient ascent. d) Purification of a thermal state (here, with $\bar{n}=2$). An adaptive strategy (found using feedback-GRAPE) clearly outperforms other approaches. e) Extracted purification strategy visualized in a decision tree. The purple boxes display the measurement parameters $(\gamma/\pi|\delta/\pi)$. f)-i) State preparation from a thermal state ($\bar{n}=1$), employing feedback. f) Gradient-ascent progress for two target states (the curves are smoothed with a moving average), and g) final infidelity vs total number of time steps. Each point is the best out of 30 runs. h) Evolution of reduced qubit and cavity state (probability as color) for one trajectory of the converged strategy (target $|1\rangle+|2\rangle+|3\rangle$); time points of measurements (with results) and controls are indicated as in (b). i) Corresponding decision tree, for the most probable sequences of measurement outcomes. The red boxes show $(\alpha/\pi|\beta/\pi)$, and "no meas" means the parameters γ,δ are such that no measurement takes place. j) Gradient-ascent progress with and without memory. k) Effect of the learning rate (see legend) during the gradient ascent.

interaction of variable duration. The angles and interaction durations of this sequence form a set of parameters, which can be found exactly using the Law-Eberly algorithm. This solution has been used to remarkable effect in experiments with superconducting qubits [51]. It will serve us as a convenient benchmark for the simpler case of pure control, forming the starting point of our exploration. In addition, the optimization of control pulses (without real-time feedback) in the Jaynes-Cummings model has been explored using GRAPE [12, 52].

However, our interest in the present work is focused on the application of feedback. Feedback is required in the presence of noise, e.g. when the initial state is mixed or when decay or decoherence are present. In those cases, feedback helps to remove entropy from the quantum system. To address this, we will extend the Jaynes-Cummings setting in a suitable way. This can be done, for example, by coupling to the cavity some ancilla qubit, which can then be read out to update our knowledge of the cavity's quantum state. In that scenario, no protocol is known and suitable feedback strategies have to be discovered from scratch.

The complete physical situation is shown in Fig. 3a,b. A cavity is coupled to both a "control qubit" and an ancillary "measurement qubit". The control qubit "c" can be driven externally to implement an arbitrary rotation around an equatorial axis, implementing the unitary gate $\hat{U}_q(\alpha) = \exp\left[-i\left(\alpha\hat{\sigma}_+^c + \alpha^*\hat{\sigma}_-^c\right)/2\right]$. Afterwards, qubit and cavity mode $\hat{a}$ can be coupled for a variable duration, exchanging excitations, $\hat{U}_{qc}(\beta) = \exp\left[-i\left(\beta\hat{a}\hat{\sigma}_+ + \beta^*\hat{a}^\dagger\hat{\sigma}_-\right)/2\right]$.

Subsequently, a measurement can take place (this is where we deviate from the basic pure-control Law-Eberly scenario). The measurement comprises several steps, which we will list individually before summarizing their combined effect on the cavity state. In a first step, the ancilla qubit "a" is prepared in the $+x$ eigenstate. Subsequently it is coupled dispersively to the cavity for a variable amount of time: $\hat{U}(\gamma) = \exp\left(-i\gamma\hat{\sigma}_z^a\hat{a}^\dagger\hat{a}\right)$. This means the qubit precesses by an angle that depends linearly on the number of photons inside the cavity. In the next, final step, the ancilla qubit is projected along some selected axis $\hat{\sigma}_x^a\cos\delta + \hat{\sigma}_y^a\sin\delta$, yielding a discrete result $m \in \{-1, +1\}$. This type of measurement (originally proposed in [53]) was indeed an important component e.g. in several experiments on Rydberg atoms interacting with microwave cavities [54]. The combined effect of these operations is to perform a POVM on the cavity, with outcome probability $P(m) = \mathrm{tr}[\hat{M}(m)^\dagger\hat{M}(m)\hat{\rho}]$ and an updated state $\hat{M}(m)\hat{\rho}\hat{M}(m)^\dagger/P(m)$. Here $\hat{\rho}$ is the state of the qubit-cavity system, excluding the measurement qubit which has been eliminated in this description. The measurement operator $\hat{M}(m)$ is given by

$$\hat{M}(m = +1) = \cos(\gamma\hat{a}^\dagger\hat{a} + \delta/2)\,, \tag{5}$$

and likewise for $m = -1$, with cos replaced by sin. This formula indicates that after the measurement the probabilities of the different cavity Fock states $|n\rangle$ will be multiplied by a sinusoidal "mask", where the period is determined by $1/\gamma$ and the phase shift is set by both δ and the measurement outcome m. This helps to pinpoint the state of the cavity, especially when multiple such measurements are carried out with suitably chosen periodicities [53] and phase shifts [54].

The set of parameters $\alpha, \beta, \gamma, \delta$ mentioned here form the control vector F. Its value will be different for each step j of the feedback-control sequence and, in particular, it will depend on previous measurement results: $F_\theta^j = F_\theta^j(m_1, \ldots, m_j)$. As explained above, in the feedback-GRAPE approach, F will be implemented either as a lookup table or as a neural network. As the number of measurements increases with time, it is most convenient to employ a single recurrent neural network.

In our numerical explorations, we will consider four separate tasks of increasing difficulty: starting with noiseless state preparation (a pure control task) as a baseline benchmark for GRAPE-type control in this scenario, then moving to purification (a task that already benefits from feedback, i.e. adaptive measurements), to feedback-based state preparation in the presence of noise and feedback-based state stabilization.

Preparing any (pure) cavity state from the ground state has the known Law-Eberly solution (reviewed in appendix B). This can serve as an interesting benchmark to explore how well different numerical techniques perform. We set the return $\mathcal{R}$ equal to the state fidelity at the final time step, prescribing a fixed number of time steps. Even in this simple setting, we encounter a first surprise: State-of-the-art model-free reinforcement learning is not able to cope well with this challenge. We employed proximal-policy optimization (PPO) [55], a powerful and widely used modern general-purpose advantage actor-critic approach, to generate the continuous controls. It performs well only for the very simple task of preparing Fock state $|1\rangle$, while getting stuck at bad final overlaps for higher Fock states. This statement holds even after training for many episodes and varying the hyperparameters, and even for other modern general model-free RL algorithms that we tried (see Appendix C). In fact, the poor performance of these nominally powerful techniques in this setting was one of the initial motivations for our development of the general feedback-GRAPE method introduced above.

In contrast, very good results are efficiently achieved, without particular effort, by using direct gradient ascent through the unitary evolution in the spirit of GRAPE (in our case, using the automatic differentiation framework of TensorFlow and a well-known modern adaptive gradient optimizer, 'Adam'). In the present example, for the specific case of pure-state preparation, we found that a neural network being fed the current quantum state as input converges better than gradient ascent on the control parameters themselves, i.e. better than what we have termed the 'lookup table' approach (the network-based results are shown in Fig. 3c). Gradient ascent allows to find optimal state preparation strategies performing as well as the known Law-Eberly algorithm even for complex superpositions of Fock states, e.g. a four-component cat state with $\bar{n} = 9$ (cf Fig. 3c). To mitigate local minima, repeated runs may be necessary (see appendix E).

Regardless of these detailed observations, this example indicates that model-based gradient ascent approaches can outperform model-free generic methods for optimizing quantum control in settings relevant for quantum technologies. Given the large performance difference already in this simple control scenario, we focused entirely on the feedback-GRAPE approach in the subsequent exploration of the more advanced challenges that do include feedback.

Moving on to a first example of a situation that requires feedback, we will now imagine that the cavity is initially in a mixed state. The goal will be to purify the cavity's state, i.e. the reward is determined by the purity $\mathrm{tr}\hat{\rho}_{\mathrm{cav}}^2$ of the cavity state at the final time. In this case, we assume there are no qubit-cavity controls (α and β are not used), but the measurement choices (determined by δ and γ) can be adaptive, i.e. depend on previous measurement results.

Fig. 3d shows the results of applying the feedback-GRAPE method to this problem (labeled 'Adaptive'). We employ a recurrent neural network to produce the controls F when provided with the measurement outcome sequence (more details on numerical parameters can be found in appendices D,E). As we see, the impurity quickly decreases with the number of allowed measurements, and it does so significantly better than in a non-adaptive scheme, where the sequence of measurement controls δ_j and γ_j is still optimized, but where these controls are not allowed to depend on previous measurement outcomes. To visualize and analyze the numerically obtained strategy, we introduce in Fig. 3e a decision tree.

This is extracted via an automated numerical procedure, by running many trajectories and noting in each case the controls suggested by the adaptive strategy. The controls are a deterministic function of previous measurement outcomes. Such a decision tree will contain all information about the adaptive strategy learned by the NN and can possibly allow the user to give it a physical interpretation and extrapolate analytical solutions for large numbers of control steps. This might require to leverage any available physical understanding of the control operations, e.g. identifying physically significant values of the control parameters. Using our understanding of the model's physics, we can choose to (programmatically) interpret the controls, e.g. trying to represent them in terms of fractional multiples of π or of $\pi/\sqrt{n}$ (Fig. 3i has an entry marked $\updownarrow 3$ which stands for $\pi/\sqrt{3}$). This kind of analysis is optional, and independent of our method, but it nicely demonstrates what can be usefully done in settings with discrete measurements, generating additional insights after running the general-purpose algorithm. For example, here, we were able to take inspiration from the decision tree for four measurements and a specific value of the temperature to extrapolate the optimal purification strategy for any temperature and any number of measurements, see appendix F.

We now turn to a task that involves both feedback and control simultaneously. Specifically, we consider state preparation out of a thermal state, for target states that are selected as arbitrary superpositions of the first few Fock states. Results for the state $(|1\rangle+|2\rangle+|3\rangle)/\sqrt{3}$ and a four-component kitten state, built from four coherent states with $\bar{n}=2$, are shown in Fig. 3f,g. Feedback-GRAPE converges in about 1000 gradient-ascent steps (each operating on a batch of 10 sampled trajectories). We ran the method several times, starting with different initial random configurations of the parameters θ, demonstrating that convergence is robust, despite the usual absence of a guarantee for such a non-convex optimization problem.

It is interesting to analyze in some more detail the convergence behaviour. As one noteworthy observation, despite the overall very good performance, we sometimes find that the algorithm may get stuck at suboptimal solutions if we increase the total number of time steps available for the feedback sequence (Fig. 3g). Ideally, an increased number of steps should always lead to an improvement (in the present scenario), but apparently the larger space of control variables then becomes challenging. This can be mitigated to some extent by running the gradient ascent repeatedly from random starting conditions.

One motivation for the use of a neural network instead of a lookup table is that the number of parameters needed for a tree-type table grows exponentially, while a neural network could in principle make use of a much smaller number of parameters. Also, it may be expected that the strategy of a network generalizes to situations with a number of time steps larger than the one it was trained on. Despite these obvious advantages of neural networks, we found (to our surprise) that lookup tables often converge to better fidelities than networks, in the present example scenario with feedback. This is evident in Fig. 3f,g, where we compare three different choices (a recurrent net, RNN, receiving the measurement sequence, a fully-connected NN, receiving the quantum state, and the lookup table). The reasons for this are still unclear and merit future investigation.

We note in passing that the difficulty of the control problem depends of course on the power of the controls available. The Jaynes-Cummings setting explored here turns out to be much more challenging than, e.g., the state-preparation tasks explored recently in [35] with the help of model-free RL, which relied on much more powerful controls.

What is the nature of the feedback strategies that the algorithm discovers? Naively, we might expect the following strategy: an optimized adaptive purification phase, of the kind discussed above, leading to some Fock state $|n\rangle$, followed by state preparation that is derived from the Law-Eberly protocol (e.g. going back down to the ground state and then building up the arbitrary target state from there). However, the actual strategies discovered by feedback-GRAPE are significantly more efficient. They interleave adaptive measurements and controls already in the first stage of the process. This can be seen in Fig. 3h,i, where the goal was to prepare the equal superposition $(|1\rangle+|2\rangle+|3\rangle)/\sqrt{3}$. Again, it is possible to obtain more information about the full strategy (as opposed to a single trajectory), by extracting a decision tree (Fig. 3i). There, we observe that measurements are sometimes deliberately performed in such a way that certain Fock states are completely ruled out (their probability is set to 0), which requires certain choices of measurement control parameters. Simultaneously, qubit-cavity interaction cycles are employed to reduce the excitation number of the cavity.

Quantum state stabilization in the presence of dephasing or decay represents another challenging task that can be investigated. Some numerical results are shown in Fig. 4, obtained for the stabilization of a four-legged kitten state (with average photon number $\bar{n}=9$) against cavity decay. Here the cavity decay rate was set to $\gamma\Delta t=0.05$, with Δt the duration of one time step.

As we explained above, lookup tables often perform surprisingly well. We now briefly demonstrate, in the context of state stabilization, one example where the power of a neural network is clearly helpful (Fig. 4e). We first train an RNN on sequences of 20 steps, with the goal to stabilize a given Fock state for an arbitrarily long time. For this example, the cumulative reward of a trajectory is not only the final fidelity, but the sum of fidelities at all time steps. After training, we test on a 10 times longer simulation, and we see that the strategy learned by the RNN generalizes well even for longer sequences. We note how the strategy can recover, even when some "unlucky" measurement outcomes significantly perturb

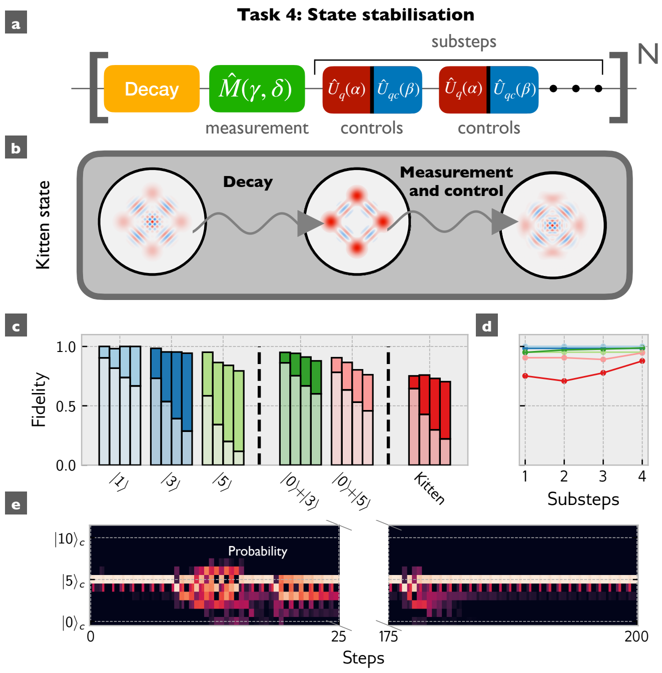

FIG. 4. State stabilization with Feedback-GRAPE. a,b) Sketch of the feedback control sequence, applied after physical decay of the cavity. We assume that multiple control substeps can be applied after a single measurement. c) Performance of the strategy found by feedback-GRAPE. For various target states, we show the time evolution of the fidelity. After each decay and measurement, a single control sequence (i.e. only one choice of α and β) is applied. The four columns represents different numbers of decay steps experienced by the state (here: N=1, 2, 3 and 4); the fidelity for each step is calculated after the controls have been applied. The bars with lower values show the bare decay of the fidelity, when no feedback strategy is employed. d) The fidelity can increase by applying more control substeps after decay and measurement; again shown for the different target states (labeled via colors), for $N = 1$. e) Stabilization of a Fock state (here $|5\rangle$) for an arbitrarily long time, employing the generalization ability of a recurrent neural network (RNN).

the quantum state.

IV. EXTENSIONS

Before concluding our discussion, we outline possible extensions of the general feedback-GRAPE technique introduced above.

A. Reducing sampling noise by using a value function

The average of the return over different measurement outcome sequences is obtained by sampling, which introduces noise into the estimate of the gradients. We can help suppress the noise by adopting value function approaches that are known as a general technique in reinforcement learning [27].

To start, we need to discuss the structure of the rewards more carefully. Above, we introduced the overall return (cumulative reward) as the quantity to be optimized. We can also assign the rewards more specifically to individual time steps. For example, during state stabilization we can evaluate the fidelity at each time step and sum it over time to obtain the return. Likewise, it is customary in some optimal control settings to punish large control amplitudes at any given time step. In all these cases, the return is a sum $\mathcal{R} = \sum_{j=1}^{N} r_j$ of individual rewards.

More precisely, in the original approach, we had simply set $\mathcal{R} = r_1(m_1|\theta) + r_2(m_2, m_1|\theta) + \ldots$. Here $r_j(m_j, m_{j-1}, \ldots|\theta)$ is the instantaneous reward obtained after time step j (which consisted of some control, some measurement yielding m_j, and possibly a further control step before assigning the reward). For any time step j, this then yields two contributions to the overall gradient ascent update. For example, at $j = 2$ we obtain, in a given trajectory with randomly sampled $m_1, m_2, \ldots$ the following contributions:

$$\partial_\theta r_2(m_2, m_1|\theta) + \partial_\theta \ln P(m_2|m_1, \theta) \cdot \{r_1(m_1|\theta) + r_2(m_2, m_1|\theta) + r_3(m_3, m_2, m_1|\theta) + \ldots\} \quad (6)$$

Adding up these contributions for all j and averaging over trajectories yields precisely Eq. (4).

This is a Monte-Carlo sampling approach. One concern in any such approach is the sampling noise, i.e. in our case the fluctuations of the quantity shown above between different trajectories. We can now take inspiration from the domain of model-free reinforcement learning and the general theory of reinforcement learning [27], where approaches have been invented to reduce the variance in estimations of the gradient update. Recall that in our case, the variance stems from the stochasticity of measurements, whereas in model-free RL it stems from the stochasticity of policy action choices that is encountered in policy-gradient and actor-critic approaches, plus any stochasticity of the environment dynamics. Even though the following steps follow very closely the corresponding tricks known in the model-free RL community, we display them explicitly here, for our modified scenario. This should help avoid any confusion and make this presentation self-contained.

First, when evaluating the gradient above, we need only include the sum of future rewards, since only those can be influenced by the present measurement result. In the example of Eq. (6), this means the term $r_1(m_1|\theta)$ on the second line may be dropped, as it is independent of m_2, i.e. the new measurement result. Mathematically, this follows because when we eventually perform the average over trajectories, we have to multiply Eq. (6) by $P(m_2, m_1|\theta) = P(m_2|m_1, \theta)P(m_1|\theta)$. Collecting terms, the m_2-dependency for the r_1 contribution ends up in a sum $\sum_{m_2} \partial_\theta P(m_2|m_1,\theta)$. This sum turns out to be zero due to the normalization of the conditional probability for any value of θ. This insight holds for any j, where it is used to drop all r_k $(k < j)$ when they multiply $\partial_\theta \ln P(m_j|m_{j-1},\ldots,\theta)$.

Second, to further suppress stochastic fluctuations one can learn a value function V, which is a function of the current state and represents the expected future cumulative reward, averaged over all possible future measurement outcomes. Thus $V(m_j, m_{j-1},\ldots|\theta)$ is defined to be

$$E(r_{j+1} + r_{j+2} + \ldots |m_j, m_{j-1},\ldots,\theta)\,,$$

where the label E stands for the expectation value over future rewards, conditioned on the preceding measurement results.

Typically, V would be expressed as a neural network, though a lookup table can also be used in the case of a (modest) number of discrete measurements. The input to the value network would be some representation of the current "state" s. This state could be identified directly with the sequence of previous measurement results, as indicated in our notation above, $s_j = m_j, m_{j-1}, \ldots$ (which uniquely determines the current state). Alternatively, this state could also be represented by some version of the current quantum state (e.g. the density matrix), if that proves easier to handle for the network. The value network would be trained to output the expected (averaged) future cumulative reward, counted from this state onwards. The value training would proceed in the fashion known from general reinforcement learning, i.e. using the Bellman update equation [27] $V^{\text{new}}(s_j) = V(s_j) + \alpha(r_j + \gamma V(s_{j+1}) - V(s_j))$, with $\alpha < 1$ some update factor and $\gamma \leq 1$ some discount factor to reduce the weight of long-term rewards ($\gamma \to 1$ in the ideal case discussed up to now). When using a neural network, V^{new} would be the new target value for the value network during a supervised-learning update. Once an approximation to the value function has been learned in this manner, we can proceed as in advantage actor-critic approaches to model-free RL. This means that in the gradient ascent procedure of the feedback-GRAPE approach, one would replace the (future) return by the *advantage* $A_j = r_j + \gamma V(s_{j+1}) - V(s_j)$, which expresses the improvement over the currently expected future return. In effect, this reduces the variance of the gradient estimates by subtracting a convenient baseline, without changing the average gradient update.

Concretely, Eq. (6), the gradient contribution from time step $j = 2$, would be replaced by the following:

$$\begin{aligned}&\partial_\theta r_2(m_2, m_1|\theta) + \partial_\theta \ln P(m_2|m_1,\theta)\cdot \\ &\{r_2(m_2,m_1|\theta) + \gamma V(m_2,m_1|\theta) - V(m_1|\theta)\}\end{aligned} \qquad (7)$$

The first line is unchanged, but in the second line r_1 was dropped, as explained before. Moreover, the sum of $r_3 + r_4 + \ldots$ has been replaced by $\gamma V(m_2, m_1|\theta)$, which is the expectation of the future return (such that averaging over $m_3, m_4, \ldots$ has already been carried out, reducing sampling noise). Finally, $V(m_1|\theta)$ was subtracted, to reduce further the variance by canceling the expected value, given m_1. This is possible for the same reason that we could drop $r_1(m_1)$, as explained above. The extension to arbitrary $j \neq 2$ is obvious.

In summary, such an enhanced feedback-GRAPE method would run trajectories with deterministic continuous controls and stochastic discrete quantum measurements just as before. However, it would learn a value function to represent expected future returns, and it would use that value function to modify the gradient ascent procedure and reduce fluctuations.

B. Multi-target quantum feedback control

Whenever we are employing neural networks to represent the feedback-based controls, a straightforward but powerful extension of feedback-GRAPE suggests itself. We may feed a representation of a variable target state Ψ (or, in general, the target task, however it is defined) into the network: $F_j(\theta_j, m_j, \ldots; \Psi)$. The whole feedback-control strategy is then trained on many different randomly chosen tasks (e.g. many possible target states).

Such approaches have been successul recently for other control challenges, e.g. they are being investigated in robotic navigation and the general field of multi-target reinforcement learning [56, 57]. Multi-target schemes have also been recently suggested to improve variational quantum circuits [58]. The benefit is data-efficiency: the network learns to generalize from the training tasks to other similar tasks, which requires less overall effort than to retrain a freshly initialized network for each task.

V. CONCLUSIONS AND OUTLOOK

In this work, we have presented a general scheme for the direct gradient-based discovery of quantum feedback strategies. This scheme, which we have labeled feedback-GRAPE, works for arbitrarily strong (discrete or continuous) nonlinear stochastic measurements, which so far had been possible only using the less data-efficient approaches of model-free reinforcement learning.

We observed very encouraging performance when testing the method on a challenging set of feedback tasks in a prototypical quantum-optical scenario. Overall, our method opens a new route towards solving challenging feedback-based control tasks, including tasks in quantum communication and quantum error correction on multi-qubit or qubit-cavity systems. Besides presenting and analyzing the basic approach, we have also discussed extensions such as advantage functions (for reducing sampling noise) and training on multiple targets (to increase data efficiency and exploit transfer learning).

Appendix A: Evaluation of the parameter gradients of the time-evolving quantum state

In the numerical results in the main text, we have employed automatic differentiation to evaluate parameter gradients, which is very convenient using modern machine learning tools. However, alternatively, it is also possible to directly work out analytical formulas to evaluate such gradients, based on our knowledge of the evolution equations. In a particular scenario, where the entries of the parameter vector θ directly correspond to the controls at different time points, this then produces a suitable extension of the approach advocated in the original GRAPE manuscript[3].

In the following formulas, we will assume for simplicity unitary evolution outside the measurements, but the extension to (Markovian) dissipative dynamics is comparatively straightforward (using a Liouvillian superoperator instead of the Hamiltonian).

We will first describe a general approach which works for any arbitrary choice of the parametrization θ. Further below, we will then specialize to a scenario where the original GRAPE idea for efficient gradient evaluation can be applied.

The general task is to obtain the gradient of the quantum state with respect to the parameters θ that enter the controls (and, likewise, the gradient of the final probability $P(m_1, m_2, \ldots)$ of a measurement sequence).

In modern machine learning language, tracking the evolution of parameter-gradients in the manner described in the following is connected to the recent developments of neural ordinary differential equations [44], where efficiency is obtained by not using automatic differentiation as a black box but rather evaluating analytically the form of the equations of motion for the gradients (and then solving those equations numerically with any efficient solver available). We can obtain the parameter gradient of the quantum state by solving the following evolution equation during measurement-free time intervals:

$$i\partial_t \partial_\theta \hat{\rho} = [\partial_\theta \hat{H}, \hat{\rho}] + [\hat{H}, \partial_\theta \hat{\rho}], \tag{8}$$

where $\hat{\rho}$ is the solution to the original equation of motion, $i\partial_t \hat{\rho} = [\hat{H}, \hat{\rho}]$, and the initial condition at time 0 would be $\partial_\theta \hat{\rho} = 0$ (we have set $\hbar \equiv 1$ for brevity). The interesting step now happens at a measurement, where $\hat{\rho}(t^+) = \hat{M}(m)\hat{\rho}(t^-)\hat{M}^\dagger(m)/P_m$, with the probability for the measurement outcome, $P_m = \mathrm{tr}[\hat{M}(m)\hat{\rho}\hat{M}^\dagger(m)]$. For brevity we suppress the index j (used in the main text) that would indicate the number of the measurement in the sequence. It now follows that we have

$$\partial_\theta \hat{\rho}(t^+) = \hat{M}(m)\partial_\theta \hat{\rho}(t^-)\hat{M}^\dagger(m)/P_m - \\ \hat{\rho}(t^+)\mathrm{tr}[\hat{M}(m)\partial_\theta \hat{\rho}(t^-)\hat{M}^\dagger(m)]/P_m. \tag{9}$$

Here the required $\partial_\theta \hat{\rho}(t^-)$ is the outcome of solving the previous continuous evolution equation up until time t. After this update, the continuous evolution of $\partial_\theta \hat{\rho}(t)$ will proceed. We note, however, that the controls (embedded inside $\hat{H}$ in the present setup) will now depend on the measurement outcome m that was selected. Likewise for later time intervals, they will depend on the whole previous sequence, as described in the main text.

At the end, we also need the gradient of the extra term, the log-likelihood of the whole measurement sequence, $\ln P(m_1, m_2, \ldots)$. One way to obtain this is to evolve an unnormalized version of the quantum state, $\tilde{\hat{\rho}}$, whose trace will give P, which follows the same evolution as the quantum state itself, but without the normalization factors that are the probabilities for the individual measurement outcomes. The θ-gradient of this unnormalized state again follows an evolution equation of the form like Eq. 8, just with $\tilde{\hat{\rho}}$ substituted for $\hat{\rho}$, during the unitary evolution intervals. However, at a measurement-induced update, we obtain the simpler rule $\tilde{\hat{\rho}}(t^+) = \hat{M}(m)\tilde{\hat{\rho}}(t^-)\hat{M}^\dagger(m)$ and consequently $\partial_\theta \tilde{\hat{\rho}}(t^+) = \hat{M}(m)\partial_\theta \tilde{\hat{\rho}}(t^-)\hat{M}^\dagger(m)$.

What we have described here so far uses less assumptions than GRAPE, because θ can enter the controls in an arbitrary manner. In GRAPE [3], an additional assumption was used to simplify the gradients further and gain efficiency: The components of the parameter vector θ were supposed to directly correspond to the control values applied at different time steps. That is, schematically speaking, we would have $\theta_1, \theta_2, \ldots$ associated with the controls at time steps $j = 1, 2, \ldots$. This then leads to a further simplification in the evaluation of the gradients. Importantly, if the number of parameters scales with the number of time steps N, then this approach has a runtime growing only linearly in N, while the general approach outlined above would need N^2 operations.

Let us briefly recall the GRAPE approach to gradient evaluation [3], before extending it. In the simplest possible version, with unitary evolution, let us consider the fidelity $\mathrm{tr}(\hat{\sigma}(T)\hat{U}(T,0)\hat{\rho}(0)\hat{U}(0,T))$. The derivative with respect to parameters θ entering the Hamiltonian will produce a contribution for each time $t \in (0, T)$ in the evolution. Specifically, the contribution from time t will be an expression of the type $\mathrm{tr}(\hat{\sigma}\hat{U}(T,t)[-i\frac{\partial \hat{H}}{\partial \theta}, \hat{\rho}(t)]\hat{U}(t,T))$. Using the cyclic property of the trace, this can be reordered to obtain $\mathrm{tr}(\hat{U}(t,T)\hat{\sigma}(T)\hat{U}(T,t)[-i\frac{\partial \hat{H}}{\partial \theta}, \hat{\rho}(t)])$. This can now be re-interpreted, namely as the overlap between a backward-evolved target state $\hat{\sigma}(t) = \hat{U}(t,T)\hat{\sigma}(T)\hat{U}(T,t)$ and the perturbation of the forward-evolved state at time t: $\mathrm{tr}(\hat{\sigma}(t)[-i\frac{\partial \hat{H}}{\partial \theta}, \hat{\rho}(t)])$.

In machine learning language, the GRAPE procedure of obtaining gradients in this way can essentially be viewed as an analytically derived version of backpropagation for this specific case of a quantum-physical evolution. It is very efficient, since the effort scales only linearly in the number of time steps, even if there is a different, independently optimizable parameter $\theta(t)$ for each time step.

The question is how this procedure needs to be modified in the presence of measurements. Let us imagine we have a particular trajectory with a given fixed sequence of measurement outcomes. We find that we can perform the temporal backpropagation (starting from the final time T) in the same manner as reviewed above, until a point in time $\tilde{t}$ where a measurement has happened (unless of course we talk about a time point t later than the last measurement). At that point $\tilde{t}$, we need to replace $\hat{\sigma}(\tilde{t}) = \hat{U}(\tilde{t},T)\hat{\sigma}\hat{U}(T,\tilde{t})$ by the following expression:

$$\hat{\sigma}'(\tilde{t}) = \frac{1}{P}\hat{M}^\dagger\hat{\sigma}(\tilde{t})\hat{M} - \frac{1}{P^2}\hat{M}^\dagger\hat{M}\mathrm{tr}(\hat{M}^\dagger\hat{\sigma}(\tilde{t})\hat{M}\hat{\rho}(\tilde{t})) \quad (10)$$

Here we have defined, for brevity, the measurement operator $\hat{M} \equiv \hat{M}_{\tilde{m}}$ at time point $\tilde{t}$, with measurement outcome $\tilde{m}$, and the associated probability $P \equiv P_{\tilde{m}} = \mathrm{tr}(\hat{M}_{\tilde{m}}\hat{\rho}(\tilde{t})\hat{M}_{\tilde{m}}^\dagger)$, where $\hat{\rho}(\tilde{t})$ is already conditioned on previous measurement outcomes, for times less than $\tilde{t}$ and has been obtained by the forward evolution starting from time 0 (with measurements and re-normalization of the state after each measurement).

After this procedure has been implemented for the measurement at $\tilde{t}$, we would proceed with the backward evolution of $\hat{\sigma}$ until point t, where the derivative is to be evaluated. There, we employ the same formula as in the usual GRAPE approach, i.e. we would evaluate $\mathrm{tr}(\hat{\sigma}(t)^\dagger[-i\frac{\partial\hat{H}}{\partial\theta},\hat{\rho}(t)])$.

If there are multiple measurements between t and T, the backward evolution will proceed by alternating unitary evolution and applying the formula in Eq. (10).

If we want to treat the unnormalized quantum state in the same manner, e.g. for obtaining the log-likelihood term, we will only need the trace of that unnormalized state $\tilde{\hat{\rho}}$ at the end of the time evolution (see our discussion above). Formally, this is as if we were to calculate the fidelity against a state $\hat{\sigma}(T) = 1$, which is given by the identity matrix. We can now evolve this state backwards in the manner discussed above, but in addition, Eq. (10) simplifies: One needs to drop the second term and also formally set $P = 1$ in the first term.

Finally, we briefly remark how the procedure will change if we are dealing with continuous measurement outcomes (strong continuous measurements, as briefly discussed in the main text, using the 'reparametrization trick'). In that case, we do not need the log-likelihood term. However, we now do need to differentiate the measurement outcome $m = f_{\hat{\rho}}^{-1}(z)$ which depends on some random variable z (of a fixed distribution, not dependent on θ) and the quantum state $\hat{\rho}$ (that does depend on θ). As a consequence, Eq. (10) needs to be modified. We have to add the following terms to the right-hand-side:

$$\frac{1}{P}\partial_\theta(\hat{M}^\dagger\hat{\sigma}(\tilde{t})\hat{M}) - \frac{1}{P^2}\mathrm{tr}(\hat{M}^\dagger\hat{\sigma}(\tilde{t})\hat{M}\hat{\rho}(\tilde{t}))\partial_\theta(\hat{M}^\dagger\hat{M}) \quad (11)$$

Here ∂_θ in both parts of this expression is supposed to act only on the $\hat{M}^\dagger$ and $\hat{M}$ terms. This derivative is to be applied in the way $\partial_\theta\hat{M}(m) = (\partial_m\hat{M}(m))(\partial_\theta m)$, where the derivative of m with respect to θ must be evaluated using the dependence of the inverse cumulative distribution function on the θ-dependent quantum state at that time-point.

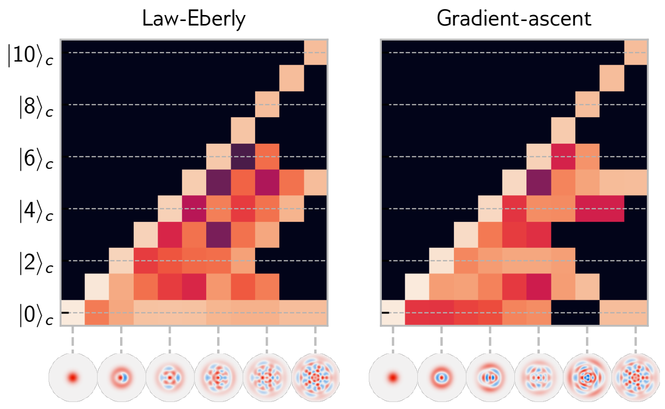

FIG. 5. Comparison between one solution obtained analytically from the Law-Eberly protocol and a strategy found by using gradient ascent. The target state is $|\psi\rangle = (|0\rangle + |5\rangle + |10\rangle)/\sqrt{3}$. Even though some details look different, we have verified that the gradient ascent strategy is a valid alternative solution for the Law-Eberly equations (which do not determine the controls uniquely).

Appendix B: Law-Eberly algorithm

As a benchmark with an analytical solution (but still without feedback), we consider the task of preparing an arbitrary pure cavity state in a cavity-qubit system. This can be achieved by exploiting the well-known Law-Eberly protocol [50]. This algorithm relies on the essential assumption that we start from the ground state. We briefly recall it in the following.
The Hamiltonian that describes the system is a Jaynes-Cummings model with controllable couplings:

$$\hat{H}(t) = (\alpha(t)\hat{\sigma}_+ + \alpha^*(t)\hat{\sigma}_-) + (\beta(t)\hat{a}\hat{\sigma}_+ + \beta^*(t)\hat{a}^\dagger\hat{\sigma}_-) \quad (12)$$

where the first term refers to the qubit excitation and the second one refers to the cavity-qubit interaction. The two complex controls $\alpha(t)$ and $\beta(t)$ can assume continuous values and, for simplicity, we will assume that when $\alpha(t) \neq 0$, then $\beta(t) = 0$, and when $\beta(t) \neq 0$, then $\alpha(t) = 0$. If we define a dynamics made of N steps, then each of those steps contains one qubit excitation step followed by one cavity-qubit coupling step.
The algorithm starts in the ground state

$$|\psi\rangle_{init} = |0, g\rangle \quad (13)$$

and ends up in an arbitrary superposition up to (including) Fock state N,

$$|\psi\rangle_{target} = \sum_{n=0}^{N} c_n |n, g\rangle \quad (14)$$

in N steps.
The time evolution operator that represents the whole dynamics is:

$$\hat{U} = \hat{B}_N \hat{A}_N \hat{B}_{N-1} \hat{A}_{N-1} ... \hat{B}_1 \hat{A}_1 \tag{15}$$

where $\hat{A}_j = \exp\{-i(\alpha_j \hat{\sigma}_+ + \alpha_j^* \hat{\sigma}_-)\}$ and $\hat{B}_j = \exp\{-i(\beta_j \hat{a}\hat{\sigma}_+ + \beta_j^* \hat{a}^\dagger \hat{\sigma}_-)\}$

The Law-Eberly idea is to start from the target state and progressively remove excitations from the cavity: at each step, the goal should be to take the contributions of the highest excited states, $|N, g\rangle$ and $|N-1, e\rangle$, and bring them down to $|N-1, g\rangle$. After the ground state is reached in this manner, one simply reverses the obtained control sequence. Due to the unitarity of the process, the desired sequence that turns $|\psi\rangle_{init}$ into $|\psi\rangle_{target}$ is achieved.
By imposing this condition on the reverse evolution $\hat{U}\,|\psi\rangle_{target}$, the problem switches to solving the following system of non-linear equations (assuming α and $\beta \in \mathbb{R}$):

$$\begin{aligned} \langle j, g|\psi_{j+1}\rangle \cos\!\Big(\beta_j \sqrt{j}\Big) + \\ +i\,\langle j-1, e|\psi_{j+1}\rangle \sin\!\Big(\beta_j \sqrt{j}\Big) = 0, \\ \langle j-1, e|\, \hat{B}_j^\dagger\, |\psi_{j+1}\rangle \cos(\alpha_j) + \\ +i\,\langle j-1, g|\, \hat{B}_j^\dagger\, |\psi_{j+1}\rangle \sin(\alpha_j) = 0 \end{aligned} \tag{16}$$

where $|\psi_j\rangle = \hat{A}_j^\dagger \hat{B}_j^\dagger \hat{A}_{j-1}^\dagger \hat{B}_{j-1}^\dagger ... |\psi_{target}\rangle$.

It should be noted that the solution of these equations is not unique. This is why Fig. 5 shows two different strategies for the same task, although both of them fulfill the Law-Eberly ansatz.

Appendix C: Model-free reinforcement learning for the Jaynes-Cummings scenario

It turns out that state-of-the-art model-free RL has surprising difficulties in addressing a physical scenario as important and conceptually simple as the Jaynes-Cummings model. In this subsection we provide some more details.

We will only consider the (simpler) no-feedback case, meaning only the two controls $\alpha(t)$ and $\beta(t)$ (see main text) are available. Since model-free RL already has severe problems in this case, we did not explore further the more challenging cases.

In our numerical experiments, we relied on the RL library Stable Baselines [59], which implements many of the most well-known optimized state-of-the-art RL algorithms. The RL environment (not to be confused with a "physical" environment) has been implemented in the following way:

- Action a_j: The two continuous controls, $\alpha(t_j)$ and $\beta(t_j)$.
- State s_j (i.e. input to the agent): In principle, the no-feedback task requires no state input. However, we chose to make it easier for the agent, by supplying the full current quantum state of the system at time t_j. Since the state is pure and the system is closed, we simplify the observation by only using the state vector $|\psi_j\rangle$ (instead of the density matrix). Since it is complex-valued, we split its real and imaginary part and so we have a vector of length $2N$, where N is the size of the Hilbert space.
- Reward r_j: the fidelity at step t_j (in various versions, see below).

We have used a variety of different approaches to solve the task of pure state preparation. These included: using either a sparse final reward (i.e. $r_j \neq 0$ only if $j = N$) or else a reward based on the fidelity at each time step, either discrete (discretized) actions or continuous actions, and several different optimization algorithms (PPO[55], A2C[60], HER[61], TRPO[62], DDPG[63]). The results shown in 3a) are the best results we could manage to produce among all these approaches. They were obtained with PPO, continuous actions and sparse rewards and using the following hyperparameters (see the Stable Baselines PPO documentation):

Parameter	Value
gamma	0.99
n_steps	0.01
ent_coef	0.999
learning_rate	0.00025
vf_coef	0.5
max_grad_norm	0.5
lam	0.95
nminibatches	4
noptepochs	4
cliprange	0.2

Appendix D: Controls via Neural network or Lookup Table, and remarks on Physical Simulations

As explained in the main text, in the feedback-GRAPE approach presented in this manuscript we can produce the control values (conditioned on previous measurement results) either with the help of a neural network or with the help of a lookup table (containing trainable control values). In this section we present more details on both of these approaches, as implemented for the specific numerical examples shown in the main text.

In our illustrative physical scenario (the feedback-controlled Jaynes-Cummings model), there are four control parameters: $\alpha_j, \beta_j, \gamma_j$ and δ_j. In the most general case, where arbitrary superpositions should be generated, α_j and β_j need to be complex. In the scenarios whose results are displayed in the main text, this was not needed

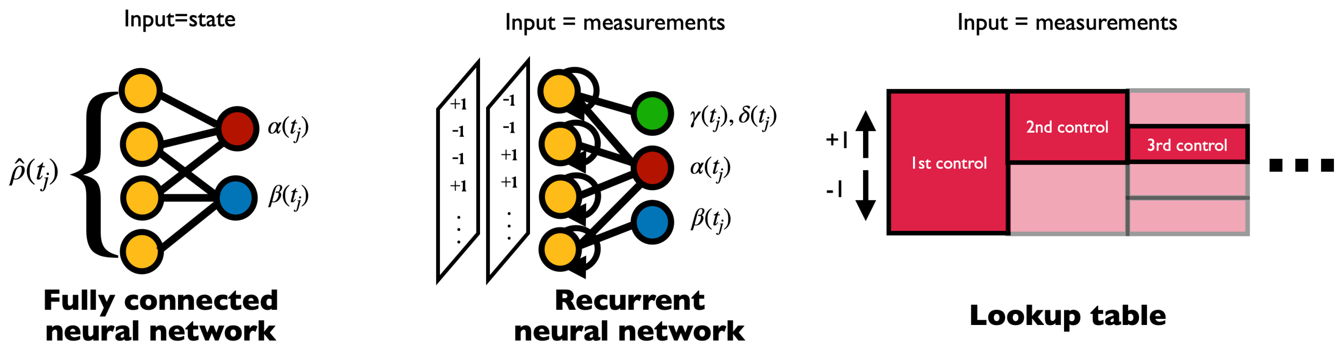

FIG. 6. Sketch of the three alternative types of trainable controls that can be employed in feedback-GRAPE: the first one is a fully connected neural network which receives the density matrix (quantum state) of the system as input and output the controls. The second one is a RNN with GRU cells as recurrent neurons. The third one is a lookup table, with $\sum_{n=0}^{N} 2^n$ entries (when feedback is required and when the measurement outcomes are binary, as shown here), and each entry contains the controls that need to be applied after observing a particular measurement sequence.

due to the nature of the target states. However, we have checked independently that the whole approach works just as well for complex control parameters.

Neural Network - We first discuss the case when the controls are computed by means of a neural network. This network can receive the measurement results so far, $m_1, m_2, \ldots, m_j$. Alternatively, we can also supply it with the quantum state as input, which has been updated according to the measurement outcomes. Both techniques supply the full information content needed to apply the next control.

For the "state as input" approach, we defined a fully connected neural network that takes the density matrix of the system as input. Since the density matrix is complex-valued, we chose to split it into its real and imaginary parts and to stack it, in such a way that for a $N_\mathrm{H} \times N_\mathrm{H}$ density matrix, the input tensor has shape $[N_\mathrm{H} \times N_\mathrm{H}, 2]$.

The fully connected NN has been employed both for the no-feedback case (pure state preparation), where in principle no such input would be needed (but can still be helpful for convergence), and also for the more interesting feedback cases.

If, on the other hand, we want to supply directly the measurement results, then we employ a recurrent neural network (RNN). For our scenario, its input at each time step is a binary measurement outcome $m_j \in \{-1, +1\}$. When a RNN network is used, due to the probabilistic outcome of the trajectories during a simulation, it is useful to feed batches of multiple randomly sampled trajectories as input to the network.

As already mentioned, both types of neural networks output real-valued controls $\alpha_j, \beta_j, \gamma_j$ and δ_j to be applied in the next time step. When complex-valued controls are required, two additional neurons can be added to the output of the neural networks, and they correspond to the imaginary parts of α_j and β_j. In the main text, we did not use complex controls, because these were not needed for the tasks considered there.

Our neural networks are implemented using Keras and their hyperparameters are shown for completeness in Tables I and II (we always used the same hyperparameters).

Parameter	Value
Neurons	$[[N_H \times N_H, 2]$, Flatten, 30, 30, 2 or 4]
Batch size	1
Activation	ReLU
Initializer	Glorot uniform
Initial bias last layer	π

TABLE I. Parameters of the fully connected neural network

Parameter	Value
Type RNN cells	GRU
Neurons	[30, 2 or 4]
Batch size	10
Dropout	0.2
Input shape	[batch_size, 1, 1]
Activation	tanh
Recurrent activation	Sigmoid
Initializer	Glorot uniform
Initial bias last layer	π

TABLE II. Recurrent neural network

Lookup Table - Another way to represent the entire feedback-based control strategy is to use a lookup table, which essentially is just a list of optimisable parameters. In the case of feedback, we have to build a lookup table that encodes the structure of a decision tree. For binary measurement outcomes (as used here), this has $\sum_{n=0}^{N} 2^n$ entries, each of which is the vector of all control parameters, i.e. in our scenario (α_j, β_j, γ_j, δ_j). Each column of this table represents the 2^j possible control parameter vectors at time step $j \in \{0, ...N\}$. At $j = 0$, we have only one set of numbers, which stand for the (only) possible control vector to apply (not dependent on any previous measurement; in our case reduced to only the entries controlling the first measurement). At step $j = 1$, we have two sets of numbers, and we apply the set of controls corresponding to the observed measurement, and so on and so forth. By doing so, we can apply controls conditioned on the "memory" of all previous measurements, at the cost of keeping an exponentially growing number of entries in the computer's memory. Many of those will likely not be explored at all, if their probabilities are too small.

In our numerical experiments, we went as far as lookup tables containing about $2^{21} \sim 2 \cdot 10^6$ entries, which still was easily handled. The initial condition for the whole table was to set each parameter value to a random number uniformly distributed within $(0, \pi)$.

In several results mentioned in the main text, we use a lookup table "without memory". This means that there is just one control parameter vector for each step j, instead of a tree-type structure with an exponentially growing number of parameters. Thus, we still optimize the controls but ignore the result of the measurements. This is used both for the "non-adaptive" scheme for the purification task in Fig. 3c) and in figure 3h).

A sketch of all of the three feedback-based strategies discussed here and in the main text (neural network with state as input, recurrent neural network with measurement sequence as input, and a tree-type lookup table) is shown in Fig. 6.

In any case, in whatever ways we choose to parametrize our controls, we have a finite number of parameters that need to be learned. In order to do so, the optimizer employed for every example is Adam [64], and its hyperparameters are shown in table III.

Parameter	Value
learning_rate	0.01*
beta_1	0.9
beta_2	0.999
epsilon	1E-7
clipnorm	1
clipvalue	0.5

*unless otherwise specified

TABLE III. Adam hyperparameters

Physical Simulations - In the unitary case, we simply apply the sequence of parametrized unitaries, as explained in the main text. In the case of decay (in the state stabilization scenario), we have solved the master equation for the density matrix during the respective time intervals (where decay is present). Specifically, we have simulated the weak Markovian coupling of the oscillator to a zero temperature bath via the Lindblad master equation,

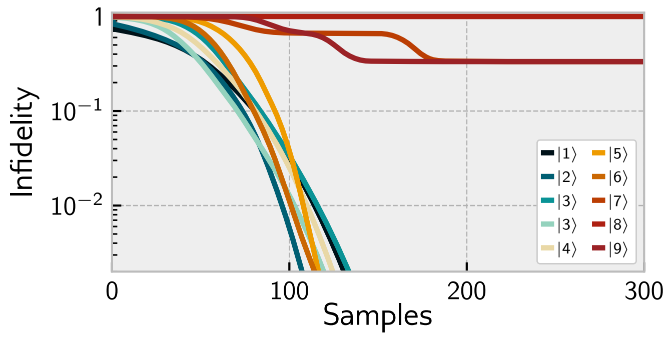

FIG. 7. Gradient-ascent-based pure state preparation of the first 10 Fock states, by starting from the ground state. Each curve is the best one among 10 tries. Instead of using a network like in Fig. 3c), here we directly optimize the control parameter vector. Note that some higher Fock states are not successfully prepared in this example. This can be mitigated by performing gradient ascent from more random initial conditions.

$$\dot{\hat{\rho}} = \gamma \left(\hat{a}\hat{\rho}\hat{a}^\dagger - \frac{1}{2}\left\{\hat{a}^\dagger\hat{a}, \hat{\rho}\right\}\right). \quad (17)$$

We discretize this continuous time-evolution applying the fourth-order Runge-Kutta method. In particular, every decay step in Fig. 4 (corresponding to the overall decay time $\Delta t = 0.05/\gamma$) corresponds to ten Runge-Kutta iterations.

We chose the Hilbert-space to have a finite dimension N_H with a cut-off in the Fock states excitation number. An appropriate choice of the cut-off depends both on the initial and the target state and ranges from 10 to 30 in our simulations.

Appendix E: Further numerical results

In this subsection we present a few more numerical results to illustrate various options or aspects of the technique.

1. Pure-state preparation with a lookup table

For pure-state preparation out of the ground state (a pure control task), in the main text we showed results for an approach where we feed the current quantum state as input to a neural network to obtain the control parameter vector. For comparison, in Fig. 7 we show how the "lookup table" approach fares in this case. Instead of the parameters of a NN, we optimize directly the control parameter vector Since there is no feedback, the number of entries in this "lookup table" grows only linearly with the total number of measurements. Surprisingly, in this case the lookup table method performs worse than the

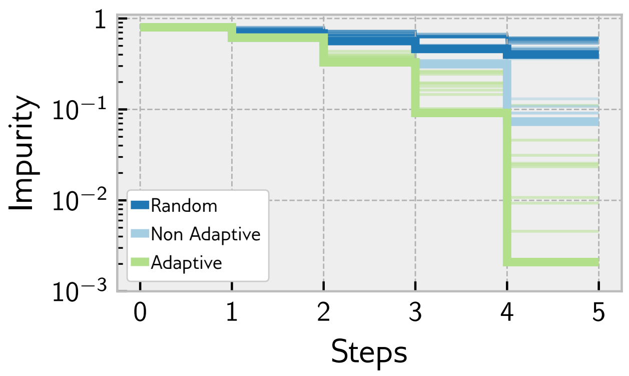

FIG. 8. Purification of a thermal state (with $\bar{n} = 2$) like in Fig. 3d) of the main text. Here, the shaded lines show 10 different strategies found by repeated runs of the algorithm, from different random starting points. The thick lines represent the best strategy found.

network approach, since it only can find a solution reliably up to Fock state $|6\rangle$. This could be mitigated to some extent by having more runs from different starting conditions.

2. Effect of different initial condition on the training

In order to assess the variability during the training, we show in this subsection how the results of Fig. 3 can change, depending on the choice of different random initial conditions of the algorithm. As a first example, we show in Fig. 8 an equivalent plot of Fig. 3d), but in this case we want to differentiate the distinct strategies found by feedback-GRAPE. The majority of the adaptive runs can systematically reach higher purities then the other strategies (Random and Non-Adaptive). Nonetheless, one should be aware of such variability of strategies at the end of the training.
To further analyse the variability of training, we focus on the state preparation case from a thermal state, like in Fig. 3f,g,h. In Fig. 9, the performance for many different target state was evaluated, along with the uncertainty due to different initial condition.

3. Impact of the batch size on generalization

A final analysis that we conducted deals with the effect of the batch size during training. We want to analyze both the performance during training and the generalization capabilities of the strategy learned. In order to asses that, we focus on the state preparation case from thermal state (Fig. 3f)-i)). We run different trainings with distinct batch sizes (ranging from 1 to 100). For each batch size, we run 5 different training. We then post-select the best one, by computing the average fidelity on a much larger batch size (i.e. 1000). In 10 we then show the best performing NN/RNN/Lookup table. Interestingly, even though the training is noisier with a lower batch size, it seems that feedback-GRAPE can converge faster and to higher fidelity solution. Also, it seems that lower batch sizes can generalize well to higher ones. The better performance of lower batch sizes could be due to the possibility of the optimizer to escape local minima more efficiently than larger ones.

Appendix F: Detailed analysis of strategies discovered by feedback-GRAPE for the Jaynes-Cummings model scenario

In our work, we chose several different tasks within a Jaynes-Cummings model to illustrate the performance of our approach. Despite being only an illustrative physical example in this context, the model is of sufficient interest as a paradigm for actual feedback control of quantum-optical systems. In this section, we describe some of the insights we were able to extract by closer inspection of the numerical results obtained by feedback-GRAPE, in situations with feedback.

In the main text, we show the decision tree for the purification of a thermal state with initial occupation number $\langle \hat{a}^\dagger \hat{a} \rangle = 2$ in four measurements. Here, we want to show how the insight gained by analyzing the decision tree for this special case allows to derive an analytical solution for an optimal purification strategy valid for arbitrary temperature and number of measurements.

We start by reviewing the physics for the building block measurement, cf Eq. (5). This type of measurement has been originally proposed in [53] and has been extensively used in quantum optics experiments with flying Rydberg atoms, e.g. to monitor the occupation number of a cavity in the presence of very small thermal fluctuations [54] or to prepare a Fock state starting from an initial coherent state [65]. After each measurement, the Fock state probability distribution $P_j(n)$ is updated by multiplying it with a sinusoidal mask,

$$P_{j+1}(n) \propto P_j(n) \cos^2\left[\gamma_i n + \frac{\delta_i}{2} + \pi(1 - m_j)/4\right]. \quad (18)$$

To better understand the effects of the measurement it is important to keep in mind two key insights: (i) If the measurement strength can be well approximated with a rational multiple of π, $\gamma_i = \pi p_i/q_i$ where p_i and q_i are co-prime numbers, the denominator q_i represents the period of the mask. Thus, the relative occupations $P(n)/P(n')$ of any pair of Fock states that have the same excitation number modulus q_i, $(n - n') \bmod q_i = 0$, do not change after the measurement. (ii) If the phase δ_i satisfies either condition

$$\pi\frac{p_i}{q_i}n_i + \frac{\delta_i}{2} = 0 \bmod \pi, \quad \text{or} \quad = \pi/2 \bmod \pi,$$

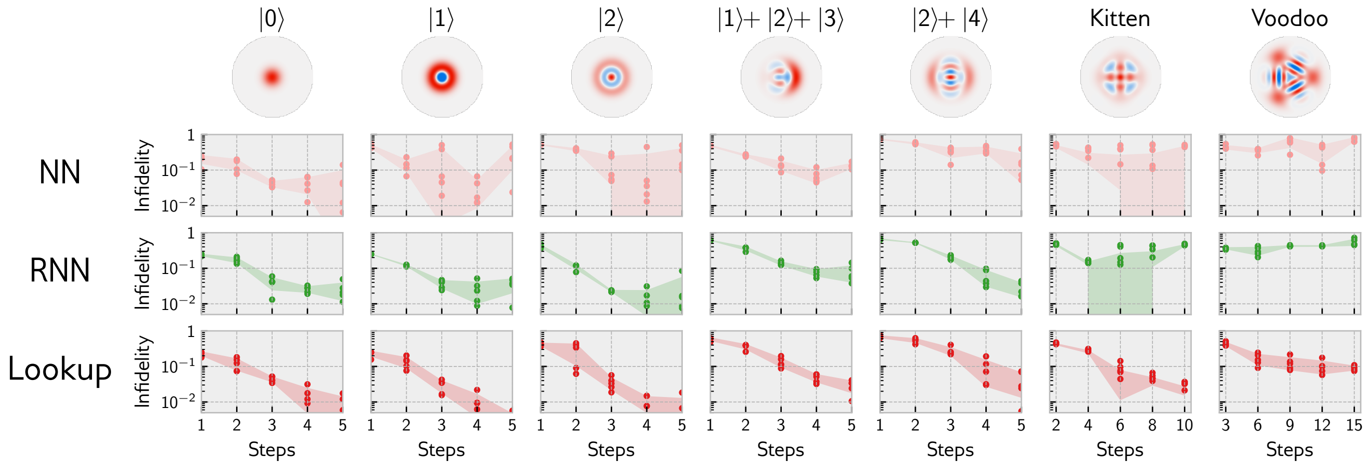

FIG. 9. State preparation from a thermal state ($\bar{n} = 1$), employing feedback, like in Fig. 3f)-i). We show the final infidelity as a function of the number of time steps available for the strategy. The columns represents various final target states, while the rows shows the three different approaches to obtain the control parameters (NN: neural network being fed the current quantum state as input; RNN: recurrent neural network obtaining the measurement sequence step-wise; Lookup: a lookup table as defined in the main text). Each dot represents a different training run with different initial condition. For each number of steps, 5 runs are shown. The shaded area represents the variance of the latter.

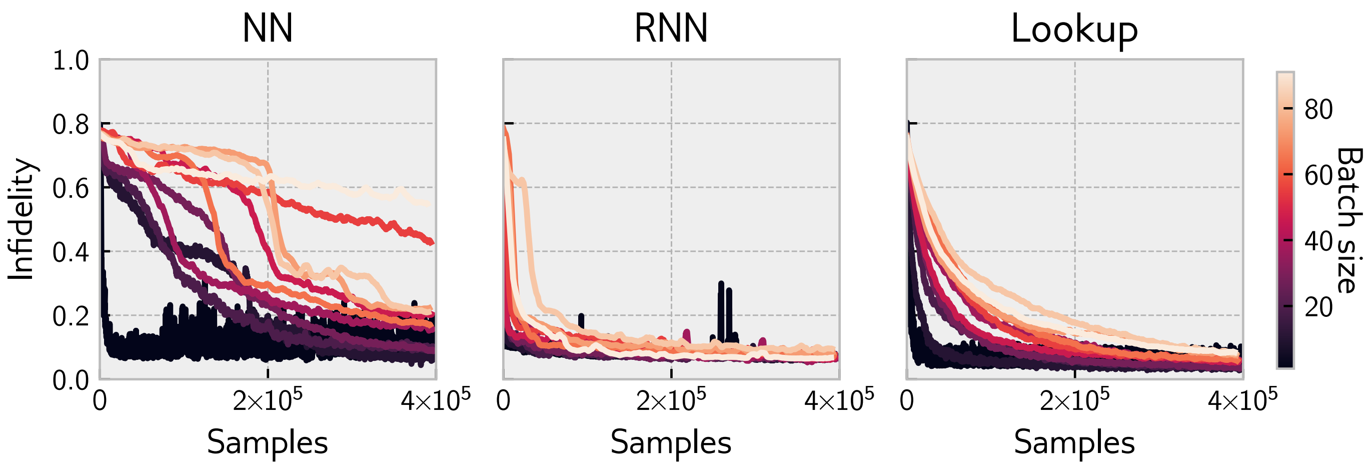

FIG. 10. State preparation from a thermal state (here $\bar{n} = 1$) and for target state $\psi = (|1\rangle + |2\rangle + |3\rangle)/\sqrt{3}$, in 6 steps. Each training was run with a different batch size (encoded in the color), and the performance of the resulting strategy (NN/RNN/Lookup) was evaluated with a larger batch size (here 1000) to suppress statistical noise. The training curves are plotted with a moving average of 50 samples in order to suppress fluctuations. The number of samples (i.e. batch size $\times$ number of gradient optimization steps) for each curve is the same.

for an integer n_i, one measurement outcome ($m_i = -1$ or $m_i = 1$, respectively) rules out the infinite set of Fock states with excitation numbers n satisfying $n \bmod q_i = n_i$. We note that if q_i is an even number any δ_i that satisfies the first condition for $n_i \equiv n_{i,-1}$ satisfies also the second condition for $n_i = n_{i,1} \equiv (n_{i,-1} + q_i/2) \bmod q_i$. In this scenario, each of the two possible measurement outcomes rules out a (different) infinite set of Fock states, $n_{i,\pm 1} \bmod q_i$ for $m_i = \pm 1$. We note further that there are infinitely many values of δ_i satisfying one of the two conditions in Eq. (18) for the same n_i. All of these values of δ_i are rational multiples of π.

Motivated by the insights (i) and (ii), we have written an algorithm that identifies values of γ_i and δ_i that are close to rational multiples of π with small denominators (we allow a deviation of 1% of π) and displays these rational values (in units of π) in the decision tree as shown in Fig. 3(d). By inspecting the decision tree, cf Fig. 3(d), one can immediately observe that the NN tends to use measurement strength γ_j corresponding to the period $q_j = 2^j$ for the j-th measurement. In order to understand this pattern, we inspect the phases δ_j selected by the NN. For the first measurement, the measurement strength is $\gamma_1 = \pi/2$ and the phase $\delta_1 = 0$. This corresponds to $n_{1,-1} = 0$ and $n_{1,1} = 1$. In other words, the Fock state 0 (1) along with all other even (odd) states are ruled out by the measurement $m_1 = -1$ ($m_1 = 1$). Thus, the net effect is that, irrespective of the measurement outcome, the probability of every second Fock state is set to zero. Such a measurement extracts exactly 1 bit of information in the large temperature limit. For the second measurement, the NN doubles the period of the sinusoidal mask, $q_2 = 4$, (independent of the outcome of the first measurement). By inspecting the phases δ_2 chosen adaptively by the NN we find out that they always allow to rule out either of the two most likely states after the measurement. For example, in the upper branch (corresponding to $m_1 = 1$) all odd states have been decimated and, thus, the two more likely states are the 0 and 2 Fock states. From the tree we see that $\delta_2 = \pi/2$ in this branch. This indeed satisfies the two conditions in Eq. (18) with $n_i = n_{2,-1} = 2$ and $n_i = n_{2,1} = 0$, respectively. In other words, the Fock states with $n \bmod 4 = 0$ ($n \bmod 4 = 2$) are ruled out by the measurement outcome $m_2 = 1$ ($m_2 = -1$). Since all odd Fock states had been already ruled out after the first measurement, the overall effect of the first two measurements is to postselect every fourth Fock state, $n \bmod 4 = 0$ ($n \bmod 4 = 2$) for $m_1 = 1$ and $m_2 = -1$ ($m_1 = m_2 = 1$). Likewise, the choice of the phase $\delta_2 = -\pi/4$ in the lower branch allows to postselect every fourth Fock state, now, $n$mod4 $= 1$ and $n$mod4 $= 3$ for $m_2 = 1$ and $m_2 = -1$, respectively. This strategy can be easily generalized for any arbitrarily large number of measurements J: the period q_i is doubled after every measurement, $q_j = 2^j$, independent of the measurement outcomes and appropriate adaptive phases δ_j are selected to always rule out either of the two most likely states. Such a strategy allows to postselect the Fock states with $n \bmod 2^J = n_i$ where n_i depends on the measurement history. More precisely there is a bijective mapping between $0 \leq n_i < 2^J - 1$ and the 2^J possible measurements outcomes. Indeed, a close inspection of the strength γ_i and phases δ_i selected by the NN shows that the NN adopts this strategy for all four measurements in most (but not all) branches. A notable exception is the third measurement in the lowest branch (corresponding to $m_1 = m_2 = -1$). This choice results in an ineffective measurement that does not allow to exclude either of the two most likeliest states. Interestingly, in this case the NN selects for the fourth measurement the measurement settings that were expected (according to the strategy identified above) already for the third measurement. We believe that this sub-optimal strategy corresponds to a local minimum for the gradient ascent. We note that the strategy whose tree is displayed in Fig. 3(d) has been obtained after selecting the best gradient ascent training run out of 10 runs with different random initializations. A tree without any such suboptimal measurements could be obtained by performing more gradient ascent runs or, more efficiently, by increasing the temperature of the initial mixed state (which will punish more suboptimal purification strategies).

The same optimal strategy discussed above can be implemented for infinitely many different choices of γ_j and δ_j. In particular, different bijective mappings between the measurement outcomes and the likeliest state n_j after j measurements can be implemented. To find a simple analytical solution for the phases γ_j for one of the implementations of the optimal strategy, we choose $p_j = 1$ and, thus, $\gamma_j = \pi/2^j$. In addition, we choose n_j as the number whose binary representation is $d_{j-1} \dots d_2 d_1$ with $d_i = (1 - m_i)/2$, e.g. for $m_1 = m_2 = -1$ corresponding to $d_1 = d_2 = 1$ we have $n_3 = 1 + 2 = 3$. This mapping is implemented, if the phase δ_j always allows to rule out the Fock state with largest probability (or, equivalently, lowest excitation number among the states that have not yet been decimated by previous measurements) for the measurement outcome $m_j = -1$. With these constraints we find a simple analytical solution for the phases, $\delta_j = \pi n_j / 2^j$.

[1] 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, The European Physical Journal D **69**, 279 (2015).

[2] C. P. Koch, Controlling open quantum systems: Tools, achievements, and limitations, Journal of Physics: Condensed Matter **28**, 213001 (2016).

[3] 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, Journal of Magnetic Resonance **172**, 296 (2005).

[4] 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, Physical Review A **84**, 022305 (2011).

[5] F. Dolde, V. Bergholm, Y. Wang, I. Jakobi, B. Naydenov, S. Pezzagna, J. Meijer, F. Jelezko, P. Neumann, T. Schulte-Herbrüggen, J. Biamonte, and J. Wrachtrup, High-fidelity spin entanglement using optimal control, Nature Communications **5**, 3371 (2014).

[6] C. H. Yang, K. W. Chan, R. Harper, W. Huang, T. Evans, J. C. C. Hwang, B. Hensen, A. Laucht, T. Tanttu, F. E. Hudson, S. T. Flammia, K. M. Itoh, A. Morello, S. D. Bartlett, and A. S. Dzurak, Silicon qubit fidelities approaching incoherent noise limits via pulse engineering, Nature Electronics **2**, 151 (2019).

[7] T. Schulte-Herbrüggen, A. Spörl, N. Khaneja, and S. J. Glaser, Optimal control-based efficient synthesis of building blocks of quantum algorithms: A perspective from network complexity towards time complexity, Physical Review A **72**, 042331 (2005).

[8] A. Spörl, T. Schulte-Herbrüggen, S. J. Glaser, V. Bergholm, M. J. Storcz, J. Ferber, and F. K. Wilhelm, Optimal control of coupled Josephson qubits, Physical

Review A **75**, 012302 (2007).

[9] R. W. Heeres, P. Reinhold, N. Ofek, L. Frunzio, L. Jiang, M. H. Devoret, and R. J. Schoelkopf, Implementing a universal gate set on a logical qubit encoded in an oscillator, Nature Communications **8**, 94 (2017).

[10] R. Fisher, F. Helmer, S. J. Glaser, F. Marquardt, and T. Schulte-Herbrüggen, Optimal control of circuit quantum electrodynamics in one and two dimensions, Physical Review B **81**, 085328 (2010).

[11] T. Schulte-Herbrüggen, A. Spörl, N. Khaneja, and S. J. Glaser, Optimal control for generating quantum gates in open dissipative systems, Journal of Physics B: Atomic, Molecular and Optical Physics **44**, 154013 (2011).

[12] S. Boutin, C. K. Andersen, J. Venkatraman, A. J. Ferris, and A. Blais, Resonator reset in circuit QED by optimal control for large open quantum systems, Physical Review A **96**, 042315 (2017).

[13] D. J. Egger and F. K. Wilhelm, Adaptive Hybrid Optimal Quantum Control for Imprecisely Characterized Systems, Physical Review Letters **112**, 240503 (2014).

[14] P. de Fouquieres, S. G. Schirmer, S. J. Glaser, and I. Kuprov, Second order gradient ascent pulse engineering, Journal of Magnetic Resonance **212**, 412 (2011).

[15] S. Machnes, E. Assémat, D. Tannor, and F. K. Wilhelm, Tunable, Flexible, and Efficient Optimization of Control Pulses for Practical Qubits, Physical Review Letters **120**, 150401 (2018).

[16] J. Zhang, Y.-x. Liu, R.-B. Wu, K. Jacobs, and F. Nori, Quantum feedback: Theory, experiments, and applications, Physics Reports Quantum Feedback: Theory, Experiments, and Applications, **679**, 1 (2017).

[17] R. van Handel, J. K. Stockton, and H. Mabuchi, Modelling and feedback control design for quantum state preparation, Journal of Optics B: Quantum and Semiclassical Optics **7**, S179 (2005).

[18] C. Sayrin, I. Dotsenko, X. Zhou, B. Peaudecerf, T. Rybarczyk, S. Gleyzes, P. Rouchon, M. Mirrahimi, H. Amini, M. Brune, J.-M. Raimond, and S. Haroche, Real-time quantum feedback prepares and stabilizes photon number states, Nature **477**, 73 (2011), arXiv:1107.4027.

[19] R. Vijay, C. Macklin, D. H. Slichter, S. J. Weber, K. W. Murch, R. Naik, A. N. Korotkov, and I. Siddiqi, Stabilizing Rabi oscillations in a superconducting qubit using quantum feedback, Nature **490**, 77 (2012).

[20] M. Hirose and P. Cappellaro, Coherent feedback control of a single qubit in diamond, Nature **532**, 77 (2016).

[21] C. K. Andersen, A. Remm, S. Lazar, S. Krinner, J. Heinsoo, J.-C. Besse, M. Gabureac, A. Wallraff, and C. Eichler, Entanglement stabilization using ancilla-based parity detection and real-time feedback in superconducting circuits, npj Quantum Information **5**, 1 (2019).

[22] A. Hentschel and B. C. Sanders, Efficient Algorithm for Optimizing Adaptive Quantum Metrology Processes, Physical Review Letters **107**, 233601 (2011).

[23] C. Ahn, A. C. Doherty, and A. J. Landahl, Continuous quantum error correction via quantum feedback control, Physical Review A **65**, 042301 (2002).

[24] J. Cramer, N. Kalb, M. A. Rol, B. Hensen, M. S. Blok, M. Markham, D. J. Twitchen, R. Hanson, and T. H. Taminiau, Repeated quantum error correction on a continuously encoded qubit by real-time feedback, Nature Communications **7**, 11526 (2016).

[25] C. Ryan-Anderson, J. G. Bohnet, K. Lee, D. Gresh, A. Hankin, J. P. Gaebler, D. Francois, A. Chernoguzov, D. Lucchetti, N. C. Brown, T. M. Gatterman, S. K. Halit, K. Gilmore, J. A. Gerber, B. Neyenhuis, D. Hayes, and R. P. Stutz, Realization of Real-Time Fault-Tolerant Quantum Error Correction, Physical Review X **11**, 041058 (2021).

[26] S. Krinner, N. Lacroix, A. Remm, A. Di Paolo, E. Genois, C. Leroux, C. Hellings, S. Lazar, F. Swiadek, J. Herrmann, G. J. Norris, C. K. Andersen, M. Müller, A. Blais, C. Eichler, and A. Wallraff, Realizing Repeated Quantum Error Correction in a Distance-Three Surface Code, arXiv:2112.03708 [cond-mat, physics:quant-ph] (2021), arXiv:2112.03708 [cond-mat, physics:quant-ph].

[27] R. S. Sutton and A. G. Barto, *Reinforcement Learning, Second Edition: An Introduction* (2018).

[28] M. Bukov, A. G. R. Day, D. Sels, P. Weinberg, A. Polkovnikov, and P. Mehta, Reinforcement Learning in Different Phases of Quantum Control, Physical Review X **8**, 031086 (2018), arXiv:1705.00565.

[29] M. August and J. M. Hernández-Lobato, Taking Gradients Through Experiments: LSTMs and Memory Proximal Policy Optimization for Black-Box Quantum Control, in *High Performance Computing*, Lecture Notes in Computer Science, edited by R. Yokota, M. Weiland, J. Shalf, and S. Alam (Cham, 2018) pp. 591–613.

[30] M. Y. Niu, S. Boixo, V. N. Smelyanskiy, and H. Neven, Universal quantum control through deep reinforcement learning, npj Quantum Information **5**, 1 (2019).

[31] R. Porotti, D. Tamascelli, M. Restelli, and E. Prati, Coherent transport of quantum states by deep reinforcement learning, Communications On Physics **2**, 1 (2019).

[32] Y. Baum, M. Amico, S. Howell, M. Hush, M. Liuzzi, P. Mundada, T. Merkh, A. R. Carvalho, and M. J. Biercuk, Experimental Deep Reinforcement Learning for Error-Robust Gate-Set Design on a Superconducting Quantum Computer, PRX Quantum **2**, 040324 (2021).

[33] T. Fösel, P. Tighineanu, T. Weiss, and F. Marquardt, Reinforcement Learning with Neural Networks for Quantum Feedback, Physical Review X **8**, 031084 (2018).

[34] S. Borah, B. Sarma, M. Kewming, G. J. Milburn, and J. Twamley, Measurement-Based Feedback Quantum Control with Deep Reinforcement Learning for a Double-Well Nonlinear Potential, Physical Review Letters **127**, 190403 (2021).

[35] V. V. Sivak, A. Eickbusch, H. Liu, B. Royer, I. Tsioutsios, and M. H. Devoret, Model-Free Quantum Control with Reinforcement Learning, arXiv:2104.14539 [quant-ph] (2021), arXiv:2104.14539 [quant-ph].

[36] R. Porotti, A. Essig, B. Huard, and F. Marquardt, Deep Reinforcement Learning for Quantum State Preparation with Weak Nonlinear Measurements, arXiv:2107.08816 [quant-ph] (2021), arXiv:2107.08816 [quant-ph].

[37] F. Schäfer, M. Kloc, C. Bruder, and N. Lörch, A differentiable programming method for quantum control, Machine Learning: Science and Technology **1**, 035009 (2020).

[38] N. Leung, M. Abdelhafez, J. Koch, and D. Schuster, Speedup for quantum optimal control from automatic differentiation based on graphics processing units, Physical Review A **95**, 042318 (2017).

[39] M. Abdelhafez, B. Baker, A. Gyenis, P. Mundada, A. A. Houck, D. Schuster, and J. Koch, Universal gates for protected superconducting qubits using optimal control, Physical Review A **101**, 022321 (2020).

[40] F. Schäfer, P. Sekatski, M. Koppenhöfer, C. Bruder, and M. Kloc, Control of stochastic quantum dynamics by differentiable programming, Machine Learning: Science and Technology **2**, 035004 (2021).

[41] H.-J. Liao, J.-G. Liu, L. Wang, and T. Xiang, Differentiable Programming Tensor Networks, Physical Review X **9**, 031041 (2019).

[42] L. Coopmans, D. Luo, G. Kells, B. K. Clark, and J. Carrasquilla, Protocol Discovery for the Quantum Control of Majoranas by Differentiable Programming and Natural Evolution Strategies, PRX Quantum **2**, 020332 (2021).

[43] M. Abdelhafez, D. I. Schuster, and J. Koch, Gradient-based optimal control of open quantum systems using quantum trajectories and automatic differentiation, Physical Review A **99**, 052327 (2019).

[44] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, Neural Ordinary Differential Equations, in *32nd Conference on Neural Information Processing Systems* (Montreal, 2018) p. 13.

[45] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems (2015).

[46] S. Hochreiter and J. Schmidhuber, Long Short-Term Memory, Neural Computation **9**, 1735 (1997).

[47] E. Jaynes and F. Cummings, Comparison of quantum and semiclassical radiation theories with application to the beam maser, Proceedings of the IEEE **51**, 89 (1963).

[48] B. W. Shore and P. L. Knight, The Jaynes-Cummings Model, Journal of Modern Optics **40**, 1195 (1993).

[49] A. Blais, A. L. Grimsmo, S. M. Girvin, and A. Wallraff, Circuit quantum electrodynamics, Reviews of Modern Physics **93**, 025005 (2021).

[50] C. K. Law and J. H. Eberly, Arbitrary Control of a Quantum Electromagnetic Field, Physical Review Letters **76**, 1055 (1996).

[51] M. Hofheinz, H. Wang, M. Ansmann, R. C. Bialczak, E. Lucero, M. Neeley, A. D. O'Connell, D. Sank, J. Wenner, J. M. Martinis, and A. N. Cleland, Synthesizing arbitrary quantum states in a superconducting resonator, Nature **459**, 546 (2009).

[52] R. W. Heeres, B. Vlastakis, E. Holland, S. Krastanov, V. V. Albert, L. Frunzio, L. Jiang, and R. J. Schoelkopf, Cavity State Manipulation Using Photon-Number Selective Phase Gates, Physical Review Letters **115**, 137002 (2015).

[53] M. Brune, S. Haroche, V. Lefevre, J. M. Raimond, and N. Zagury, Quantum nondemolition measurement of small photon numbers by Rydberg-atom phase-sensitive detection, Physical Review Letters **65**, 976 (1990).

[54] S. Gleyzes, S. Kuhr, C. Guerlin, J. Bernu, S. Deléglise, U. Busk Hoff, M. Brune, J.-M. Raimond, and S. Haroche, Quantum jumps of light recording the birth and death of a photon in a cavity, Nature **446**, 297 (2007).

[55] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal policy optimization algorithms (2017), arXiv:1707.06347 [cs.LG].

[56] A. Mousavian, A. Toshev, M. Fiser, J. Kosecka, A. Wahid, and J. Davidson, Visual Representations for Semantic Target Driven Navigation, arXiv:1805.06066 [cs] (2019), arXiv:1805.06066 [cs].

[57] K. Kim, M. W. Lee, Y. Kim, J.-H. Ryu, M. Lee, and B.-T. Zhang, Goal-Aware Cross-Entropy for Multi-Target Reinforcement Learning, arXiv:2110.12985 [cs] (2021), arXiv:2110.12985 [cs].

[58] C. N. Self, K. E. Khosla, A. W. R. Smith, F. Sauvage, P. D. Haynes, J. Knolle, F. Mintert, and M. S. Kim, Variational quantum algorithm with information sharing, npj Quantum Information **7**, 1 (2021).

[59] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, Stable baselines (2018).

[60] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning (2016), arXiv:1602.01783 [cs.LG].

[61] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, Hindsight experience replay (2018), arXiv:1707.01495 [cs.LG].

[62] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, Trust region policy optimization (2017), arXiv:1502.05477 [cs.LG].

[63] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, Continuous control with deep reinforcement learning (2019), arXiv:1509.02971 [cs.LG].

[64] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization (2017), arXiv:1412.6980 [cs.LG].

[65] C. Guerlin, J. Bernu, S. Deléglise, C. Sayrin, S. Gleyzes, S. Kuhr, M. Brune, J.-M. Raimond, and S. Haroche, Progressive field-state collapse and quantum non-demolition photon counting, Nature **448**, 889 (2007).