

Neural-Network Heuristics for Adaptive Bayesian Quantum Estimation

Lukas J. Fiderer^{1,2,*}, Jonas Schuff^{1,3} and Daniel Braun¹

¹*Eberhard-Karls-Universität Tübingen, Institut für Theoretische Physik, 72076 Tübingen, Germany*

²*School of Mathematical Sciences, The University of Nottingham, University Park, Nottingham, NG7 2RD, United Kingdom*

³*Department of Materials, University of Oxford, Parks Road, Oxford, OX1 3PH, United Kingdom*



(Received 27 May 2020; accepted 17 March 2021; published 7 April 2021)

Quantum metrology promises unprecedented measurement precision but suffers in practice from the limited availability of resources such as the number of probes, their coherence time, or nonclassical quantum states. The adaptive Bayesian approach to parameter estimation allows an efficient use of resources because of adaptive experiment design. For its practical success, fast numerical solutions for the Bayesian update and the adaptive experiment design are crucial. Here we show that neural networks can be trained to become fast and strong experiment-design heuristics using a combination of an evolutionary strategy and reinforcement learning. Neural-network heuristics are shown to outperform established heuristics for the technologically important example of frequency estimation of a qubit that suffers from dephasing. Our method of creating neural-network heuristics is very general and complements the well-studied sequential Monte Carlo method for Bayesian updates to form a complete framework for adaptive Bayesian quantum estimation.

DOI: [10.1103/PRXQuantum.2.020303](https://doi.org/10.1103/PRXQuantum.2.020303)

I. INTRODUCTION

In quantum metrology we aim to design quantum experiments such that one parameter or multiple parameters can be estimated from the measurement outcomes. Experiment design can involve the preparation of initial states, controlling the dynamics, or choosing measurements for readout. The estimation of parameters is a problem of statistical inference, and the most common approaches to tackle it are the frequentist one and the Bayesian one.

In the frequentist approach, experiments are typically repeated several times, which allows one to estimate the parameters from the statistics of measurement outcomes using, for example, maximum likelihood estimation. The problem of experiment design is often addressed with the Cramér–Rao bound formalism [1,2] by maximizing the quantum Fisher information with respect to experiment designs [3].

The Bayesian approach, on the other hand, relies on updating the current knowledge of the parameters after each experiment by means of Bayes’ law. Examples of

Bayesian quantum estimation involve state and process tomography [4–9], and phase and frequency estimation [10–13] with various experimental realizations [14–21]. The Bayesian approach is particularly suitable for adaptive experiment design: experiments can be optimized depending on the current knowledge of the parameters and the available resources. While adaptivity can increase the precision and save time and other resources compared with nonadaptive (frequentist) approaches [22,23], it involves a computational challenge: the Bayesian update and the consecutive optimization of the experiment design are both analytically intractable (with rare exceptions under idealized conditions [10,24]). In view of the short timescale of quantum experiments, slow numerical computation of the Bayesian update and the experiment design can drastically increase the total time consumed. To approximate the Bayesian update efficiently, a framework based on a sequential Monte Carlo (SMC) algorithm has been developed [6,7,25–27]. This framework, however, does not solve the problem of adaptive experiment design, which represents a second computational step.

In practice, one has to rely on so-called experiment-design heuristics; that is, fast-to-evaluate functions that take available information as input and return an experiment design as output; see Fig. 1(a). If the output depends on the available information, we speak of an adaptive heuristic. So far, adaptive-experiment-design heuristics for Bayesian estimation have been found mostly manually,

*lukasjfiderer@gmail.com

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article’s title, journal citation, and DOI.

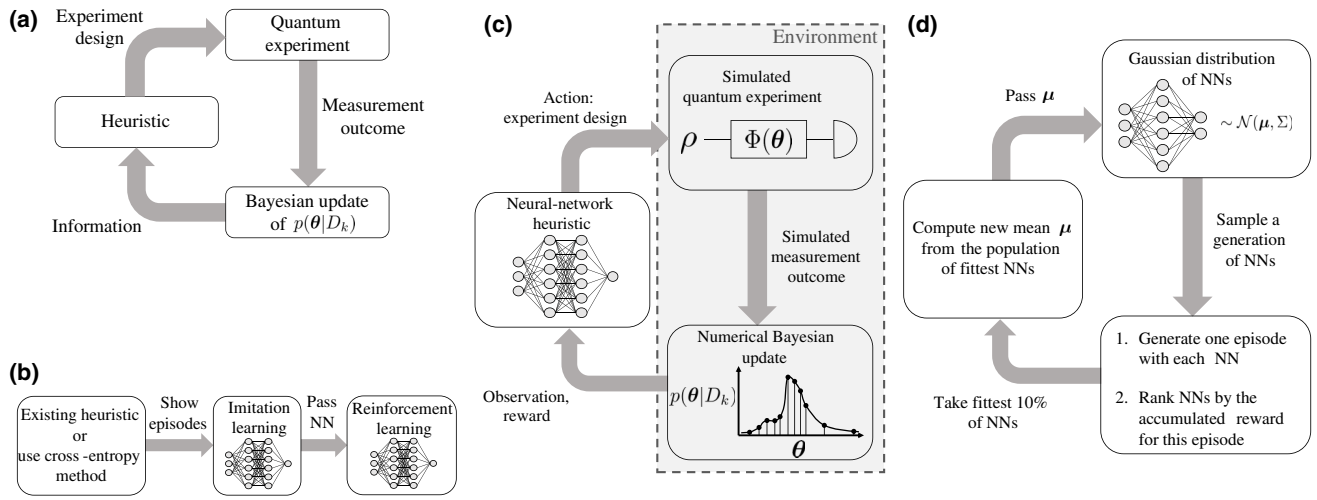


FIG. 1. (a) The setup for adaptive Bayesian quantum estimation. First, prior information [e.g., about the prior $p(\theta)$ and the available resources] is passed to the heuristic, which returns an experiment design e_1 . From the quantum experiment, we obtain a measurement outcome d_1 , which is used to numerically compute the Bayesian update of $p(\theta)$. Then, (updated) information is again passed to the heuristic and this cycle is repeated in the same manner until some exit condition is fulfilled. (b) The training procedure for the NNs. First, the NN learns to imitate the behavior of a known heuristic (which may have been found by the cross-entropy method). Then we use this NN as a starting point for RL. (c) The RL setup for generating training data. An agent (depicted as a NN) interacts with a RL environment. The RL environment defines the estimation problem. It consists of a simulated quantum experiment [depicted with initial state ρ and parameter-dependent quantum channel $\Phi(\theta)$] and a numerical Bayesian update. It also stores information about available resources for experiment design. (d) The evolutionary principle of a simple cross-entropy method. NNs are treated as vectors of parameters sampled from a Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ with constant covariance. At the beginning, the NNs are initialized randomly.

typically motivated by analytic arguments derived for idealized conditions concerning the experimental model and the available resources [9–11,20,28]. In one case, a manually found heuristic [28] was fine-tuned offline using a particle swarm algorithm [29]. Another approach is to optimize experiment designs between the experiments with respect to a restricted set of experiment designs so as to keep the problem numerically tractable [30]. Apart from Bayesian inference, particle swarm and differential evolution algorithms have been used to search a certain class of experiment-design heuristics (represented by binary decision trees) so as to optimize the scaling of the uncertainty in phase estimation with the number of entangled photons in an interferometer [20,31–33]. A general numerical framework for finding adaptive-experiment-design heuristics for Bayesian quantum estimation is missing so far.

We consider an approach to experiment-design heuristics that uses reinforcement learning (RL). Recently, RL has been used with great success to create programs that play chess and Go better than any other program or human [34]. RL has also been used in quantum physics [35–38] and, in particular, in quantum metrology for improving the dynamics of quantum sensors [39,40]. Problems in quantum metrology have also been tackled with other machine-learning methods using neural networks (NNs):

supervised learning has been used for calibrating quantum sensors [41], unsupervised learning has been proposed for retrieving Hamiltonian parameters from experimental data [42], and a Bayes classifier has been used for the identification of light sources [43].

Here we propose using neural NNs as experiment-design heuristics. We provide a general method based on a combination of an evolutionary strategy with RL for the creation of such NN experiment-design heuristics. This method builds upon and complements the SMC framework for approximate Bayesian updates [6,7,25–27]. It is general in the sense that (i) it can be easily adjusted to all kinds of estimation problems, (ii) it uses a very general ansatz for the experiment-design heuristic in the form of a NN, and (iii) it creates a NN experiment-design heuristic that can take into account not only knowledge of the parameters to be estimated but also additional information, for instance, concerning available resources. Furthermore, our method numerically approaches global optimization in the sense that it tries to find heuristics that are optimal for all future experiments as opposed to local (greedy) strategies that always optimize only the next experiment. The trained neural networks represent ready-to-use experiment-design heuristics, and we envisage their application in Bayesian quantum sensors in combination with the SMC framework for approximate Bayesian updates [6,7,25–27].

II. BAYES RISK

Let $\theta = (\theta_1, \dots, \theta_d) \in \Theta$ be a vector of parameters we want to estimate, and we assume that Θ restricts each θ_j to a finite interval. The prior $p(\theta)$ is a probability distribution on Θ that represents our knowledge before the first measurement, and we imagine that before each Bayesian estimation θ is sampled from $p(\theta)$. After each experiment, our knowledge of θ is updated with Bayes' law (see Appendix A). Let $p(\theta|D_k, E_k)$ represent this updated knowledge regarding θ after the k th measurement, where $D_k = (d_1, \dots, d_k)$ denotes the measurement outcomes from a sequence of k experiments $E_k = (e_1, \dots, e_k)$. In the following, we omit the dependence on experiment designs for clarity; for example, we write $p(\theta|D_k)$ instead of $p(\theta|D_k, E_k)$.

An experiment-design heuristic h is a function that maps available information—for instance, about θ or about available resources—to an experiment design for the next experiment; see Fig. 1(a). The idea is to consult the experiment-design heuristic before each experiment and design the experiment accordingly. Imagine that the available resources for one Bayesian estimation are such that we can make k experiments. Then we aim to choose an experiment-design heuristic h that minimizes the expected traced covariance over $p(\theta|D_k)$:

$$r[h|p(\theta)] = E_{D_k}(\text{tr}[\text{cov}_{\theta|D_k}(\theta)]), \quad (1)$$

where $\text{cov}_{\theta|D_k} = E_{\theta|D_k}(\theta\theta^T) - E_{\theta|D_k}(\theta)E_{\theta|D_k}(\theta)^T$ is the covariance, and the notation $E_{a|b}(c)$ denotes the expected value of c with respect to $a \in A$ distributed as $p(a|b)$, $E_{a|b}(c) = \int_A da p(a|b)c$. If a takes discrete values, the integral becomes a sum, and if a is distributed as $p(a)$, we write $E_a(c)$ instead of $E_{a|b}(c)$.

In Appendix B we show that $r[h|p(\theta)]$ corresponds to the Bayes risk for a loss function $L[\hat{\theta}_k(D_k)|\theta] = \|\hat{\theta}_k(D_k) - \theta\|_2^2$ with the Bayes estimator $\hat{\theta}_k$ after k measurements given by $\hat{\theta}_k(D_k) = E_{\theta|D_k}(\theta)$. We also discuss in Appendix B how to generalize Eq. (1) if other resources are available, and how the expected values used for the computation of Eq. (1) are approximated numerically.

For a given estimation problem and prior $p(\theta)$, the Bayes risk $r[h|p(\theta)]$ represents our figure of merit (smaller values are better) for experiment-design heuristics.

III. EXPERIMENT DESIGN AS A RL PROBLEM

Our idea is to train a NN to become an experiment-design heuristic. To this end, we simulate the experiment and the Bayesian update offline many times to generate training data [see Fig. 1(c)] and train the NN with them. Instead of simulating the experiment, one could use measurement data from an actual experiment to train the

NN; however, the calculation of the Bayesian update still depends on the model for the experiment. Either way, once the NN is trained, it represents an adaptive heuristic that can be used as part of a real Bayesian quantum sensor.

Instead of imitating the behavior of existing heuristics with a neural network, we are interested in creating heuristics that surpass conventional heuristics. To this end, we phrase the problem of experiment-design heuristics in the language of RL, which provides us with a suitable framework to optimize heuristics.

RL is an iterative method in which a RL agent, in our case the neural network, learns from training data. These training data are generated in each iteration when the RL agent interacts with the RL environment, as depicted in Fig. 1(c): The agent chooses an action (i.e., an experiment design) and passes it to the RL environment. In the RL environment, an experiment is simulated according to the action taken by the agent, and the measurement outcome is used for the Bayesian update. In return, the agent obtains an observation and a reward. By repeating this cycle several times, one generates training data consisting of actions, observations, and rewards. The learning phase is prescribed by a RL algorithm of choice and generally consists of a combination of learning from experience (using the training data) and random exploration (for more details, see Sec. IV).

The RL framework is based on the agent-environment model and can be understood in the context of Bayesian quantum estimation; see Fig. 1(c): One episode of training data is generated from one simulated Bayesian estimation that consists of a sequence of k simulated experiments including Bayesian updates and experiment designs chosen by the RL agent. The number of experiments k depends on the available resources. In the simplest case, the number of experiments is the limiting resource (referred to as “experiment limited” in the following); that is, each episode consists of the same number of experiments, and the Bayes risk is given by Eq. (1). More generally, the available resources set more complicated constraints such that k can vary between different episodes (see Appendix B for a generalization of the Bayes risk for such cases). If a resource is exhausted, the episode ends, the RL environment [see Fig. 1(c)] is reset to default values [the current posterior is reset to the prior $p(\theta)$ and a new true parameter θ is sampled from the prior], and another episode starts. Training data for one iteration typically consist of many episodes (e.g., approximately 10^3 in our model study below).

Crucial for the success of RL are the observations and rewards for the agent; see Fig. 1(c). The observations may contain information about the current knowledge $p(\theta|D_k)$, about the past, such as prior actions, and about resources, such as the remaining time. The reward should reflect the goodness of the behavior (actions) of the agent (a larger reward is better) and is used by the RL algorithm to enforce

behavior that leads to larger rewards; the RL agent learns from its experience. The negative Bayes risk seems to be an obvious choice for a reward function. However, the computation of the Bayes risk is too time-consuming. Instead, we define the reward after the k th experiment as the difference in the traced covariance over the posterior after the k th experiment and the $(k - 1)$ th experiment:

$$R(D_k) = \text{tr}[\text{cov}_{\theta|D_{k-1}}(\boldsymbol{\theta})] - \text{tr}[\text{cov}_{\theta|D_k}(\boldsymbol{\theta})]. \quad (2)$$

The idea behind this reward function is that (i) Eq. (2) is straightforward to compute in the Bayesian SMC framework, (ii) it reflects the difference in our uncertainty about $\boldsymbol{\theta}$ before and after the current step, and (iii) the expected value of the rewards accumulated from the beginning of an episode with respect to possible measurement outcomes D_k yields the negative Bayes risk (up to a constant); see also Appendix B. For example, in the experiment-limited case with N experiments per episode we have

$$E_{D_N} \left[\sum_{j=1}^N R(D_j) \right] = \text{const} - r[h|p(\boldsymbol{\theta})], \quad (3)$$

where the constant is given by the uncertainty in the prior, $\text{const} = \text{tr}[\text{cov}_{\theta}(\boldsymbol{\theta})]$. RL has the goal to maximize the expected discounted reward, which equals the left-hand side of Eq. (3) because we set the discount factor to 1. From Eq. (3), we thus see that RL indeed attempts to minimize the Bayes risk $r[h|p(\boldsymbol{\theta})]$.

IV. ALGORITHMS

We have seen that the problem of finding experiment-design heuristics can be formulated in the language of RL. There is a wide choice of algorithms that can be used to train the agent in a RL setting, ranging from easy-to-implement benchmarks to complicated state-of-the-art reinforcement learning algorithms. We choose two algorithms from both ends of this spectrum. The cross-entropy method (CEM) for continuous action spaces is a simple evolutionary strategy [44] that is a good benchmark for more complicated reinforcement learning algorithms [45–47]. The algorithm is explained in Fig. 1(c); for details, see Appendix D. Despite its simplicity, we will see that it is capable of yielding very good results.

On the other hand, we use trust region policy optimization (TRPO) [46] as implemented in the PYTHON package STABLE BASELINES [48]. We try several RL algorithms from STABLE BASELINES and choose TRPO because it shows the greatest stability and best performance. However, this is by no means a systematic comparison, which would require extensive simulations and hyperparameter optimization for each algorithm. Instead, we tune the hyperparameters of TRPO on the basis of general reasoning [49]. TRPO is an approximation of an iterative

procedure for optimizing policies with guaranteed monotonic improvement and is to date one of the most successful reinforcement learning algorithms [46].

The training of the NNs consists of two steps; see Fig. 1(b). We initialize the NN using imitation learning (pretraining) as implemented in Ref. [48]. The idea of imitation learning is to take advantage of an already existing (e.g., manually found) heuristic. The NN is trained to imitate the behavior from episodes created with the existing heuristic; see Fig. 1(c). This pretraining step is not strictly necessary but speeds up the training and makes RL stabler.

However, there might not always be a good heuristic available to imitate. For such cases and for the sake of comparison, we consider CEM. CEM is used to train a neural network from scratch starting from a randomly initialized neural network. Once the NN is pretrained, we use RL as a second step.

V. MODEL STUDY

We demonstrate our method of creating NN heuristics with an example of high practical relevance for magnetic field estimation with nitrogen-vacancy centers with applications in single-spin magnetic resonance [18,19,21,50]. Let us consider a qubit that evolves under the Hamiltonian $H(\omega) = (\omega/2)\sigma_z$, and we want to estimate the frequency ω . The qubit is prepared in $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$, evolves under $H(\omega)$ for a controllable time t , and is measured in the σ_x basis (assuming a strong projective measurement with outcomes labeled 0 and 1 corresponding to a measurement of $|+\rangle$ and $|-\rangle$). Let us further assume that the qubit suffers from an exponential decay of phase coherence, with characteristic time T_2 . According to the Born rule, the likelihood of finding an outcome $d \in \{0, 1\}$ with the σ_x measurement can be expressed as [7,10]

$$p(0|\omega, T_2, t) = e^{-t/T_2} \cos^2\left(\frac{\omega}{2}t\right) + \frac{1 - e^{-t/T_2}}{2} \quad (4)$$

for measuring $d = 0$ and as $p(1|\omega, t, T_2) = 1 - p(0|\omega, t, T_2)$ for measuring $d = 1$. Equation (4) defines all relevant properties of the experiment. A single experiment design consists of specifying the evolution time t . We consider the following estimation problems: (i) the estimation of ω without decoherence ($T_2 = \infty$; see the top row of panels in Fig. 2), (ii) the estimation of ω with known T_2 relaxation (we consider this problem twice with different values for T_2 ; see the second and third rows of panels in Fig. 2), and (iii) the simultaneous estimation of ω and T_2^{-1} ; that is, $\boldsymbol{\theta} = (\omega, T_2^{-1})$ (see the bottom row of panels in Fig. 2). In all cases we consider $\omega \in (0, 1)$ (making the problem dimensionless). We chose these problems to cover a wide range of estimation problems; the parameters are not cherry-picked, and similar results can be expected for different parameter choices (we discuss the limitations of our method in Sec. VI).

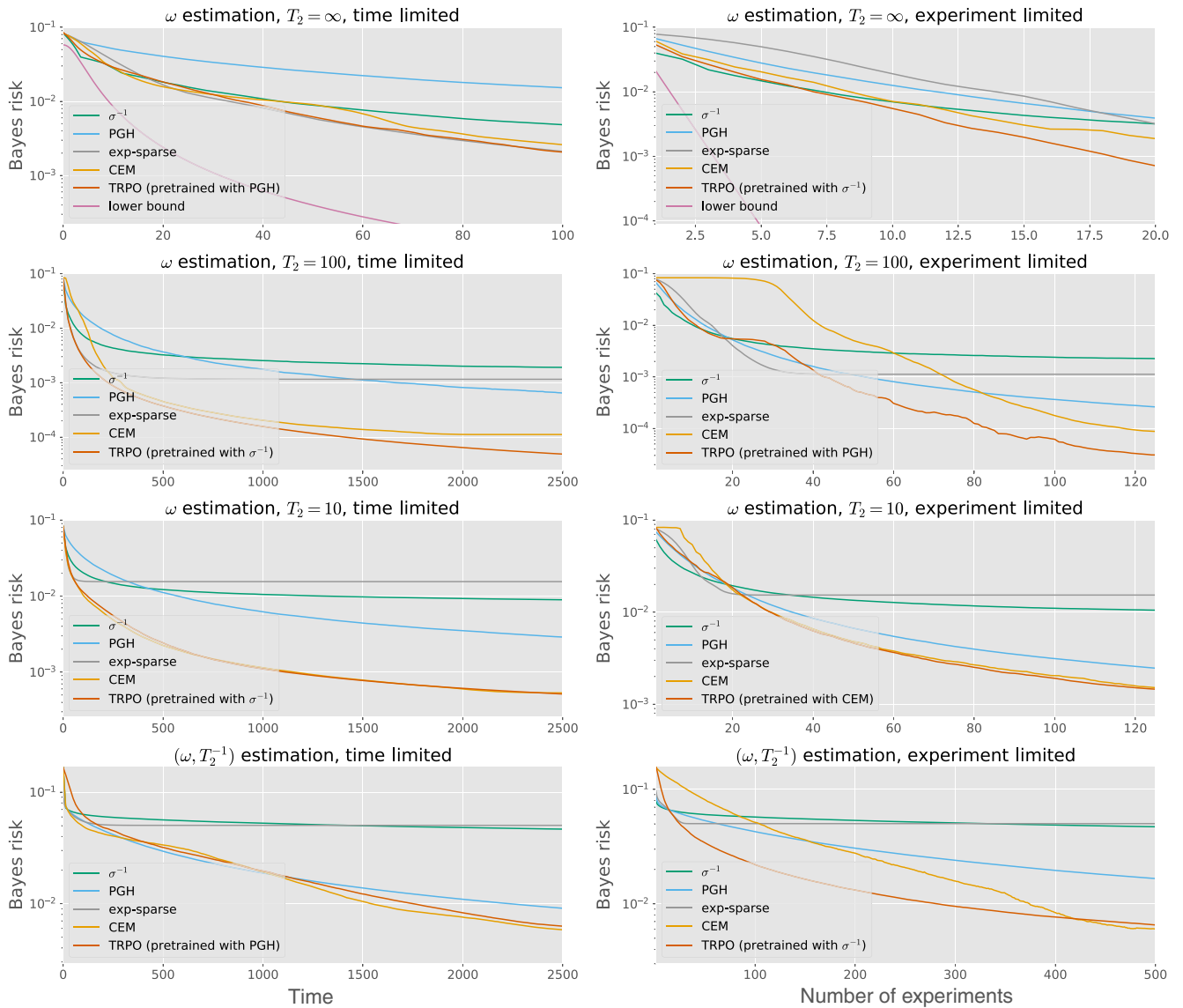


FIG. 2. Comparison of the Bayes risk for different experiment-design heuristics. The right column shows Bayesian estimation with a limited number of experiments and the left column shows Bayesian estimation with a limit on the available time (limited to the maximal value plotted on the x axis). We study frequency estimation without T_2 relaxation (top row) and with T_2 relaxation (second and third rows, with different values of T_2 as stated in the plot titles) as well as the simultaneous estimation of the frequency ω and relaxation rate T_2^{-1} (bottom row). The Bayes risk is calculated numerically from 10^4 episodes; see Appendix B for details. TRPO is pretrained with the heuristic specified in parentheses. The lines are linear interpolants to guide the eye.

Each estimation problem defines a RL environment [see Fig. 1(c)], which is either time limited or experiment limited. In the former case, the available time T per episode is limited, while in the latter case, the number of experiments N per episode is fixed. The first case is relevant if time is the limiting resource, while the second case is relevant if measurements are expensive (e.g., if experiments involve probing sensitive substances such as biological tissue). In practice, there may be constraints on both T and N , which could be easily taken into account by creating a RL environment accordingly.

As an observation for the NN after the k th experiment, we choose the expected value $E_{\theta|D_k}(\theta)$ and the covariance $\text{cov}_{\theta|D_k}(\theta)$ over the posterior (that generalizes the variance in the case of single-parameter estimation), the previous actions from the current episode (maximum of 30 actions), and the spent time or the number of experiments in the current episode (for the time-limited case or the experiment-limited case, respectively).

Several heuristics have been developed for estimation problems (i) and (ii). As an example for a nonadaptive strategy, we consider a heuristic that chooses exponentially

sparse times [10], $t_k = (9/8)^k$, denoted as the “exp-sparse heuristic” in the following.

Furthermore, we consider two adaptive heuristics. We define the first one as $t_k = \text{tr}[\text{cov}_{\theta|D_{k-1}}(\boldsymbol{\theta})]^{-1/2}$, and we call it the “ σ^{-1} heuristic.” This represents a generalization to multiparameter estimation of a heuristic that was derived for estimation problem (i) (ω estimation, $T_2 \rightarrow \infty$) by Ferrie *et al.* [10]. For single-parameter estimation, the σ^{-1} heuristic chooses the times t_k as the inverse standard deviation of $\boldsymbol{\theta}$ over the posterior and is optimal in the greedy sense and only in the asymptotic limit $N \rightarrow \infty$.

The second adaptive heuristic that we consider is the particle guess heuristic (PGH) [11]. It is based on the SMC framework, which uses a particle filter to represent probability distributions such as $p(\boldsymbol{\theta}|D_k)$ [6] (see Appendix A). The PGH chooses times as the inverse distance between two particles $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \Theta$ sampled from $p(\boldsymbol{\theta}|D_{k-1})$, $t_k = \|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2^{-1}$. In the case of single-parameter estimation, the PGH is a proxy for the σ^{-1} heuristic but it is faster to compute (given the particle filter) and introduces additional randomness (compared with the σ^{-1} heuristic).

Let us turn to the results depicted in Fig. 2. In all examples, we consider uniform priors for $\omega \in (0, 1)$ and, in the case of multiparameter estimation, also for $T_2^{-1} \in (0.09, 0.11)$. Uniform priors are the logical choice when no prior knowledge of the parameters is available. Also note that over the course of a Bayesian estimation, many different posteriors (i.e., priors for future experiments) are reached. For the multiparameter estimation examples, we consider, instead of one experiment, 100 independent and identical experiments in each step so as to facilitate the estimation of T_2^{-1} , and we also give the averaged outcome

of these experiments as an additional observation to the RL agent.

We consider three different heuristics for pretraining TRPO—the σ^{-1} heuristic, the PGH, and a CEM-trained NN—but plot only the results for the best of these three. For the data in Fig. 2, CEM is implemented with one hidden layer with 16 neurons, and TRPO is implemented with a NN with two hidden layers with 64 neurons each. The NN heuristics outperform the conventional heuristics in all examples. TRPO performs better than CEM when the limiting resource is used only partly. In the presence of T_2 relaxation, times that exceed T_2 tend to yield no information, which explains why the Bayes risk saturates for the exp-sparse heuristic. The largest advantage of a NN heuristic is found for the example of time-limited ω estimation with $T_2 = 10$. Compared with the PGH, we find an improvement in the Bayes risk by more than 1 order of magnitude.

Generally, the performance of NN heuristics is remarkable given that (for the single-parameter estimation problems considered in Fig. 2) conventional heuristics such as the PGH are used in experiments [18–21] and are considered to be the best practical choice, with near-optimal performance [11,12]. This raises questions about the ultimate performance that could be achieved with an optimal heuristic. However, known lower bounds for the Bayes risk such as the Bayesian Cramér–Rao bound (BCRB) [51,52] usually become very loose when they are optimized with respect to heuristics (for a detailed discussion, see Appendix C). In Fig. 2, we provide lower bounds for estimation problem (i). Remarkably, the lower bound in the experiment-limited case can, in principle, be

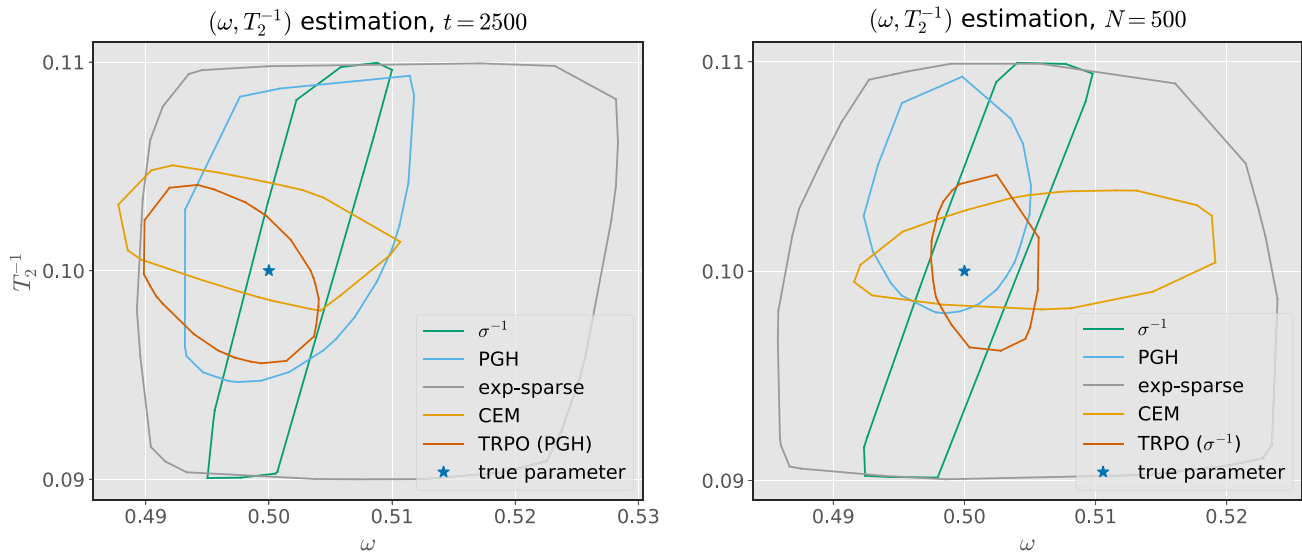


FIG. 3. Ninety-five percent credible regions for multiparameter estimation of (ω, T_2^{-1}) . The credible regions correspond to the uncertainty in the posterior distributions obtained by running one Bayesian estimation (episode) for each heuristic while keeping the true parameter fixed. All heuristics use a uniform prior for $\omega \in (0, 1)$, $T_2^{-1} \in (0.09, 0.11)$.

saturated with use of a phase estimation protocol based on the semiclassical quantum Fourier transform [24]. However, this protocol requires the implementation of σ_z rotations dependent on previous measurement outcomes. This is not possible in our example because σ_z rotations are not part of the available resources. Lower bounds provide an ultimate limit to the performance but, considering their looseness, leave us in the dark about the true potential of experiment-design heuristics.

Another question concerns how experiment designs (i.e., the evolution times) chosen by the NN heuristics compare with those from the conventional heuristics. In Appendix E, we compare the distribution of experiment designs for the adaptive heuristics used in Fig. 2.

While the Bayes risk is used to compare the expected (average) performance of heuristics, it is an important advantage of the Bayesian approach over the frequentist one that it provides credible regions as a practical tool for comparing single runs of parameter estimation (episodes) [26,27]. Let us revisit the multiparameter problem considered in the two bottom panels in Fig. 2. This time, we run only one Bayesian estimation of (ω, T_2^{-1}) with each heuristic and visualize their performance by plotting 95% credible regions [26,27]; see Fig. 3. Since the Bayesian estimation is subject to fluctuations (such as stochastic measurement outcomes), the shape of the credible regions fluctuates between different estimations. For the specific example shown, we see that TRPO provides the smallest uncertainties compared with the other heuristics as judged by the area of the credible regions.

VI. DISCUSSION AND CONCLUSION

We propose using NNs as experiment-design heuristics and training (i.e., optimize) them using RL. The properties of the estimation problem and the quantum experiments and the availability of resources are taken into account during the training, which results in tailored NN-based experiment-design heuristics.

A big advantage of our method lies in its versatility and adaptivity, as we discuss in the following. On the computational side, limitations are mostly related to slow numerical Bayesian updates or a large number of calls to the heuristic, which slows down the generation of training data and thus increases the run time for training the neural networks. In cases when Bayesian estimation involves many experiments, one can partly sacrifice adaptivity and switch to calling the NN heuristic only every n th experiment, in which case the NN would design the next n experiments rather than just one experiment, and thus the number of calls to the heuristic is, in principle, under control. On the other hand, efficient numerical Bayesian updates are a prerequisite for efficient Bayesian quantum estimation in the first place. Therefore, as a rule of thumb, our method can

be applied to adaptive Bayesian quantum estimation whenever the Bayesian updates can be computed efficiently. This applies to all kinds of Bayesian quantum sensors and arbitrary constraints on the available resources.

When it comes to the performance of NN heuristics, we obviously cannot guarantee that RL always succeeds or that in practice the NN heuristics will be more successful than existing heuristics. However, modern RL algorithms such as trust region policy optimization are known to perform well for a wide range of problems [46]. While we demonstrate the success of TRPO for a total of eight estimation problems in the context of phase estimation including examples of multiparameter estimation, we can be optimistic that RL will also work for other estimation problems. Also note that despite phase estimation being one of the best studied examples of Bayesian quantum estimation [7,10,11,19–21], our NN heuristics are able to outperform the established heuristics.

The practical success of adaptive Bayesian estimation often depends on the run time of data processing (Bayesian update, choice of an adaptive experiment design). Use of NNs as experiment-design heuristics introduces a computational overhead compared with the fastest existing experiment-design heuristics such as the PGH. On the other hand, improvements in measurement precision achieved by the NN heuristics may easily outweigh the drawback of an increased run time. This trade-off has to be assessed dependent on the estimation problem and its concrete realization. In our model study, the run time for one call to a NN heuristic is always shorter than the corresponding numerical Bayesian update (single-core computation in both cases). For more complicated modeling of the sensor or for an estimation up to larger precision, the run time of the Bayesian update increases even more, and we expect that the run time will be dominated by the numerical Bayesian update. Moreover, the effective run time of the NN per experiment could be reduced by use of smaller NNs or, as discussed before, by calling the NN heuristics only every n th experiment. Alternatively, it could be interesting to use NN heuristics as “warm-up heuristics” (i.e., using another heuristic afterward), especially if faster heuristics are available that are known to be asymptotically optimal. Furthermore, run-time considerations can be less important when other resources are scarce; for example, when experiments involve probing sensitive substances, we may want to minimize the number of experiments rather than the time (which is the reason why we consider not only time-limited estimation problems but also experiment-limited estimation problems).

In conclusion, we propose and demonstrate a machine-learning method to create fast and strong experiment-design heuristics for Bayesian quantum estimation. The method uses imitation and reinforcement learning for training NNs to become experiment-design heuristics. To

make the method independent of the availability of an expert heuristic for imitation learning, we show that expert heuristics can be found with an evolutionary strategy (the cross-entropy method).

We provide the complete source code [53] used for this work so as to facilitate the application of the presented method in experiments and to related problems such as the detection of time-dependent signals [26,54] and adaptive Bayesian state tomography [8,9].

ACKNOWLEDGMENTS

L.F. thanks Paul Knott for helpful discussions about RL. L.F. and D.B. acknowledge support from the Deutsche Forschungsgemeinschaft (Grant No. BR 5221/1-1). L.F. acknowledges financial support from the European Research Council under the Starting Grant GQCOP (Grant No. 637352) and from the Austrian Science Fund (FWF) through SFB BeyondC (Grant No. F7102). J.S. was supported by the Otto A. Wipprecht Stiftung.

APPENDIX A: BAYES' LAW AND THE SEQUENTIAL MONTE CARLO ALGORITHM

Bayes' law for updating our knowledge of $\theta \in \Theta$ according to the measurement outcome d_k of the k th experiment is given by

$$p(\theta|D_k) = \frac{p(d_k|\theta)p(\theta|D_{k-1})}{p(d_k)}, \quad (\text{A1})$$

where $p(\theta|D_k)$ is our updated knowledge (posterior), $p(\theta|D_{k-1})$ is our knowledge before the k th experiment, $p(d_k|\theta)$ is the likelihood of measuring d_k , and $p(d_k)$ is a normalization, $p(d_k) = E_\theta [p(d_k|\theta)]$. The exact solution for the Bayesian update is generally intractable. Instead, we use an inference algorithm based on the sequential Monte Carlo algorithm [6,7,55]. The idea is to represent the probability distributions $p(\theta)$ and $p(\theta|D_k)$ by a discrete approximation $\sum_{j=1}^n w_j \delta(\theta - \theta_j)$ with n so-called particles with positive weights w_j and positions $\theta_j \in \Theta$. Then, for a Bayesian update, we only need to update the weights of each particle by calculating $p(d_k|\theta_j)$. This means that for the j th particle we have to simulate the experiment for $\theta = \theta_j$ and use the Born rule to find $p(d_k|\theta_j)$. The expected value in the definition of $p(d_k)$ reduces to a simple sum over the particles of the prior.

The particle locations need to be resampled if too many weights are close to zero (i.e., the particle filter of $p(\theta|D_k)$ is impoverished). We use QInfer's implementation [27] of the Liu-West resampling algorithm [25] (with default parameter $a = 0.98$ [27]) with $n = 2 \times 10^3$ particles for RL environments without decoherence and for RL environments with multiparameter estimation. For the environments with ω estimation and finite T_2 , we use $n = 2 \times 10^4$ particles. Particle numbers are chosen sufficiently large to

provide an accurate approximation of $p(\theta|D_k)$ (i.e., when increasing the number of particles, the Bayes risk does not change). Generally, there is a trade-off between a smaller number of particles, which leads to reduced run times but sacrifices accuracy, and a larger number of particles, which provides a more accurate approximation at the cost of an increased run time.

APPENDIX B: BAYES RISK

Let us consider a sequence of k experiments designed with the experiment-design heuristic h . Let $\hat{\theta}_k : D_k \mapsto \hat{\theta}_k(D_k)$ be an estimator of θ after k experiments, and let $L[\hat{\theta}_k(D_k)|\theta]$ be a loss function that quantifies the deviation of $\hat{\theta}_k(D_k)$ from θ . For an experiment-design heuristic h , the risk s of $\hat{\theta}_k$ is defined as (usually denoted by R , which denotes the reward in this work)

$$s(\hat{\theta}_k, h|\theta) = E_{D_k|\theta} \left(L[\hat{\theta}_k(D_k)|\theta] \right). \quad (\text{B1})$$

The experiment-design heuristic h determines the experiment designs E_k that influence the estimate $\hat{\theta}_k(D_k, E_k)$. However, to simplify the notation we write $\hat{\theta}_k(D_k)$ instead of $\hat{\theta}_k(D_k, E_k)$. For a definition of the expected value $E_{D_k|\theta}$, see the main text after Eq. (1). Equation (B1) represents the risk typically associated with a choice of an estimator $\hat{\theta}_k$ and corresponds to the expected (with respect to measurement outcomes D_k) loss. The Bayes risk represents a way to additionally take into account prior knowledge of θ in the form of a probability distribution $p(\theta)$ (the prior) on Θ . Given a prior $p(\theta)$, the Bayes risk is defined as

$$r[\hat{\theta}_k, h|p(\theta)] = E_\theta \left[s(\hat{\theta}_k, h|\theta) \right]. \quad (\text{B2})$$

A common choice for a loss function is the quadratic loss $L[\hat{\theta}_k(D_k)|\theta] = \|\hat{\theta}_k(D_k) - \theta\|_2^2$ [7]. Then the estimator that minimizes the Bayes risk, the Bayes estimator, is given by the expectation over the posterior, $\hat{\theta}_k(D_k) = E_{\theta|D_k}(\theta)$. For the quadratic loss, the risk is also known as the ‘‘mean squared error’’ and the Bayes estimator is also known as the ‘‘minimum mean squared error estimator.’’

1. Derivation of Eq. (1)

We find from Eq. (B2) that

$$r[\hat{\theta}_k, h|p(\theta)] = E_\theta \left[E_{D_k|\theta} \left(\|\hat{\theta}_k(D_k) - \theta\|_2^2 \right) \right] \quad (\text{B3})$$

$$= E_{D_k} \left[E_{\theta|D_k} \left(\|\hat{\theta}_k(D_k) - \theta\|_2^2 \right) \right], \quad (\text{B4})$$

where we used the definition of the expected values together with Bayes' law $p(D_k|\theta) = p(\theta|D_k)p(D_k)/p(\theta)$.

Next we insert the Bayes estimator in Eq. (B4) so as to write the Bayes risk solely as a function of the heuristic h , which corresponds to Eq. (1):

$$r[h|p(\theta)] = E_{D_k} \left[E_{\theta|D_k} \left(\left\| E_{\theta|D_k}(\theta) - \theta \right\|_2^2 \right) \right] \quad (\text{B5})$$

$$= E_{D_k} \left(\sum_j \text{var}_{\theta_j|D_k} [\theta_j] \right) \quad (\text{B6})$$

$$= E_{D_k} [\text{tr}(\text{cov}_{\theta|D_k}[\theta])]. \quad (\text{B7})$$

2. The Bayes risk in the presence of limited resources

As discussed in the main text, the available resources for experiment designs may lead to episodes with different numbers of experiments. A Bayes risk that corresponds to the situation where the available resources are exhausted can easily be generalized from Eq.(1) as

$$r[h, p(\theta)] = E_{D_{\text{end}}} (\text{tr}[\text{cov}_{\theta|D_{\text{end}}}(\theta)]), \quad (\text{B8})$$

where D_{end} denotes all measurement outcomes for an episode. This means the expectation $E_{D_{\text{end}}}$ must be taken with respect to all data sets D_k that are compatible with the available resources.

3. Numerical approximation of the Bayes risk

Let us first consider how to approximate Eq. (B5), which is relevant for the experiment-limited case considered in our model study. The Bayes risk in Eq. (B5) involves expected values over $p(\theta|D_k)$ and $p(D_k)$. The posterior $p(\theta|D_k)$ is represented in the SMC framework by a particle filter with weights w_j and particle locations $\theta_j \in \Theta$, which allows us to approximate the expected value of a function $f(\theta)$ as $E_{\theta|D_k}[f(\theta)] = \sum_j w_j f(\theta_j)$. The same approximation is used to compute the reward. Note that we set

$$\text{tr}(\text{cov}_{\theta}[\theta]) = \text{const} \quad (\text{B9})$$

to a constant numerical value to avoid fluctuations of the reward originating from numerical uncertainties in the particle filter representing the prior $p(\theta)$.

The expected value over $p(D_k)$ is approximated as a sample mean. We sample 10^4 episodes with the RL environment. To each episode corresponds a series of measurement outcomes D_k , and we can calculate $\text{tr}[\text{cov}_{\theta|D_k}(\theta)]$. The properties of the RL environment ensure that an episode with data D_k is sampled with probability $p(D_k)$. Then we can approximate the expected value over $p(D_k)$ as the mean $(1/10^4) \sum_{D_k} \text{tr}[\text{cov}_{\theta|D_k}(\theta)]$, where the sum

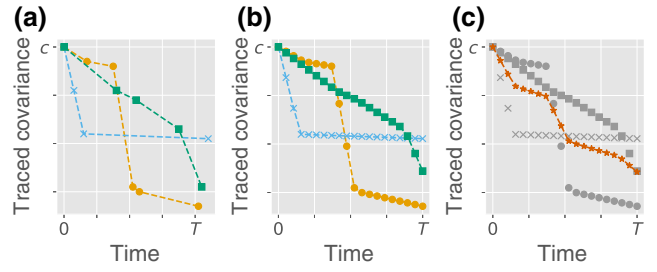


FIG. 4. Illustration of Bayes risk calculation for time-limited environments. (a) Three time series, each of which corresponds to one episode (i.e., one Bayesian estimation): the “zeroth” data point at time 0 corresponds to the uncertainty c in the prior [see Eq. (B9)], and the k th data point in each time series corresponds to the k th experiment of that episode; its x coordinate is the accumulated time used for the first k experiments, and its y coordinate is the traced covariance $\text{tr}[\text{cov}_{\theta|D_k}(\theta)]$. The time limit is T , which is overstepped only with the last experiment of each episode. Linear interpolation of each time series (dashed lines) and evaluation of the linear interpolants at equally spaced times spanning $[0, T]$ yields (b), depicted with 20 equally spaced times. Then, we can take a timewise mean of the data (i.e., for each of the 20 equally spaced times, we calculate the mean over all time series). This gives us the mean of the interpolated traced covariance [depicted as red stars in (c)], which serves as an approximate time-resolved Bayes risk.

runs over all 10^4 sampled series of measurement outcomes D_k .

The time-limited case we consider in our model study is an example where episodes can have different numbers of experiments. Moreover, we defined the time-limited case such that an episode ends when the agent has consumed more time than the given time limit. This means with the last experiment the RL agent (as well as the other heuristics) can go beyond the time limit. For the comparison of heuristics, this is not relevant mainly because all other experiment designs must lie within the time limit.

For the time-limited case, it would be insightful to have a time-resolved Bayes risk in analogy to the experiment-limited case, where for every number of experiments we can compute a Bayes risk. Figure 4 illustrates how we approximate such a time-resolved Bayes risk (actually, we average over 10^4 episodes and use 200 equally spaced times for interpolation). This approximation of the time-resolved Bayes risk is used for the time-limited cases shown in Fig. 2.

APPENDIX C: LOWER BOUNDS FOR THE BAYES RISK

Since we are interested in optimizing experiment-design heuristics, we would like to know whether our heuristics are actually close to optimal or whether we are missing out on potentially much larger improvements. To this end, a lower bound L for the Bayes risk would be useful such

that $r[h|p(\theta)] \geq L$ for all heuristics h . A well-known lower bound for the Bayesian risk is the BCRB (also known as the van Trees inequality [51]) [see Eq. (10) in Ref. [52]]:

$$r[h|p(\theta)] \geq \frac{1}{E_\theta [\text{tr}(I_\theta [p(D_k|\theta)] + I_\theta [p(\theta)])]}, \quad (\text{C1})$$

where we defined

$$I_\theta [f(\theta)] = E_{D_k|\theta} ([\nabla_\theta \log f(\theta)]^\top [\nabla_\theta \log f(\theta)]), \quad (\text{C2})$$

where ∇_θ is the vector differential operator with respect to θ . In Eq. (C1), $I_\theta [p(D_k|\theta)]$ is the Fisher information matrix and $I_\theta [p(\theta)]$ is a contribution due to the initial prior $p(\theta)$. We can see that the BCRB depends on the experiment-design heuristic since the Fisher information matrix depends on the likelihood and thus on the experiment designs. Therefore, a lower bound L could be defined as

$$L = \inf_h \frac{1}{E_\theta [\text{tr}(I_\theta [p(D_k|\theta)] + I_\theta [p(\theta)])]} \quad (\text{C3})$$

$$= \frac{1}{\sup_h E_\theta [\text{tr}(I_\theta [p(D_k|\theta)])] + E_\theta [\text{tr}(I_\theta [p(\theta)])]}, \quad (\text{C4})$$

where we find the infimum (supremum) with respect to all heuristics h , which respect the available resources, such that $r[h|p(\theta)] \geq L$ for all h .

Before we discuss the problems related to the lower bound (C3), we have to address another issue related to the BCRB. The BCRB can be applied only if some regularity conditions are fulfilled [52]; in particular, the prior $p(\theta)$ must converge toward zero at the endpoints of domain $\Theta \ni \theta$, where Θ is assumed to be a finite interval for each parameter θ_j . Therefore, the BCRB cannot be applied directly in our case because we use uniform priors. However, the particle filter used to represent a uniform prior can be seen as an equally good approximation of a prior that vanishes at the endpoints but is otherwise flat. For example, the following ansatz for a prior (up to normalization) converges toward a uniform prior on (a, b) for $k \rightarrow \infty$:

$$p_k(\theta) \propto \frac{2}{(1 + e^{-2k(x-a)})(1 + e^{2k(x-b)})} - 1. \quad (\text{C5})$$

Figure 5 shows $p_k(\theta)$ for different values of k .

Let us compute the BCRB for estimation problem (i), frequency estimation without dephasing. In the case of single-parameter estimation, the Fisher information matrix reduces to a scalar, the Fisher information. Using the likelihood in Eq. (4) with $T_2 \rightarrow \infty$, we find the Fisher

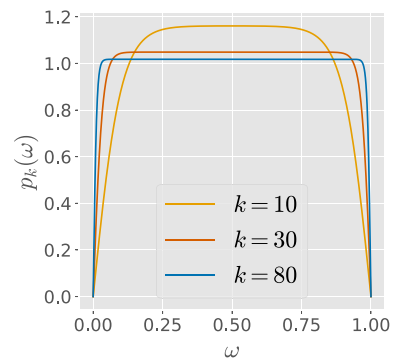


FIG. 5. Analytic approximations of a flat prior that vanishes at the endpoints of its domain. The plotted functions correspond to Eq. (C5) with $a = 0$ and $b = 1$ for different values of k .

information for the first experiment,

$$I_\omega [p(D_1|\omega)] = t_1^2, \quad (\text{C6})$$

where t_1 is the evolution time of the experiment. The Fisher information for k experiments designed with an adaptive heuristic h reads

$$I_\omega [p(D_k|\omega)] = \sum_{j=1}^k I_\omega [p(D_k^{(j)}|\omega)], \quad (\text{C7})$$

where $D_k^{(j)}$ denotes the first j coefficients (measurement outcomes) of D_k . We see from Eq. (C6) that $I_\omega [p(D_k^{(j)}|\omega)] = t_{D_k^{(j)}}^2$, where $t_{D_k^{(j)}}$ is the evolution time of the j th experiment when we observe data D_k . This means that the total Fisher information in Eq. (C7) does not depend on the parameter ω . Thus, taking the expected value in Eq. (C4) is trivial. Similarly, the maximization with respect to heuristics h is straightforward: in the time-limited case with time limit T , it is easy to see that the sum in Eq. (C7) is maximized by choosing a single experiment of time T . Then the supremum with respect to heuristics h_T (i.e., heuristics that consume at most time T) is given by

$$\sup_{h_T} I_\omega [p(D_k|\omega)] = T^2, \quad (\text{C8})$$

which would correspond to a heuristic h_T that deterministically chooses a single experiment with evolution time T . The lower bound for the Bayes risk reads

$$L_T = \frac{1}{T^2 + E_\theta [\text{tr}(I_\theta [p(\theta)])]}. \quad (\text{C9})$$

This also shows that optimizing the BCRB does not necessarily provide useful insights when it comes to finding good experiment designs; in the case of qubit measurements, performing only a single experiment yields at best one bit of information about θ .

In the experiment-limited case, where heuristics are allowed to consume arbitrary amounts of time, minimizing the BCRB is pointless because $I_\omega [p(D_k|\omega)]$ becomes infinite if $t_k \rightarrow \infty$. Alternatively, an information-based lower bound can be derived. Similarly to Refs. [10,24], we write ω in its binary expansion,

$$.\omega_1\omega_2\omega_3\cdots \quad (\text{C10})$$

k experiments yield at best k bits of information about ω such that we know the first k bits $\omega_1, \dots, \omega_k$ with certainty. Then possible ω values are restricted to an interval $[\omega_{\min}, \omega_{\max}]$ whose end points are obtained by setting all remaining bits to zero or to 1 (i.e., $\omega_{\min} = .\omega_1 \cdots \omega_k$ and $\omega_{\max} = .\omega_1 \cdots \omega_k 111 \cdots$) such that $\omega_{\max} - \omega_{\min} = 2^{-k}$. Since we use uniform priors in this work, we assume a uniform probability density for ω on this interval. This means our best estimate is to choose the mean of this interval, $\hat{\omega}_k = .\omega_1 \cdots \omega_k 1$. This yields the following Bayes risk:

$$L_k = E_\omega \left[E_{D_k|\omega} \left(\|\hat{\omega}_k - \omega\|_2^2 \right) \right] \quad (\text{C11})$$

$$= E_\omega \left[(\hat{\omega}_k - \omega)^2 \right] \quad (\text{C12})$$

$$= 2^{-k} \int_{\omega_{\min}}^{\omega_{\max}} d\omega (\hat{\omega}_k - \omega)^2 \quad (\text{C13})$$

$$= \frac{2^{-2(k+1)}}{3}. \quad (\text{C14})$$

Remarkably, this Bayes risk is achievable with an adaptive phase estimation protocol, which is equivalent to a semiclassical quantum Fourier transform [24]. However, to implement the semiclassical quantum Fourier transform, one needs to implement σ_z rotations dependent on previous measurement outcomes (or equivalently one needs a freedom in the choice of the measurement basis), which is not allowed in our scenario and therefore cannot be realized with our available resources.

To sum up, this illustrates how difficult it is to find meaningful lower bounds for the Bayes risk of experiment-design heuristics that are subject to certain available resources. In Fig. 2, we use L_k from Eq. (C14) and L_T from Eq. (C9) [with a prior as specified in Eq. (C5) with $k = 30$] as lower bounds for the Bayes risk for estimation problem (i).

APPENDIX D: DETAILS ON THE TRAINING OF THE NEURAL NETWORKS

Our results are obtained with the NumPy 1.17.3 [56], QInfer 1.0a1 [27], GYM 0.14.0 [57], PyTorch 1.3.1 [58], STABLE BASELINES 2.9.0 [48], and TENSORFLOW 1.14.0 [59] libraries for PYTHON 3.6.7. CEM and TRPO use fully connected neural networks (CEM, one hidden layer with 16 neurons; TRPO, two hidden layers with 64 neurons each).

TABLE I. Limits on the number of experiments and the available time for all estimation problems considered in this work.

Estimation problem	Experiment limit	Time limit
ω estimation, $T_2 = \infty$, experiment limited	20	10^{27}
ω estimation, $T_2 = \infty$, time limited	100	100
ω estimation, $T_2 = 100$, experiment limited	125	10^{27}
ω estimation, $T_2 = 100$, time limited	1000	2500
ω estimation, $T_2 = 10$, experiment limited	125	10^{27}
ω estimation, $T_2 = 10$, time limited	1000	2500
(ω, T_2^{-1}) estimation, experiment limited	500	10^{27}
(ω, T_2^{-1}) estimation, time limited	4000	2500

1. The combination of CEM and TRPO

While CEM takes into account only the accumulated reward of full episodes, RL uses all the training data consisting of actions with corresponding observations and rewards. This allows TRPO to take into account the performance of single actions to improve the overall performance (at the end of episodes). If we pretrain TRPO with a CEM-trained NN (as a known heuristic), we obtain a purely machine-learning-based method for finding strong NN heuristics. The advantage of this two-step procedure over using only one of the algorithms is that we can use CEM to explore the policy space (for our RL environments, CEM has proven to be better than TRPO in this respect) and TRPO to optimize the heuristic further. A further speedup and possibly an improvement of this method could be achieved by passing the neural network directly from CEM to TRPO (avoiding imitation learning). However, in the current implementation, this is not possible because the neural networks used for CEM and TRPO are of different shape and not compatible.

2. Cross-entropy method

The input layer of the NN is defined by the observation. The output layer is determined by the number of actions (one action: time), and we choose 16 neurons in the hidden layer. The layers are fully connected. The hidden layer has the rectified linear unit as its activation function and the output layer has the softmax function as its activation function [60].

A schematic representation of the algorithm is given in Fig. 1(d). The weights of the neural network form a vector \mathbf{x} . A generation is sampled from a Gaussian distribution $\mathbf{x}_i \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ with mean $\boldsymbol{\mu}$ and covariance $\Sigma = \mathbb{1}/2$. Before the first iteration, we sample $\boldsymbol{\mu}$ from $\mathcal{N}(\mathbf{0}, \Sigma)$. A generation consists of $K = 100$ individuals (NNs). By running one episode (interacting with the RL environment) with each NN, we determine the $K_\epsilon = 10$ fittest individuals

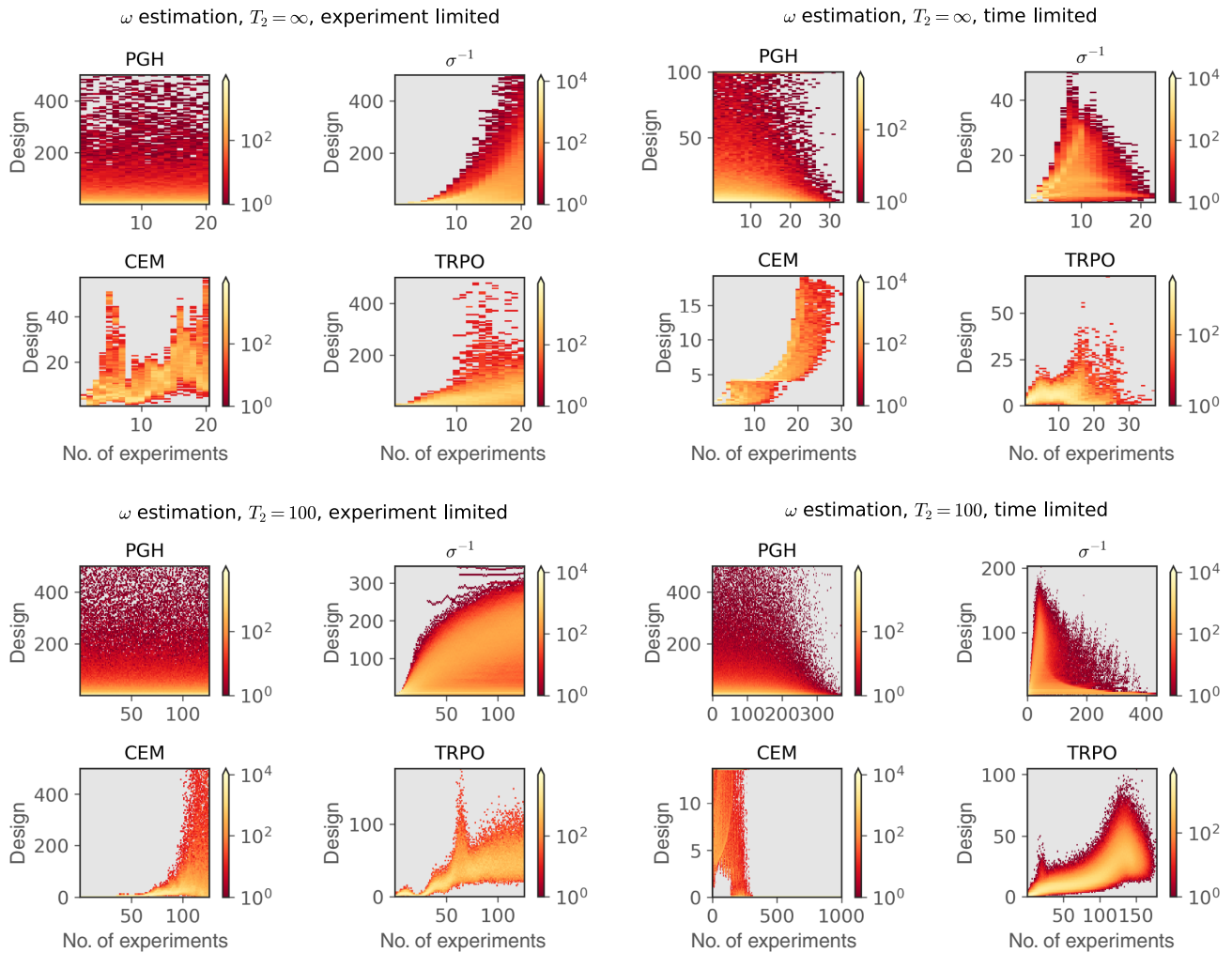


FIG. 6. Distribution of experiment designs (y axis). On the x axis we show the experiment number (e.g., $x = 10$ corresponds to the tenth experiment in each episode). For readability, we cut off y data at $t = 500$. In the time-limited cases, CEM sometimes reaches the numerical threshold for the number of experiments by choosing many experiments with very small t . This manifests itself in the corresponding plots as a bright line close to $t = 0$ (see, e.g., the case of ω estimation, $T_2 = 100$, time limited). If the gray background of the plots is visible, there are no experiment designs for the corresponding values.

$(\mathbf{x}_1, \dots, \mathbf{x}_{K_\epsilon})$ as those with the largest reward accumulated over an episode. This concludes one iteration. The next generation is sampled from the distribution $\mathcal{N}(\boldsymbol{\mu}_{\text{new}}, \Sigma)$ with $\boldsymbol{\mu}_{\text{new}} = 1/K_\epsilon \sum_{j=1}^{K_\epsilon} \mathbf{x}_j$. We run CEM for $N = 1000$ iterations. The final solution is given by $\boldsymbol{\mu}_{\text{new}}$ calculated in the last iteration. CEM is used to find five NNs for each RL environment, and we plot the results in Fig. 2 only for the NN heuristic that achieves the smallest Bayes risk after the maximum time or the maximal number of measurements. There are also slightly more sophisticated versions of the cross-entropy method; for instance, using Gaussian distributions with an adaptive variance [47]. Note that the cross-entropy method for discrete actions is a reinforcement learning algorithm [40], while the cross-entropy method for continuous action spaces, which we use in this work, is despite its similar name an evolutionary strategy.

3. Trust region policy optimization

We use TRPO [46] with the default multilayer perceptron policy as implemented in STABLE BASELINES 2.9.0 [48]. The hyperparameters that differ from their default values are $\gamma = 1$, $\lambda = 0.92$, and $\text{vf}_{\text{stepsize}} = 0.0044$. These hyperparameters are found to yield good results during initial testing of the algorithm, without our doing a rigorous hyperparameter tuning. In particular, we do not use discounted rewards by setting the discount factor $\gamma = 1$, because even without reward damping, the rewards are defined such that they tend to decay exponentially with the number of experiments. Furthermore, smaller values for λ such as 0.92 often work well together with large values of γ .

Pretraining is also implemented in STABLE BASELINES 2.9.0 [48] using the Adam optimizer [60]. Deviations from

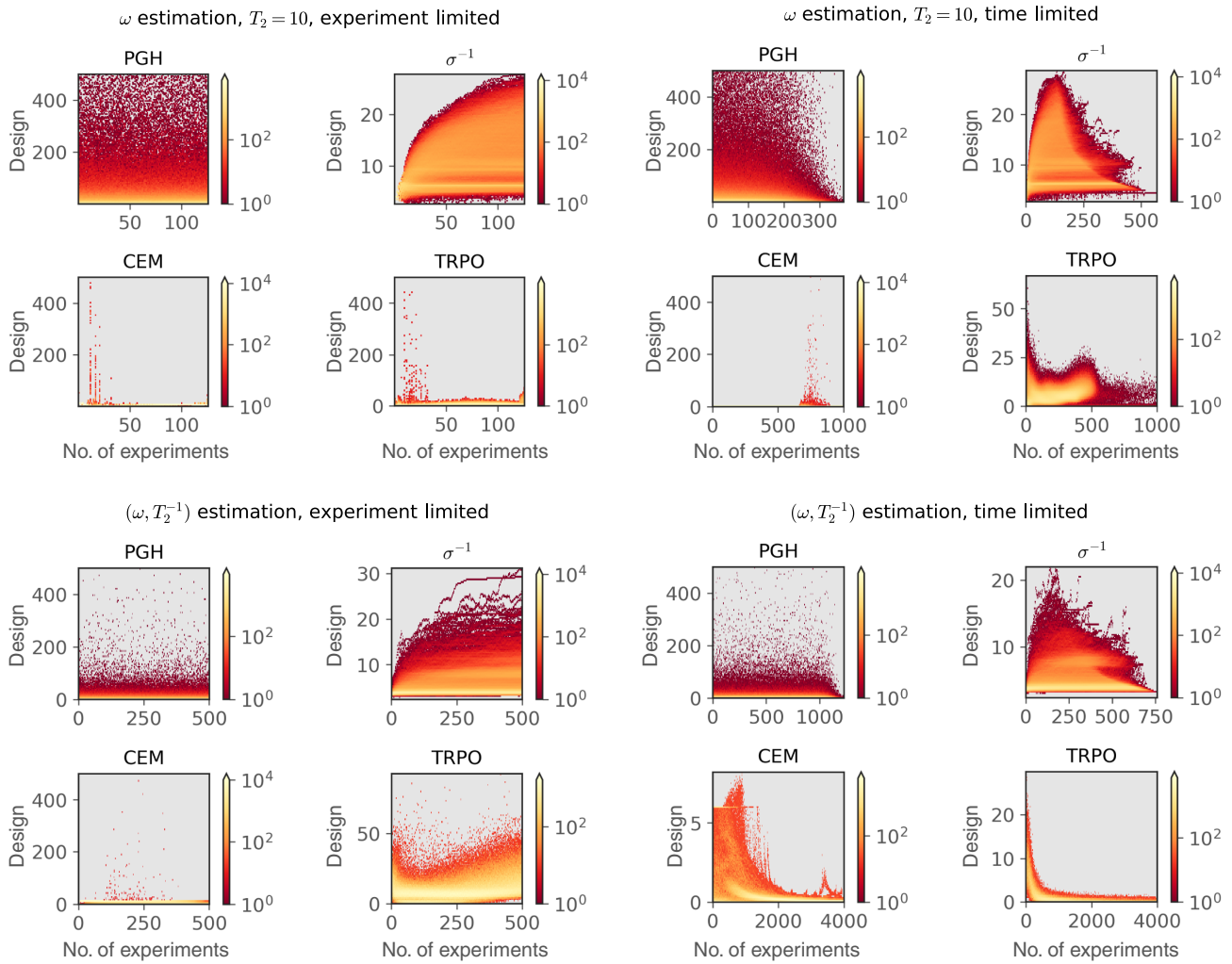


FIG. 7. As for Fig. 6, the only difference being that this figure shows data for different estimation problems as specified in the titles of the subfigures.

the default parameters [48] are as follows: we use 10^4 (10^3) episodes, sampled from a known heuristic, as an expert data set. Pretraining runs with 10^4 (10^3) epochs (the number of training iterations on the expert data set) and a batch size for the expert data set of 100 (10) (values in parentheses are used for the time-limited (ω, T_2^{-1}) estimation to speed up the computation).

Training runs for 500 iterations. However, we use an exit condition that can stop the training earlier. The exit condition stops training if the policy entropy drops below 0.005 [48].

TRPO is pretrained once per heuristic (PGH, σ^{-1} , the best of five CEM-trained NNs), and then trained five times for each of the pretrainings; we plot the results in Fig. 2 only for the best NN heuristic (i.e., with the smallest Bayes risk at the end of the episodes).

4. Resource limits of the RL environments

Table I shows the limits on the number of experiments and the available time for all estimation problems. For numerical convenience, time-limited (experiment-limited) problems also have a limit on the number of measurements (the available time). In particular, for time-limited problems, a limit on the number of experiments helps to avoid a RL agent choosing very many small experiment times, which would lead to very long episodes (i.e., to very long run times). Also from a physical perspective, small experiment times are not sensible due to overheads for readout and preparation. Such overheads can be taken into account as a dead time of the sensor between experiments, which can be easily implemented as a part of the RL environments. In the case of experiment-limited environments, the time limit 10^{27} avoids issues with large numerical values

and it is chosen large enough for the exp-sparse heuristic with 500 experiments, which chooses exponentially increasing times.

APPENDIX E: COMPARING EXPERIMENT DESIGNS FROM DIFFERENT HEURISTICS

Here we provide additional information about the distribution of experiment designs (i.e., experiment times) chosen by the four adaptive heuristics discussed in the main text: the PGH, the σ^{-1} heuristic, a neural network trained with the CEM, and a neural network trained with TRPO. Figures 6 and 7 show the frequency of experiment designs for different experiments; that is, which times are chosen for the first experiment of each episode (Bayesian estimation), which times are chosen for the second experiment, and so on. The plotted experiment designs correspond precisely to the data used for Fig. 2. The data consist for each estimation problem and each heuristic of 10^4 episodes. The diversity we find for different experiment-design heuristics is remarkable. Also the diversity between different environments underpins that experiment-design heuristics should be tuned to the particular estimation problem. However, the data presented in Figs. 6 and 7 do not reveal information about the adaptivity of the heuristics.

-
- [1] C. W. Helstrom, *Quantum Detection and Estimation Theory*, Mathematics in Science and Engineering Vol. 123 (Elsevier, 1976).
- [2] S. L. Braunstein and C. M. Caves, Statistical Distance and the Geometry of Quantum States, *Phys. Rev. Lett.* **72**, 3439 (1994).
- [3] L. J. Fiderer, J. M. E. Fraïsse, and D. Braun, Maximal Quantum Fisher Information for Mixed States, *Phys. Rev. Lett.* **123**, 250502 (2019).
- [4] V. Bužek, R. Derka, G. Adam, and P. Knight, Reconstruction of quantum states of spin systems: From quantum bayesian inference to quantum tomography, *Ann. Phys.* **266**, 454 (1998).
- [5] R. Blume-Kohout, Optimal, reliable estimation of quantum states, *New J. Phys.* **12**, 043034 (2010).
- [6] F. Huszár and N. M. T. Houlby, Adaptive Bayesian quantum tomography, *Phys. Rev. A* **85**, 052120 (2012).
- [7] C. E. Granade, C. Ferrie, N. Wiebe, and D. G. Cory, Robust online hamiltonian learning, *New J. Phys.* **14**, 103013 (2012).
- [8] C. Granade, J. Combes, and D. G. Cory, Practical Bayesian tomography, *New J. Phys.* **18**, 033024 (2016).
- [9] C. Granade, C. Ferrie, and S. T. Flammia, Practical adaptive quantum tomography, *New J. Phys.* **19**, 113017 (2017).
- [10] C. Ferrie, C. E. Granade, and D. G. Cory, How to best sample a periodic probability distribution, or on the accuracy of Hamiltonian finding strategies, *Quantum Inf. Process.* **12**, 611 (2012).
- [11] N. Wiebe, C. Granade, C. Ferrie, and D. G. Cory, Hamiltonian Learning and Certification Using Quantum Resources, *Phys. Rev. Lett.* **112**, 190501 (2014).
- [12] N. Wiebe and C. Granade, Efficient Bayesian Phase Estimation, *Phys. Rev. Lett.* **117**, 010503 (2016).
- [13] N. Friis, D. Orsucci, M. Skotiniotis, P. Sekatski, V. Dunjko, H. J. Briegel, and W. Dür, Flexible resources for quantum metrology, *New J. Phys.* **19**, 063044 (2017).
- [14] K. S. Kravtsov, S. S. Straupe, I. V. Radchenko, N. M. T. Houlby, F. Huszár, and S. P. Kulik, Experimental adaptive Bayesian tomography, *Phys. Rev. A* **87**, 062122 (2013).
- [15] G. I. Struchalin, I. A. Pogorelov, S. S. Straupe, K. S. Kravtsov, I. V. Radchenko, and S. P. Kulik, Experimental adaptive quantum tomography of two-qubit states, *Phys. Rev. A* **93**, 012103 (2016).
- [16] R. J. Chapman, C. Ferrie, and A. Peruzzo, Experimental Demonstration of Self-Guided Quantum Tomography, *Phys. Rev. Lett.* **117**, 040402 (2016).
- [17] Z. Hou, J.-F. Tang, C. Ferrie, G.-Y. Xiang, C.-F. Li, and G.-C. Guo, [arXiv:1908.01082](https://arxiv.org/abs/1908.01082) (2019).
- [18] J. Wang, S. Paesani, R. Santagati, S. Knauer, A. A. Gentile, N. Wiebe, M. Petruzzella, J. L. O'Brien, J. G. Rarity, A. Laing, and M. G. Thompson, Experimental quantum Hamiltonian learning, *Nat. Phys.* **13**, 551 (2017).
- [19] S. Paesani, A. A. Gentile, R. Santagati, J. Wang, N. Wiebe, D. P. Tew, J. L. O'Brien, and M. G. Thompson, Experimental Bayesian Quantum Phase Estimation on a Silicon Photonic Chip, *Phys. Rev. Lett.* **118**, 100503 (2017).
- [20] A. Lumino, E. Polino, A. S. Rab, G. Milani, N. Spagnolo, N. Wiebe, and F. Sciarrino, Experimental Phase Estimation Enhanced by Machine Learning, *Phys. Rev. Appl.* **10**, 044033 (2018).
- [21] R. Santagati, A. A. Gentile, S. Knauer, S. Schmitt, S. Paesani, C. Granade, N. Wiebe, C. Osterkamp, L. P. McGuinness, J. Wang, M. G. Thompson, J. G. Rarity, F. Jelezko, and A. Laing, Magnetic-Field Learning Using a Single Electronic Spin in Diamond with One-Photon Readout at Room Temperature, *Phys. Rev. X* **9**, 021019 (2019).
- [22] K. Chaloner and I. Verdinelli, Bayesian experimental design: A review, *Stat. Sci.* **10**, 273 (1995).
- [23] A. Sergeevich, A. Chandran, J. Combes, S. D. Bartlett, and H. M. Wiseman, Characterization of a qubit Hamiltonian using adaptive measurements in a fixed basis, *Phys. Rev. A* **84**, 052315 (2011).
- [24] A. M. Childs, J. Preskill, and J. Renes, Quantum information and precision measurement, *J. Mod. Opt.* **47**, 155 (2000).
- [25] J. Liu and M. West, in *Sequential Monte Carlo Methods in Practice* (Springer, New York, 2001) p. 197.
- [26] C. Granade, C. Ferrie, I. Hincks, S. Casagrande, T. Alexander, J. Gross, M. Kononenko, and Y. Sanders, QInfer: Statistical inference software for quantum applications, *Quantum* **1**, 5 (2017).
- [27] C. Granade, C. Ferrie, S. Casagrande, I. Hincks, M. Kononenko, T. Alexander, and Y. Sanders, QInfer: Library for Statistical Inference in Quantum Information (2016).

- [28] M. P. V. Stenberg, Y. R. Sanders, and F. K. Wilhelm, Efficient Estimation of Resonant Coupling Between Quantum Systems, *Phys. Rev. Lett.* **113**, 210404 (2014).
- [29] M. P. V. Stenberg, O. Köhn, and F. K. Wilhelm, Characterization of decohering quantum systems: Machine learning approach, *Phys. Rev. A* **93**, 012122 (2016).
- [30] I. Hincks, T. Alexander, M. Kononenko, B. Soloway, and D. G. Cory, [arXiv:1806.02427](https://arxiv.org/abs/1806.02427) (2018).
- [31] A. Hentschel and B. C. Sanders, Machine Learning for Precise Quantum Measurement, *Phys. Rev. Lett.* **104**, 063603 (2010).
- [32] A. Hentschel and B. C. Sanders, Efficient Algorithm for Optimizing Adaptive Quantum Metrology Processes, *Phys. Rev. Lett.* **107**, 233601 (2011).
- [33] N. B. Lovett, C. Crosnier, M. Perarnau-Llobet, and B. C. Sanders, Differential Evolution for Many-Particle Adaptive Quantum Metrology, *Phys. Rev. Lett.* **110**, 220501 (2013).
- [34] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, and *et al.*, A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play, *Science* **362**, 1140 (2018).
- [35] P. Palittapongarnpim, P. Wittek, E. Zahedinejad, S. Vedaie, and B. C. Sanders, Learning in quantum control: High-dimensional global optimization for noisy quantum dynamics, *Neurocomputing* **268**, 116 (2017).
- [36] T. Fösel, P. Tighineanu, T. Weiss, and F. Marquardt, Reinforcement Learning with Neural Networks for Quantum Feedback, *Phys. Rev. X* **8**, 031084 (2018).
- [37] V. Dunjko and H. J. Briegel, Machine learning & artificial intelligence in the quantum domain: A review of recent progress, *Rep. Prog. Phys.* **81**, 074001 (2018).
- [38] F. Schäfer, M. Kloc, C. Bruder, and N. Lörch, A differentiable programming method for quantum control, *Mach. Learn.: Sci. Technol.* **1**, 035009 (2020).
- [39] H. Xu, J. Li, L. Liu, Y. Wang, H. Yuan, and X. Wang, [arXiv:1904.11298](https://arxiv.org/abs/1904.11298) (2019).
- [40] J. Schuff, L. J. Fiderer, and D. Braun, Improving the dynamics of quantum sensors with reinforcement learning, *New J. Phys.* **22**, 035001 (2020).
- [41] V. Cimini, I. Gianani, N. Spagnolo, F. Leccese, F. Sciarrino, and M. Barbieri, Calibration of Quantum Sensors by Neural Networks, *Phys. Rev. Lett.* **123**, 230502 (2019).
- [42] A. A. Gentile, B. Flynn, S. Knauer, N. Wiebe, S. Paesani, C. E. Granade, J. G. Rarity, R. Santagati, and A. Laing, [arXiv:2002.06169](https://arxiv.org/abs/2002.06169) (2020).
- [43] C. You, N. Bhusal, A. Lambert, C. Dong, A. Perez-Leija, R. d. J. Leon-Montiel, A. Javaid, and O. S. Magana-Loaiza, [arXiv:1909.08060](https://arxiv.org/abs/1909.08060) (2019).
- [44] P. A. Knott, A search algorithm for quantum state engineering and metrology, *New J. Phys.* **18**, 073033 (2016).
- [45] R. Rubinstein, The cross-entropy method for combinatorial and continuous optimization, *Methodol. Comput. Appl. Probab.* **1**, 127 (1999).
- [46] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, in *International Conference on Machine Learning* (2015) p. 1889.
- [47] A. Pourchot and O. Sigaud, [arXiv:1810.01222](https://arxiv.org/abs/1810.01222) (2018).
- [48] 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, <https://github.com/hill-a/stable-baselines> (2018).
- [49] J. Schulman, Deep RL Bootcamp Lecture 6: Nuts and Bolts of Deep RL Experimentation (Berkeley CA, 2017), <https://youtu.be/8EcdCK9KaQ>.
- [50] S. Schmitt, T. Gefen, F. M. Stürner, T. Unden, G. Wolff, C. Müller, J. Scheuer, B. Naydenov, M. Markham, S. Pezzagna, J. Meijer, I. Schwarz, M. Plenio, A. Retzker, L. P. McGuinness, and F. Jelezko, Submillihertz magnetic spectroscopy performed with a nanoscale quantum sensor, *Science* **356**, 832 (2017).
- [51] H. L. V. Trees, *Detection, Estimation, and Modulation Theory, Part I* (John Wiley & Sons, Inc., New York, 1968).
- [52] R. D. Gill, and B. Y. Levit, Applications of the van trees inequality: A bayesian cramér-rao bound, *Bernoulli* **1**, 59 (1995).
- [53] The source code and data are available at <https://doi.org/10.6084/m9.figshare.11927043>.
- [54] M. Isard and A. Blake, Condensation—conditional density propagation for visual tracking, *Int. J. Comput. Vis.* **29**, 5 (1998).
- [55] A. Doucet and A. M. Johansen, A tutorial on particle filtering and smoothing: Fifteen years later, *Handb. Nonlinear Filter.* **12**, 3 (2009).
- [56] S. van der Walt, S. C. Colbert, and G. Varoquaux, The NumPy array: A structure for efficient numerical computation, *Comput. Sci. Eng.* **13**, 22 (2011).
- [57] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, [arXiv:1606.01540](https://arxiv.org/abs/1606.01540) (2016).
- [58] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, in *Advances in Neural Information Processing Systems* (2019) p. 8024.
- [59] M. Abadi, *et al.*, TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems (2015), software available from tensorflow.org.
- [60] M. A. Nielsen, *Neural Networks and Deep Learning*, Vol.25, (Determination press San Francisco, CA, USA, 2015).
- [61] D. P. Kingma and J. Ba, [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014).