Model-aware reinforcement learning for high-performance Bayesian experimental design in quantum metrology

Federico Belliardo[1], Fabio Zoratti[2], Florian Marquardt[3], and Vittorio Giovannetti[4]

[1]NEST, Scuola Normale Superiore, I-56126 Pisa, Italy
[2]Scuola Normale Superiore, I-56126 Pisa, Italy
[3]Max Planck Institute for the Science of Light and Physics Department, University of Erlangen-Nuremberg, 91058 Erlangen, Germany
[4]NEST, Scuola Normale Superiore and Istituto Nanoscienze-CNR, I-56126 Pisa, Italy

Quantum sensors offer flexibility in control during estimation, allowing manipulation across various parameters by the experimenter. For each sensing platform, determining the optimal controls to enhance the sensor's precision remains a challenging task. While an analytical solution may be unattainable, machine learning presents a promising approach for many systems of interest, especially considering the capabilities of modern hardware. We introduce a versatile procedure capable of optimizing a wide range of problems in quantum metrology and estimation by combining model-aware reinforcement learning (RL) with Bayesian estimation via particle filtering. To achieve this, we addressed the challenge of integrating the many non-differentiable steps of the estimation process, such as measurements and particle filter resampling, into the training routine. Our RL-based approach is suitable for optimizing both non-adaptive and adaptive strategies using a neural network. We provide an implementation of this technique in the form of a Python library called qsensoropt, along with several pre-built applications for relevant physical platforms, including NV centers, photonic circuits, and optical cavities. Using our method, we have achieved results that surpass the current state-of-the-art in experimental design for numerous tasks. Beyond Bayesian estimation, by leveraging model-aware RL, it is also possible to find optimal controls for minimizing the Cramér-Rao bound, based on Fisher information.

1 Introduction

In recent years, the synergy between machine learning and quantum information has attracted increasing attention. These two technological fields can complement each other in various ways. On the one hand, quantum technologies, particularly quantum computers, have the potential to address classical machine learning challenges, such as classification and sampling, using both classical and quantum data [1, 2, 3]. On the other hand, traditional machine learning can enhance quantum information tasks, including state preparation [4, 5, 6, 7], optimal quantum feedback [8], error correction [9], device calibration [10, 11, 12, 13], characterization [14], and quantum tomography [15, 16, 17]. This work falls within the latter category, utilizing model-aware reinforcement learning [18, 19, 20, 8] (RL) to identify optimized adaptive and non-adaptive control strategies for application-relevant tasks in quantum metrology and estimation [21]. The problem of optimal experimental design [22] has already been addressed using machine learning techniques [23, 24, 25, 26, 27, 28]. In this manuscript, we advance further by proposing a tool applicable to a broader range of problems, providing control solutions that are either superior to or easier to implement than their model-free counterparts. We present the theory, along with two relevant examples, and the details of the mathematical approach, while in [29], the full range of optimization problems we have solved with this technique is discussed.

1.1 Model-based and model-free reinforcement learning.

The purpose of this section is to briefly review the key definitions in the field of reinforcement learning (RL) to better convey the novelty of our approach and clarify the associated terminology. Reinforcement learning is a mathematical framework for modeling decision-making in environments where outcomes are random but can be influenced by the actions of an agent, which is the mechanism making decisions. RL provides a systematic approach to optimizing the actions of the agent, taking into account both immediate and future consequences.

The objective of the actions and observations performed by the RL agent is to execute a task within the environment. After each execution, a loss function is computed, and the agent's strategy is updated based on this loss value. This process is referred to as training. If the model of the environment is known, it can be incorporated into the gradient of the loss function to update the agent's parameters. This forms the basis of model-based RL. Conversely,

if the dynamic of the environment is completely unknown, the agent must implicitly learn how the environment responds to its actions in order to minimize the loss. While several studies have applied RL to quantum metrology [30, 31, 32, 33, 34], the innovation introduced here is the use of a model-aware approach, which, as we demonstrate through comparison with the results of [34], offers a more effective method for training optimized strategies. For further clarification on terminology and the distinction between RL and other decision-making frameworks, we refer to [35]. While the term "model-based reinforcement learning" typically refers to scenarios where the model of the system and the optimization of the strategy occur simultaneously, in this manuscript, we rely on a pre-characterized model of the system, hence we adopt the term "model-aware RL". Our use of model-aware RL is akin to the concept of digital twins in industry, where a complete model of a product or system is simulated on a computer to optimize its performance. We provide evidence that incorporating the system model into the training process enhances efficiency, enabling better control solutions than those derived from model-free RL. With our tool, we achieve high-performance Bayesian experimental design, where high performance refers to the ability of our methods to surpass strategies obtained through other techniques.

1.2 Review of the literature

In the following paragraph, we review the field of optimization in quantum metrology, discussing the strengths and limitations of various approaches in comparison to the framework we propose. Similar challenges to those addressed by our method have been studied previously. These works on optimizing quantum metrology can be broadly categorized into four classes, which we will now present one by one.

The first class includes well-established competitor frameworks, such as the toolbox proposed by Meyer *et al.*, which employs a variational approach for optimizing measurements and states [36]. In this work, the authors introduce a scheme to optimize probe state preparation and measurements to maximize the classical Fisher information obtained from the process. They demonstrate the success of this approach by applying it to multiphase estimation with GHZ states and to the problem of triangulating the position of a spin with three NV centers. In contrast to our approach, the variational toolbox introduced by Meyer *et al.* does not allow for Bayesian estimation or consider adaptive strategies. A similar approach can be found in the library QuantEstimation [37], which implements different bounds for quantum metrological tasks (Fisher information, Holevo-Cramér-Rao bound, quantum Ziv-Zakai bound, and Bayesian estimation), along with various optimization algorithms that we have not considered (particle swarm optimization and differential evolution). However, both QuantEstimation and the variational toolbox studied by Meyer *et al.* do not account for the use of neural networks for adaptive experiments, unlike our framework. We now turn to the two libraries, QInfer [38] and Optbayesexpt [39], which are similar tools that optimize Bayesian experimental design for a range of experimental situations, but only consider greedy optimizations, i.e., one measurement at a time, via an approximation of the information gain per measurement. In contrast, the approach we present in this manuscript can plan measurements several steps ahead, potentially for the entire duration of the estimation. Another important tool recently introduced in the domain of optimal quantum metrology is a quantum comb-based approach for the simultaneous optimization of states, measurement, and estimator in one-shot Bayesian experiments, as proposed in [40]. Here, the authors consider a single encoded probe and a single measurement, from which they perform Bayesian estimation. Using the formalism of higher-order quantum operations, they develop a mechanism based on semidefinite programming to find the optimal probe, quantum measurement, and estimator function in both single- and multi-parameter scenarios. The main limitation of this approach is its extension to the multi-shot scenario, where the complexity of quantum comb optimization grows exponentially.

The second class of papers concerns those based on the optimization of Fisher information [41, 42, 43, 44, 32, 45, 33, 46]. Most of these approaches rely on some form of reinforcement learning or GRAPE, but the Fisher information analysis is limited to local estimation, and these works typically lack coverage of adaptive measurements or are only applicable to specific platforms (e.g., NV centers).

The third class includes theoretical works that advocate for the necessity of optimal control in quantum metrology and conceptually shape the working principles of our approach, though without presenting any concrete implementation [24, 25, 20, 47, 48]. In [24], the authors introduce MODE (Machine-learning Optimized Design of Experiments), a collaborative research program aimed at leveraging modern machine learning and statistical programming tools across different scientific domains.

The fourth class consists of applications of variational quantum circuits to specific platforms and tasks. These are generally non-adaptive (with two exceptions [49, 50]) and can be Bayesian [51, 52, 53, 54, 55] or based on the quantum Fisher information [56, 57, 58]. The work [56] is particularly noteworthy, as it studies the optimization of super-resolution imaging for observing Earth.

Among all the works discussed in this literature review, none present a framework capable of addressing both Bayesian and frequentist estimations, adap-

tive and non-adaptive metrology, across different platforms. This manuscript introduces a new framework designed to fill this gap through an innovative combination of techniques from statistics and machine learning. We have chosen an approach to Bayesian estimation based on particle filtering (or sequential Monte Carlo) due to the speed of the estimation performed in this manner and the ability to parallelize multiple simulations of the experiment. This approach involves many non-differentiable steps, such as simulating measurements and resampling from the posterior distribution, which we needed to account for when computing gradients.

2 Quantum metrology with reinforcement learning

In this section, we discuss how reinforcement learning (RL) is applied to quantum metrology. After providing a general overview of our scheme's working principle, we discuss how parameters of interest are encoded in quantum metrology, followed by a description of the Bayesian inference technique we employ. We then introduce two fundamental concepts in our approach: the measurement loop and the resources-loss paradigm. Finally, we define the loss function for various estimation tasks and comment on the technical aspects of the training process.

Given a specific physical platform and metrological task, the set of tunable parameters in the experiment is identified. These are the parameters that can be adjusted using various experimental controls, such as knobs and dials. The agent's role is to decide the optimal settings for these controls before each measurement on the system. We train the agent to optimally control these parameters, minimizing the error metric through a gradient descent optimization procedure, using backpropagation to compute the derivatives throughout the entire history of the estimation process. The loss function minimized during the training is related to the final estimation error obtained after completing the experiment and performing Bayesian inference. In the examples presented in this paper, the agent is a small neural network, while the environment it interacts with encompasses the entire experimental setup, including the system that stores the estimation results and performs the Bayesian inference. In other words, the processed information extracted from the experiment is considered part of the environment.

This estimation procedure has been abstracted and decoupled from the specifics of any particular sensor or physical platform, allowing our tool to function as a versatile optimization method for quantum sensors. We demonstrate the broad applicability of our methodology by optimizing a wide range of estimation tasks on the nitrogen-vacancy (NV) center platform [59, 43], for both single- and multi-parameter metrology, including DC magnetometry [34], AC magnetometry, decoherence estimation [60], and hyperfine coupling characterization [61]. In the domain of photonic circuits, we studied tasks such as multiphase discrimination, the agnostic Dolinar receiver [62], and coherent state classification, both for cases where the states are classically known and where they must be learned from a quantum training set. For frequentist estimation, we explored the sensing of detuning frequency in a driven optical cavity [31]. In this paper, we present the applications to DC magnetometry and quantum communication, while other applications to NV centers (AC-field, decoherence, hyperfine coupling estimation) and photonic circuits (quantum machine learning with photonic circuits, multiphase estimation, coherent state classification) are discussed in [29].

Encoding of the probe

In quantum metrology, we deal with an environment or process characterized by a fixed number of parameters, denoted as $\boldsymbol{\theta} \in \Theta$. These parameters are unknown, and our goal is to estimate them. To accomplish this, a *quantum probe* with known dynamics interacts with the environment or undergoes the process of interest. By measuring the state of the probe, which now depends on $\boldsymbol{\theta}$, we can extract information about these parameters, assuming the dynamic of the interaction is completely understood. In this context, quantum probes are systems that are well-characterized, easily manipulable, and often quite simple. Further details on the encoding of the probe can be found in the Supplementary Information Appendix A.

For the purpose of optimizing control strategies, the evolution of the probe and the measurement outcomes are simulated. The training phase is separated from the deployment, where the measurement process takes place on the actual sensor during in the experiment.

Bayesian estimation and particle filter

Bayesian estimation is a step-by-step method for updating information about the unknown parameters of a system we are measuring, by refining a probability distribution after each measurement. The process begins with the definition of a *prior distribution* $\pi(\boldsymbol{\theta})$ over the parameters $\boldsymbol{\theta}$, which encapsulates our initial belief about the value of the unknown parameters before any measurements are taken. After the first measurement, this prior is updated to form the *posterior distribution*, denoted as $P(\boldsymbol{\theta})$. To represent the posterior distribution, we use the particle filter method [63, 64, 65] (PF), which approximates it as an ensemble of points $\{\boldsymbol{\theta}_j\}_{j=1}^N$ in the parameter space Θ, with each point assigned a weight $\{w_j\}_{j=1}^N$, where N is the number of particles. Essentially, we approximate

the posterior distribution with a sum of δ-functions, as follows:

$$P(\boldsymbol{\theta}) \simeq \sum_{j=1}^{N} w_j^t \delta(\boldsymbol{\theta} - \boldsymbol{\theta}_j) \,, \tag{1}$$

Initially, the particles are sampled from $\pi(\boldsymbol{\theta})$ and the weights are set to $w_j = \frac{1}{N}$. As new information becomes available, the weights are updated accordingly. From the particle filter, we derive an estimator $\widehat{\boldsymbol{\theta}} \in \Theta$ for $\boldsymbol{\theta}$, which in our application is either the mean of the posterior or the most likely value for $\boldsymbol{\theta}$. When the measurements on the quantum probe are weak (as opposed to projective), it is also necessary to account for the measurement backaction for each possible value of the unknown parameters $\boldsymbol{\theta}$. For further details, refer to the Supplementary Information Appendix B.

Input and output of the controlling agent

A "summary" of the information encapsulated in the Bayesian posterior represented by the particle filter (PF)—such as the mean and covariance matrix of the distribution $P(\boldsymbol{\theta})$—is provided as input to the neural network agent, which subsequently outputs the control settings. It is crucial for the agent to be specifically trained on the experiment it is designed to optimize. This necessitates that precise values for parameters like decoherence rates and visibilities are known and integrated into the simulation, unless these parameters are included in the set of $\boldsymbol{\theta}$ to be estimated. In this manner, the knowledge gained about $\boldsymbol{\theta}$ through measurements can be adaptively utilized by the agent to control both the evolution of the system and the measurements performed on the probe, with the objective of maximizing the final precision of the estimation. We envision conducting experiments using a small, trained agent deployed on fast hardware, such as a Field Programmable Gate Array (FPGA), situated in close proximity to the experimental setup.

The measurement loop

The metrological task is simulated as a sequence of consecutive operations, referred to as the *measurement loop*, as illustrated in Fig. 1. Within this loop, for each iteration numbered from $t = 0$ to $M - 1$, a single measurement is conducted. We proceed by describing the generic iteration of the loop (specifically the t+1-th iteration), which consists of three steps. As indicated in the caption of Fig. 1, we denote the controls generated by the agent for the evolution of the probe and the settings for its measurements as x_{t+1}. The outcome of the measurement is denoted by y_{t+1}, both obtained at the t+1-th iteration of the loop. The objects $\boldsymbol{x}_t := (x_0, x_1, \ldots, x_t)$ and $\boldsymbol{y}_t := (y_0, y_1, \ldots, y_t)$ are tuples containing the controls and measurement outcomes up to time t. The distribution $P(\boldsymbol{\theta}|\boldsymbol{x}_t, \boldsymbol{y}_t)$ represents the Bayesian posterior updated with the outcomes up to step t of the measurement loop.

1. In the case of the adaptive strategy, the selection of x_{t+1} by the agent can be expressed, without loss of generality, through the mapping

$$x_{t+1} := \mathcal{F}_{\boldsymbol{\lambda}}\{P(\boldsymbol{\theta}|\boldsymbol{x}_t, \boldsymbol{y}_t); \boldsymbol{y}_t; R_t; t\} \,, \tag{2}$$

where r_j denotes the resource consumption at the j-th step of the protocol, and the total resource consumed up to the t-th step is computed as $R_t := \sum_{j=0}^{t} r_j$. Non-adaptive strategies are described by mappings $\mathcal{F}$ that do not depend functionally on $P(\boldsymbol{\theta}|\boldsymbol{x}_t, \boldsymbol{y}_t)$ or $\boldsymbol{y}_t$, expressed as

$$x_{t+1} := \mathcal{F}_{\boldsymbol{\lambda}}\{R_t; t\} \,. \tag{3}$$

The mapping $\mathcal{F}_{\boldsymbol{\lambda}}$ depends on the trainable parameters of the strategy, collectively denoted as $\boldsymbol{\lambda}$, which are later optimized. In the context of the non-adaptive strategies discussed here, the agent simply consists of a list of controls that are applied sequentially in the measurement loop, leading to $\boldsymbol{\lambda} = \boldsymbol{x}_{M-1}$. For all the examples discussed in this manuscript, the neural network has five hidden layers with 64 neurons each, and the activation function used is tanh, known for its effectiveness in approximating smooth functions [66].

2. Assuming the measurements are projective and the probe's state is reinitialized after each iteration, the probability of observing the outcome y_{t+1} at the t+1-th step is given by $p(y_{t+1}|x_{t+1}, \boldsymbol{\theta})$, which is computed using the Born rule according to the known quantum dynamics of the probe coded in the simulation. This probability, henceforth referred to as the "model" relies solely on the controls x_{t+1} and the parameters to be estimated, $\boldsymbol{\theta}$. At this second step of the measurement loop, the outcome y_{t+1}, a stochastic vari-

The precision-resources paradigm

In our framework, each measurement conducted on the probe consumes a certain amount r of a specific "resource", which is considered costly within the context of the experiment and must be defined by the user based on the limitations of the setup. Once the total available resources R are exhausted, the estimation process concludes, and the final value of the estimator $\widehat{\boldsymbol{\theta}}$ is computed. Examples of resources include the total estimation time, which is relevant for the NV center platform, the average number of photons consumed, or the amplitude of a signal, as seen in the Dolinar receiver. For optimizing the metrological task, defining the resource is as crucial as establishing the precision figure of merit. There is no universally correct or incorrect resource for an estimation task; it ultimately depends on the experimentalist's choices and their understanding of the laboratory limitations in implementing the task.

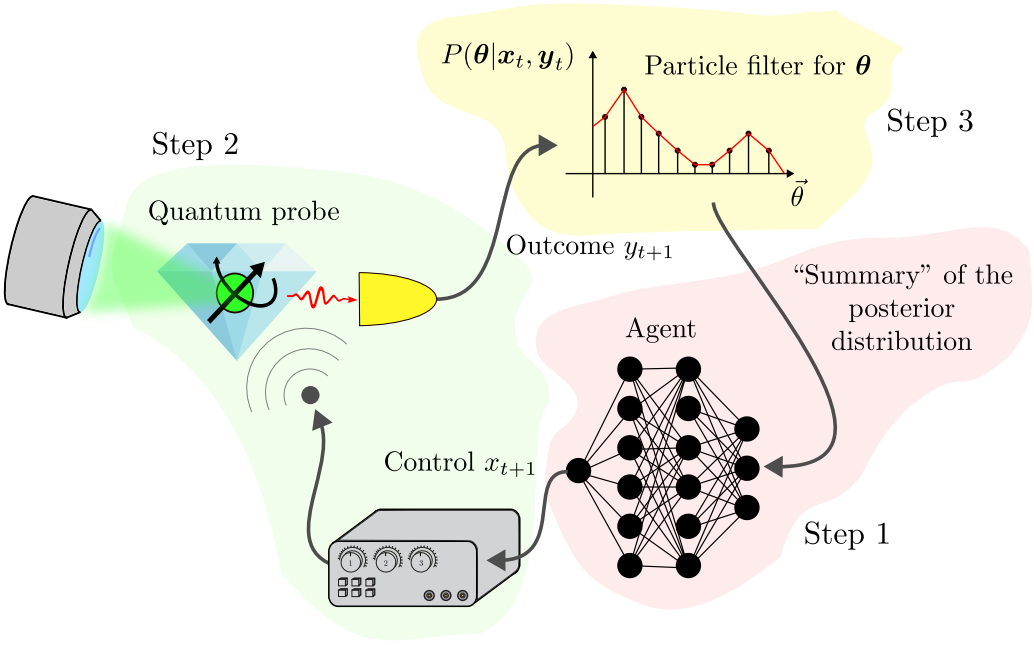


Figure 1: Schematic representation of the three steps of information flow within the measurement loop. The labels correspond to the $(t+1)$-th iteration. In the first step (pink region of the figure), the summary information computed from the particle filter is input into the agent (depicted here as a neural network), which determines the control parameters for both the evolution and measurement of the probe during this iteration, collectively represented by the variable x_{t+1}. In the second step (green region), the parameters $\boldsymbol{\theta}$ are encoded in the probe state, and the measurement is performed, yielding the outcome y_{t+1}. In the third step (yellow region), this outcome is fed into the particle filter, leading to an update of the Bayesian posterior distribution on the parameters $\boldsymbol{\theta}$ and the state of the probe (if applicable).

able, is drawn from the model distribution, expressed as

$$y_{t+1} \sim p(y_{t+1}|x_{t+1}, \boldsymbol{\theta}) \,. \tag{4}$$

If the probe is subjected to a weak measurement, then the outcome probability depends on the entire sequence of previous controls and outcomes because of the measurement backreaction. In this case the sampled outcome is expressed as

$$y_{t+1} \sim p(y_{t+1}|\boldsymbol{x}_{t+1}, \boldsymbol{y}_t, \boldsymbol{\theta}) \,. \tag{5}$$

3. The observation of y_{t+1} is subsequently incorporated into the posterior using Bayes' rule, formulated as

$$P(\boldsymbol{\theta}|\boldsymbol{y}_{t+1}, \boldsymbol{x}_{t+1}) \propto p(y_{t+1}|x_{t+1}, \boldsymbol{\theta}) P(\boldsymbol{\theta}|\boldsymbol{x}_t, \boldsymbol{y}_t) \,. \tag{6}$$

During the first iteration, the prior $\pi(\boldsymbol{\theta})$ is utilized in place of the posterior. If the measurements are weak, the model probability takes the form described in Eq. (5).

The stopping condition of the measurement loop can be trivial, such as setting a maximum number of iterations M, or based on the available resources, for example, imposing a limit on R_t.

Training with model-aware reinforcement learning

The figure of merit for precision depends on the specific metrological task. In the examples concerning the NV center platform, where the parameters $\boldsymbol{\theta}$ are continuous, the mean square error (MSE) is employed. The loss for a single estimation is expressed as follows:

$$\ell(\widehat{\boldsymbol{\theta}}, \boldsymbol{\theta}) := \mathrm{tr}\left[G \cdot (\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta})(\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta})^{\intercal}\right] \,, \tag{7}$$

where $G \geq 0$ represents a positive semidefinite weight matrix, and $\widehat{\boldsymbol{\theta}}$ denotes the mean of the posterior distribution. The weight matrix G determines the contribution of various errors to the loss $\ell(\widehat{\boldsymbol{\theta}}, \boldsymbol{\theta})$ and delineates the parameters of interest from the nuisance parameters, with the latter being assigned corresponding entries of zero in the G matrix.

In discrete estimation tasks, as illustrated subsequently for a photonic platform, both $\boldsymbol{\theta}$ and $\widehat{\boldsymbol{\theta}}$ are discrete, such that $\boldsymbol{\theta}, \widehat{\boldsymbol{\theta}} \in \Theta = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_k\}$. The loss for a single instance of this task can be represented using a Kronecker delta:

$$\ell(\widehat{\boldsymbol{\theta}}, \boldsymbol{\theta}) := 1 - \delta(\widehat{\boldsymbol{\theta}}, \boldsymbol{\theta}) \,, \tag{8}$$

where

$$\widehat{\boldsymbol{\theta}} := \arg\max_{\boldsymbol{\theta}} P(\boldsymbol{\theta}|\boldsymbol{x}_t, \boldsymbol{y}_t) \,, \tag{9}$$

denotes the maximum *a posteriori* estimator. Optimizing the control strategy involves identifying the

agent that minimizes the average loss $\mathbb{E}[\ell(\widehat{\boldsymbol{\theta}}, \boldsymbol{\theta})]$, averaged over all potential choices of $\boldsymbol{\theta}$ and all stochastic processes involved in the estimation of $\boldsymbol{\theta}$, as detailed in the Supplementary Material Appendix D.

Each potential agent is characterized by a set of trainable variables, denoted as $\boldsymbol{\lambda}$, which influence the individual losses of the problem as well as the associated $\mathbb{E}[\ell(\widehat{\boldsymbol{\theta}}, \boldsymbol{\theta})]$. The optimal strategy can be abstractly defined by the value $\boldsymbol{\lambda}^\star := \operatorname{argmin}_{\boldsymbol{\lambda}} \mathbb{E}[\ell(\widehat{\boldsymbol{\theta}}, \boldsymbol{\theta})]$ that minimizes the average loss. The training of the agent is implemented as an iterative algorithm aimed at discovering a strategy that closely approximates the performance of the optimal $\boldsymbol{\lambda}^\star$ through a sequence of recursive updates, denoted as TS(1), TS(2), $\cdots$, TS(I), where "TS" stands for "training step":

$$\boldsymbol{\lambda}_0 \overset{\text{TS}(1)}{\longrightarrow} \boldsymbol{\lambda}_1 \overset{\text{TS}(2)}{\longrightarrow} \cdots \overset{\text{TS}(I)}{\longrightarrow} \boldsymbol{\lambda}_I \simeq \boldsymbol{\lambda}^\star \,, \tag{10}$$

with $\boldsymbol{\lambda}_0$ representing the initial weights of the neural network, initialized using the normal Glorot initializer, and the initial bias set to zero.

The construction of the learning trajectory in Eq. (10) relies on the computation of an estimation $\mathcal{L}(\boldsymbol{\lambda})$ of the average loss $\mathbb{E}[\ell(\widehat{\boldsymbol{\theta}}, \boldsymbol{\theta})]$ associated with an agent $\boldsymbol{\lambda}$. This is typically accomplished by simulating in parallel B estimations of randomly selected values $\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_B$ of the parameters $\boldsymbol{\theta}$. Consequently, we can express:

$$\mathcal{L}(\boldsymbol{\lambda}) := \frac{1}{B} \sum_{k=1}^{B} \ell(\widehat{\boldsymbol{\theta}}_k, \boldsymbol{\theta}_k) \simeq \mathbb{E}[\ell(\widehat{\boldsymbol{\theta}}, \boldsymbol{\theta})] \,, \tag{11}$$

where $\ell(\widehat{\boldsymbol{\theta}}_k, \boldsymbol{\theta}_k)$ denotes the local loss of the k-th estimation, which possesses a functional dependence on $\boldsymbol{\lambda}$ due to the multiple controlling actions of the agent.

Exploiting this dependence, we can compute the gradient $\mathcal{G}(\boldsymbol{\lambda}) := \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda})$ of $\mathcal{L}(\boldsymbol{\lambda})$ using automatic differentiation (AD), executed in reverse through all operations of the measurement loop. The agent parameters are updated at each training step using stochastic gradient descent, as follows:

$$\boldsymbol{\lambda}_i \overset{\text{TS}(i+1)}{\longrightarrow} \boldsymbol{\lambda}_{i+1} = \boldsymbol{\lambda}_i - \alpha \mathcal{G}(\boldsymbol{\lambda}_i) \,, \tag{12}$$

with $\alpha \in (10^{-4}, 10^{-1})$ representing the learning rate. In the reported examples, the Adam optimizer [67] is utilized. This algorithm accumulates past gradients observed during training and uses them to adaptively modify the learning rate for each parameter. The purpose of this modified gradient descent approach is to enhance the training process by smoothing updates, allowing the optimizer to gain momentum in directions with consistent gradients while mitigating oscillations in others. This aids the algorithm in converging more rapidly and prevents it from becoming trapped in areas with noisy or minimal gradients. The algorithm accepts an external learning rate, which serves as a baseline for the adaptive learning rates.

Since the derivatives are propagated through the model for the sensor as described in Eq. (4), this training constitutes a form of model-aware policy gradient reinforcement learning. The gradient descent training of $\boldsymbol{\lambda}$ will converge to a minimum of the loss; however, there is no guarantee that this minimum will be $\boldsymbol{\lambda}^\star$. Given that the loss is defined in terms of the stochastic outcomes $\boldsymbol{y}_t$, special precautions are required to compute an unbiased estimator for its gradient [8], which entails incorporating the log-likelihood terms $\log p(y_{t+1}|\boldsymbol{x}_{t+1}, \boldsymbol{y}_t, \boldsymbol{\theta})$ into the loss. For further details regarding the loss definition and its gradient, refer to the Supplementary Material Appendix D.

When conducting an estimation with a fixed number of measurements $M_{\max}$ or a fixed maximum amount of resources $R_{\max}$, selecting a loss $\mathcal{L}(\boldsymbol{\theta})$ that is sensitive solely to the performance of the estimator $\boldsymbol{\theta}$ at the conclusion of the estimation does not necessarily yield optimal strategies for $M < M_{\max}$ and $R < R_{\max}$. A straightforward solution would be to repeat the optimization for each smaller $R_{\max}$ that requires characterization. However, it is possible to find an approximate solution for all $R \leq R_{\max}$ through a training that optimizes the cumulative loss instead of Eq. (11), expressed as:

$$\mathcal{L}_{\text{cum}}(\boldsymbol{\lambda}) := \frac{1}{M_{\max} B} \sum_{t=0}^{M_{\max}-1} \sum_{k=1}^{B} \ell(\widehat{\boldsymbol{\theta}}_{k,t}, \boldsymbol{\theta}_k) \,. \tag{13}$$

This cumulative loss encourages the agent to learn a strategy that is optimal for all $R \leq R_{\max}$ and has been employed in the examples involving the NV center platform. An alternative version of the loss is the logarithmic loss, which employs the logarithm of the average loss on the batch rather than the average loss in Eq. (13). Further details can be found in Appendix D.4.

Differentiability of the particle filter

The primary component of our approach is the integration of particle filter (PF) Bayes updates with model-based reinforcement learning. This presents a challenge, as PF updates encompass steps where the differentiability necessary for gradient computation is not immediately apparent. As the estimation progresses, the weights of the PF become concentrated on a limited number of particles. To optimize memory utilization, we implement a resampling procedure that, when invoked, extracts a new set of particles $\{\boldsymbol{\theta}'_j\}_{j=1}^N$ in accordance with the posterior distribution $P(\boldsymbol{\theta})$, resetting the weights to $w'_j = \frac{1}{N}$. This resampling procedure comprises three steps, which can be toggled on and off at discretion. These steps include: resampling from the posterior $P(\boldsymbol{\theta})$, perturbing the newly extracted particles, and proposing new particles. We have optimally integrated these steps through a trial-and-error procedure, as detailed in

Supplementary Information Appendix B.3. All these steps entail the extraction of discrete stochastic variables, an operation that, in principle, lacks differentiability and would substantially hinder the subsequent gradient propagation necessary for reinforcement learning.

While the latter two steps can be made differentiable using the reparametrization trick (see Supplementary Information Appendix C.1), addressing the challenge of resampling the discrete PF ensemble necessitates modifying the loss function by incorporating the log-likelihood of the stochastic outcomes, as we do for the measurements. However, for a large number of particles N, this modification would adversely affect the variance of the estimated gradient during simulations. Instead, we utilize importance sampling to extract the new particles from a distribution $Q(\boldsymbol{\theta})$ distinct from the posterior, and we set the new weights proportional to the factor $\frac{P(\boldsymbol{\theta})}{Q(\boldsymbol{\theta})}$, ensuring that the PF consistently represents the posterior [68]. In this manner, the gradient can propagate through a resampling event via the term $P(\boldsymbol{\theta})$ in the weights. Along with the importance sampling, we have implemented the correction introduced by Ścibior and Wood [69] to achieve differentiable resampling, demonstrating its efficacy for the mean square error loss (see Supplementary Information Appendix C.2). This correction complements importance sampling and aims to add the fewest possible log-likelihood terms for particle extraction events to the loss, thereby maintaining the stability of the training. The Bayes rule, being the product of the model probability and the previous posterior, is trivially differentiable.

3 Applications

In this section, we present two applications of model-aware reinforcement learning (RL) to static field magnetometry using nitrogen-vacancy (NV) centers and to quantum communication utilizing the Dolinar receiver. We provide a brief overview of the two distinct problems to demonstrate the utility of our approach.

- **Magnetometry**. In this example, we address magnetic field estimation employing solid-state quantum sensors, specifically a single qubit in diamond known as an NV center. We estimate a single, continuous parameter, which is the precession frequency of the spin, that is proportional to the external magnetic field we want to measure.
- **Quantum communication**. In this example we evaluate the application of RL to the agnostic Dolinar receiver. This task entails the discrimination of two coherent states transmitted by a distant laboratory. There are two variables to be estimated: one is discrete and contains information regarding the message, while the other is continuous, representing the intensity of the signal to be detected. The latter parameter is a nuisance.

For both problems, we successfully identified adaptive and non-adaptive strategies that surpassed the results previously reported in the literature.

Magnetometry with NV centers

The nitrogen-vacancy (NV) centre in diamond is a point defect that enables initialization, detection, and control of its electronic spin, featuring very long quantum coherence time, even at room temperature. As such, it has been used in applications such as magnetometry, thermometry, and stress sensing [70, 71, 59, 72, 73]. The electronic spin is sensitive to magnetic fields; for example, static fields determine the electron Larmor frequency, which can be measured as an accumulated phase by a Ramsey experiment. These experiments are realized by applying two $\pi/2$ pulses to the spin, followed by illumination with green light and detection of the photoluminescence. A single measurement has a binary outcome, yielding ± 1 with probabilities

$$p(\pm 1|\omega, T_2^\star, \tau) := \frac{1}{2} \pm \frac{1}{2} e^{-\tau/T_2^\star} \cos(\omega\tau) \ . \qquad (14)$$

The free evolution time τ is controlled by a trainable agent, while $\omega := \gamma B$ represents the unknown precession frequency to be estimated, which is proportional to the static magnetic field B with $\gamma \simeq 28\,\mathrm{MHz/mT}$. The parameter $T_2^\star$ denotes the transverse relaxation time, serving as the time scale for the dephasing induced by magnetic noise. The optimization of the NV center as a magnetometer has been extensively studied in the literature with analytical tools [74, 75], with numerical methods [76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88], and with machine learning [89, 34, 90]. We conducted multiple estimations over the same parameter ranges chosen in the work of Fiderer *et al.* [34], in order to facilitate an easy comparison of the results. The prior for the frequency ω is uniform in $(0, 1)\,\mathrm{MHz}$. Fig. 2 compares the performances of the optimized adaptive (NN) and non-adaptive strategies against the Particle Guess Heuristic (PGH) [91], which is a commonly referenced strategy in the literature. According to this strategy, the evolution time is then computed as $\tau = (||\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2||_2 + \varepsilon)^{-1}$ with $\varepsilon := 10^{-5}\,\mu\mathrm{s}^{-1}$. The concept behind it is to gauge the width of the probability distribution by extracting two particles from it at random, i.e. $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$. There is also another common approach to assess this dispersion of the posterior, which is the use of the σ^{-1} strategy, i.e., the evolution time τ is selected adaptively as the inverse of the standard deviation of the posterior distribution. Additionally, we introduced a variant of the σ^{-1} strategy [75], named $\sigma^{-1}\&T^{-1}$,

which accounts for the finite coherence time. According to the $\sigma^{-1}\&T^{-1}$ strategy, the next evolution time τ is computed from the covariance matrix Σ of the current posterior distribution as $\tau = \left[\mathrm{tr}(\Sigma)^{\frac{1}{2}} + 1/T_2^\star\right]^{-1}$.

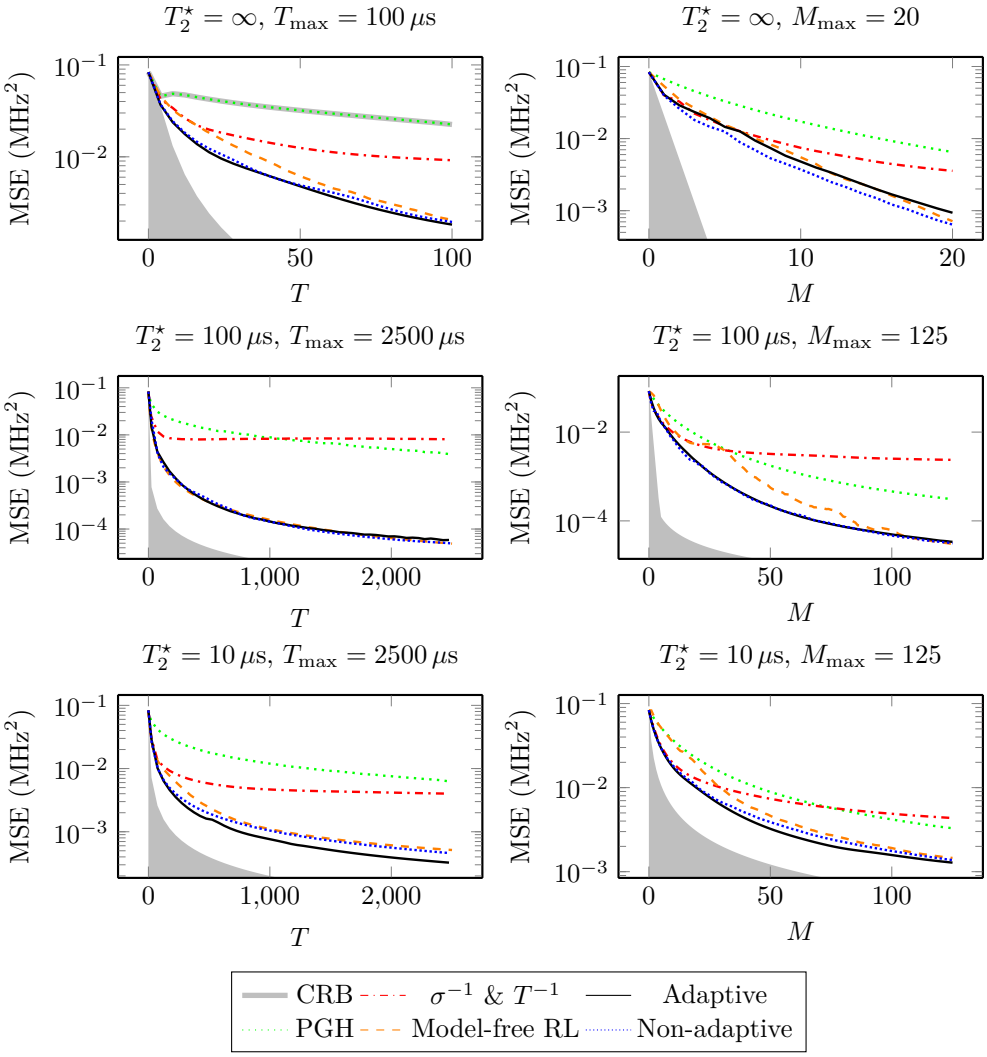


Figure 2: These plots refer to the static field magnetometry with an NV center, conducted under different conditions. The mean squared error (MSE) on ω is plotted as a function of the total number of consumed resources, which are either the maximum number of measurements $M_{\max}$ or the maximum total free evolution time of the probe, i.e., $T_{\max} \geq \sum_{t=0}^{M-1} \tau_k$. The adaptive and non-adaptive strategies are optimized using model-aware reinforcement learning. The formula used to compute the evolution time when employing the $\sigma^{-1}\&T^{-1}$ and Particle Guess Heuristic (PGH) strategies can be found in the main text. With the label "Model-free RL", we denote the performances obtained in [34] using model-free RL, which are never better than the non-adaptive strategy optimized with our techniques. The shaded grey area represents the (non-tight) ultimate precision bound, computed either from the Cramér-Rao bound (CRB) or from bit-counting arguments, as detailed in the Supplementary Information Appendix G.3. The title of each plot includes the transverse relaxation time $T_2^\star$, which is the time scale of the dephasing noise, along with the maximum amount of resources utilized in the simulations. The number of particles in the particle filter was $N = 480$ for the first and third rows, and $N = 1024$ (left) and $N = 1536$ (right) for the second row, which are significantly smaller numbers than those used in [34].

For each plot, the better performance between our optimized adaptive and non-adaptive strategies outperforms all the other approaches, as illustrated in Fig. 2. There are two comparisons to be made: on one hand, we have the optimized adaptive versus the non-adaptive strategies, which are both original results of this work; on the other hand, we have model-free versus model-aware reinforcement learning (RL), where the application of the latter to NV center magnetometry has been studied in [34]. We shall begin with the first comparison. Notably, the optimal results for extended coherence times ($T_2^\star = 100\,\mu\mathrm{s}, \infty$) are achieved using non-adaptive strategies, which offer several practical advantages in experimental implementation. Primarily, since the controls are fixed *offline* before the experiment, there is no requirement for real-time feedback via rapid electronics. Furthermore, there is also no need to update the Bayesian

posterior on the fly due to the absence of adaptivity. Instead, the measurement outcomes can be processed offline, post-measurement, using more powerful hardware. This would significantly reduce online memory usage as there is no need for real-time updates to the particle filter. In the third row of Fig. 2, we observe a gap between the performances of "Adaptive" and "Non-adaptive", and in [21] we provide additional examples of the usefulness of adaptivity for NV centers. Regarding the second comparison of model-aware and model-free RL, we observe that no strategy trained with model-free RL can surpass even the non-adaptive strategy, indicating that the results of [34], although similar to ours, cannot prove that the neural network has been trained to exploit adaptivity and that it has not simply learned an optimal non-adaptive sequence of measurement times τ. Moreover, we notice that our "Adaptive" strategy and the "Model-free" approach yield results that are closer toward the end of the estimation, while they differ for intermediate times. This is attributed to our use of cumulative loss. In the simulation with $T_2^\star = \infty$, $M_{\max} = 20$, the NN strategy performs worse than the non-adaptive one because it becomes stuck in a local minimum during training. In Fig. 3, we present five examples of optimal adaptive trajectories for the estimation of $\omega = 0.2\,\mathrm{MHz}$ corresponding to $T_2^\star = 10$, along with the optimal non-adaptive strategy. We observed also that multiple runs of the agent training will yield consistent performance but not necessarily the same optimized agent. In conclusion, we aim to provide an explanation for why adaptive control appears to offer limited advantage compared to the optimized non-adaptive strategy. For adaptivity to be beneficial, the phase $\omega\tau$ must be known to some extent. As the error on ω decreases, the evolution time increases, resulting in the uncertainty on $\omega\tau$ not approaching zero even after many measurements, which leaves very little room for adaptivity to enhance estimation precision. We also wish to point out that the Cramér-Rao (CR) bound (the gray area in the plot of Fig. 2) is not achievable without entanglement in the probes and measurements, and even then, the gap cannot be fully closed.

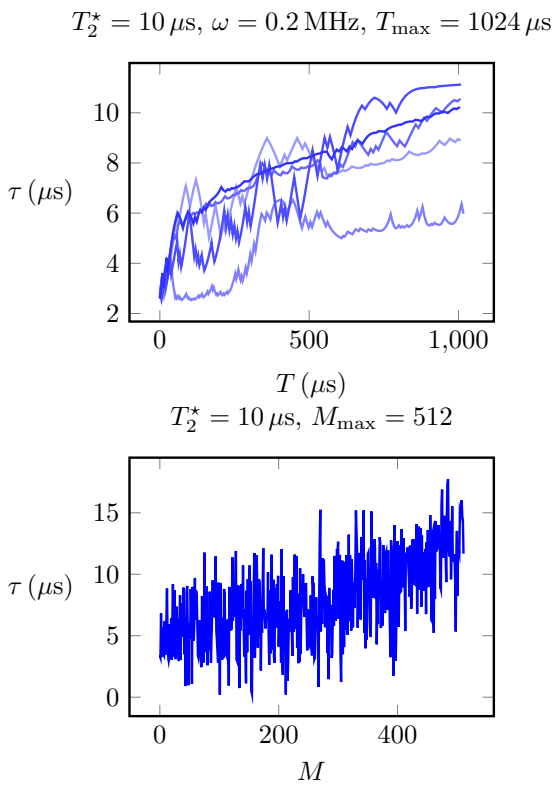


Figure 3: Control strategies for the estimation of ω in direct current (DC) magnetometry. For the time-limited estimation (in the upper panel), five instances of the trajectory of the control τ produced by the neural network (NN) are plotted for $\omega = 0.2\,\mathrm{MHz}$. The optimized strategies proposed by the NN start from the same point but gradually diverge over time as more data accumulates due to the stochastic nature of the measurement outcomes. The observed "diffusion" is attributed to the NN following the estimation adaptively. In the lower panel, the optimal non-adaptive strategy for the measurement-limited estimation is presented. This prescribes an increasing τ with a random pattern superimposed, which is interpreted as beneficial for compensating for the non-adaptivity of the strategy compared to the upper panel, with the multiple adaptive trajectories.

Agnostic Dolinar receiver

In this section, we address the challenge of distinguishing between two known coherent states, $|-\alpha\rangle$ and $|\alpha\rangle$, where $\alpha \in \mathbb{R}$ and $\alpha > 0$, using a single copy of the signal $|\pm\alpha\rangle$. The Dolinar receiver optimally addresses this problem through linear optics and photon counting [92, 93, 94, 95, 96, 97, 98]. For this device, multiple machine learning approaches can be found in the literature [99, 100]. In some recent studies [62, 101], a variant of this device was introduced that does not require a local oscillator (LO) on the receiver side, which must be in phase with the sender's laser. This is the agnostic Dolinar receiver, in which, instead of the LO, n copies of $|\alpha\rangle$, referred to as the reference states, are sent to the receiver from the sender, alongside the signal $|\pm\alpha\rangle$. We furthermore assume that classical knowledge about the state $|\alpha\rangle$ is absent, i.e., α is an unknown parameter of the estimation. In Fig. 4, we schematically represent this device, which leverages the states $|\alpha\rangle^{\otimes n}$ to perform the discrimination task on the sign of the signal $|\pm\alpha\rangle$. The signal $|\pm\alpha\rangle$ enters from the left and is sequentially combined with one of the reference states $|\alpha\rangle$ on a programmable beam splitter with adjustable reflectivity θ_i. At each beam splitter, one of the two ports undergoes measurement by a photon counter, while the residual signal $|\psi_i\rangle$ from the other port is fed forward to the subsequent beam splitter. The photon counting result is used to update the Bayesian posterior on α and on the signal's sign, from which the reflectivity

for the upcoming beam splitter is determined via a neural network. In this task, which combines estimation and discrimination, there are two undetermined parameters: one continuous, i.e., the signal's amplitude $\alpha \in \mathbb{R}$, and one discrete, i.e., the signal's sign. The receiver's performance is assessed based on the error probability in the task of signal classification, with the loss being defined in Eq. (8), while the amplitude α is a nuisance parameter. See [21] for details regarding the loss and the input to the NN.

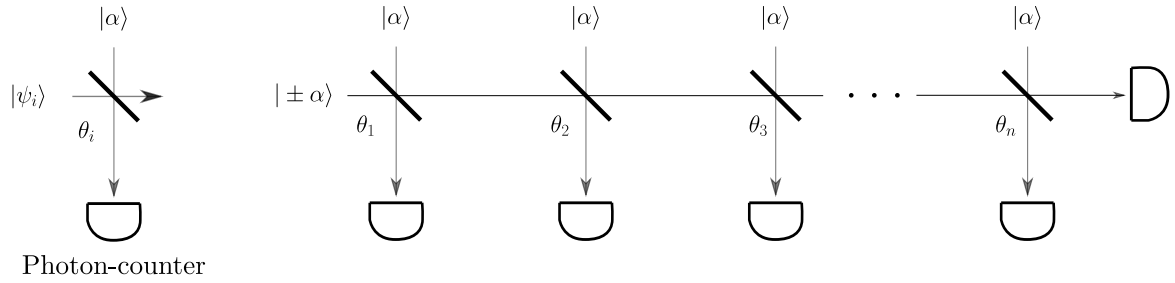


Figure 4: Schematic representation of the agnostic Dolinar receiver: each thick diagonal line symbolizes a beam splitter with programmable reflectivity θ_i. Each "D" device denotes a photon counter. On the left side of the figure, the fundamental building block of this apparatus is illustrated. Here, the input state at step i, denoted as $|\psi_i\rangle$, is combined with one of the n training states $|\alpha\rangle$. One of the two ports undergoes a measurement via photon counting. At the device's end, the second output port is also measured, ensuring that no information is left unused. It is important to note that the values of the control θ_i for the i-th measurement are determined based on the outcomes of all previous measurements.

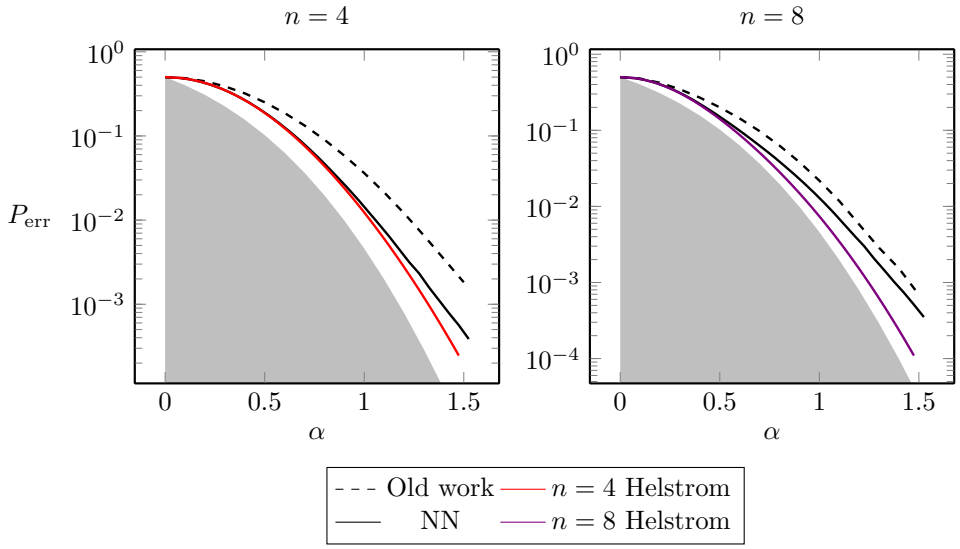


Figure 5: Comparison of error probabilities for various strategies with different numbers of copies of $|\alpha\rangle$, specifically $n = 4$ and $n = 8$. The shaded gray area represents the region excluded by the Helstrom bound [102, 103], which is the lowest error probability theoretically achievable when assuming access to an infinite number of reference states ($n = \infty$). The solid red and violet lines are the Helstrom bound calculated for a finite number of copies of $|\alpha\rangle$ [62], specifically $n = 4$ and $n = 8$. For details on the computation of the Helstrom bound, see [21]. The black dashed line showcases the lowest error found in the earlier work [62], without machine learning, while the black solid line represents the performance achieved using the neural network (NN). The performances of the optimal non-adaptive strategies have not been reported as they cannot rival those of the NN. For both the training and the performance evaluation, we used $N = 512$ particles. The weights and biases of the NN have been initialized randomly.

The simulation results are presented in Fig. 5, where we compared the performances of our adaptive procedure with the current state-of-the-art solution for this problem [62]. In each scenario, we achieved superior results with the neural network (NN). Notably, we nearly reached the theoretical bound in our primary area of interest, relevant for long-distance communications, which is the full quantum limit with $\alpha \lesssim 1$, and a small number of reference states (specifically, $n = 4$). For large α, the error probability is already very small, placing us in the classical limit. See Fig. 6 for an example of series of trajectories for the control of the beam splitter phase, as it is determined by the network. For completeness, we mention that we have chosen a prior on α that is uniform in the interval $[0.05, 1.50]$, and the sign is also uniform

in ± 1. Additionally, we report the trajectory for five executions of the state discrimination.

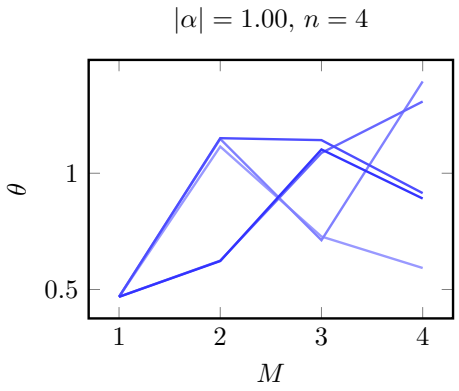


Figure 6: Examples of trajectories for the control sequence in the Dolinar receiver are presented. The plots illustrate the selected values for θ as a function of the measurement step M. The total number of measurements is $n = 4$, and the intensity of the signal is $|\alpha| = 1$. The different shades of blue represent the five distinct trajectories that are being considered.

Choice of the hyperparameters

In this section, we provide a brief commentary on the selection of hyperparameters for the examples presented. These parameters include the batch size B, the number of particles in the particle filter N, and the initial learning rate α_0. The selection of these parameters must align with the memory limitations of the computer during training. Specifically, we first empirically determine the number of particles to a value sufficient to ensure that the discretization of the posterior does not compromise the precision of the estimation. This choice subsequently influences the batch size, i.e., the number of simulations that can be executed in parallel. The batch size, in conjunction with the type of loss function, then determines the initial learning rate. For instance, in Fig. 2, we utilized $B = 128$ and $\alpha_0 = 10^{-2}$ for the cumulative loss, and $B = 1024$ and $\alpha_0 = 10^{-3}$ for the logarithmic loss. The batch size can also be increased through gradient accumulation, which involves averaging the gradients from multiple executions of a batch of simulations for the update in Eq. (12). For the Dolinar receiver, we employed $\alpha = 10^{-2}$ and $B = 4096$. Refer to Appendix E for further information.

4 Technical remarks

We emphasize that our technique for optimizing sensors is fundamentally based on two steps. First, a model for the system must be constructed; this model can be analytical if the physics of the system is well understood and characterized, or numerical if necessary. Subsequently, based on the constructed model, the optimal strategy for estimation can be derived through reinforcement learning, as demonstrated in this work. An online approach that involves learning the model concurrently with the training of the strategy exceeds the scope of our current achievements. In both examples presented, we provided the neural network with a summary of the information extracted via Bayesian estimation. A straightforward extension of this approach would involve supplying the network with higher moments of the posterior distribution, allowing the network to learn more intricate details of the optimal estimation strategy. In theory, it is feasible to input the entire posterior distribution into the NN in the form of all particles and their corresponding weights. However, this is rarely practical if the parameters are continuous, although this method is employed in those examples discussed in [29], which involve the estimation of discrete parameters only.

5 Conclusions

In this section, we summarize the results obtained from applying model-aware reinforcement learning to metrology and draw conclusions regarding the utility of this approach. Overall, our research underscores the advantages of integrating machine learning with modern quantum technologies. We have introduced a framework, complemented by a versatile library, capable of addressing a wide range of quantum parameter estimation and metrology challenges within both Bayesian and frequentist frameworks, applicable to various platforms. Our methods possess the potential to expedite the development of practical applications in quantum metrology. The ability to accurately estimated physical parameters through quantum systems could revolutionize multiple sectors, including biology, fundamental physics, and quantum communication. Through the application of model-aware reinforcement learning, we aim to facilitate progress in these domains, easing the transition of quantum-based metrology from proof-of-principle experiments to industrial applications. This work aims to accelerate the search for optimal control strategies in quantum sensors, potentially speeding up their widespread industrial adoption. The technique of model-aware RL for agent optimization can, in principle, be applied to a broad spectrum of problems in quantum information, including quantum error correction and entanglement distillation, though this would necessitate engineering different loss functions. The primary challenge in extending this approach to other fields of quantum information beyond metrology is the rapidly increasing dimensionality of the quantum state spaces that would need to be simulated.

6 Methods

The library qsensoropt has been implemented in Python 3 on the Tensorflow framework. All of the simulations have been done on the High-Performance Computing cluster of Scuola Normale Superiore. The simulations ran on an NVIDIA Tesla GPU with 32GB of dedicated VRAM. The training and evaluation of each strategy took $\mathcal{O}(1)$ hours.

7 Data availability

All the data referenced in this work can be found in the GitLab repository, in the folders `qsensoropt/examples/nv_center_dc` and `qsensoropt/examples/dolinar`. The repository can be found at the URL https://gitlab.com/federico.belliardo/qsensoropt.

8 Code availability

The open source library qsensoropt can be found in the GitLab repository and can be installed with `pip` following the instruction in the README file. The repository can be found at the URL https://gitlab.com/federico.belliardo/qsensoropt. The package is also available on PyPI at https://pypi.org/project/qsensoropt/.

9 Acknowledgments

We gratefully acknowledge the computational resources of the Center for High Performance Computing (CHPC) at SNS. F. B. thanks T. Shah for useful discussions. We aknowledge finantial support by MUR (Ministero dell'Istruzione, dell'Università e della Ricerca) through the following projects: PNRR MUR project PE0000023-NQSTI, PRIN 2017 Taming complexity via Quantum Strategies: a Hybrid Integrated Photonic approach (QU-SHIP) Id. 2017SRNBRK.

10 Author Contributions

F. Belliardo conceived of the presented idea under the supervision of F. Marquardt and V. Giovannetti. Federico B. and Fabio Z. have programmed the library and performed the simulations, all the authors have discussed the results and contributed to the final manuscript, with F. Belliardo being the main contributor.

11 Competing interests statement

The authors declare no competing interests.

12 References

[1] Fulvio Flamini, Arne Hamann, Sofiène Jerbi, Lea M Trenkwalder, Hendrik Poulsen Nautrup, and Hans J Briegel. "Photonic architecture for reinforcement learning". New Journal of Physics **22**, 045002 (2020).

[2] Michael Broughton, Guillaume Verdon, Trevor McCourt, Antonio J. Martinez, Jae Hyeon Yoo, Sergei V. Isakov, Philip Massey, Ramin Halavati, Murphy Yuezhen Niu, Alexander Zlokapa, Evan Peters, Owen Lockwood, Andrea Skolik, Sofiene Jerbi, Vedran Dunjko, Martin Leib, Michael Streif, David Von Dollen, Hongxiang Chen, Shuxiang Cao, Roeland Wiersema, Hsin-Yuan Huang, Jarrod R. McClean, Ryan Babbush, Sergio Boixo, Dave Bacon, Alan K. Ho, Hartmut Neven, and Masoud Mohseni. "TensorFlow Quantum: A Software Framework for Quantum Machine Learning" (2021).

[3] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Shahnawaz Ahmed, Vishnu Ajith, M. Sohaib Alam, Guillermo Alonso-Linaje, B. AkashNarayanan, Ali Asadi, Juan Miguel Arrazola, Utkarsh Azad, Sam Banning, Carsten Blank, Thomas R. Bromley, Benjamin A. Cordier, Jack Ceroni, Alain Delgado, Olivia Di Matteo, Amintor Dusko, Tanya Garg, Diego Guala, Anthony Hayes, Ryan Hill, Aroosa Ijaz, Theodor Isacsson, David Ittah, Soran Jahangiri, Prateek Jain, Edward Jiang, Ankit Khandelwal, Korbinian Kottmann, Robert A. Lang, Christina Lee, Thomas Loke, Angus Lowe, Keri McKiernan, Johannes Jakob Meyer, J. A. Montañez-Barrera, Romain Moyard, Zeyue Niu, Lee James O'Riordan, Steven Oud, Ashish Panigrahi, Chae-Yeun Park, Daniel Polatajko, Nicolás Quesada, Chase Roberts, Nahum Sá, Isidor Schoch, Borun Shi, Shuli Shu, Sukin Sim, Arshpreet Singh, Ingrid Strandberg, Jay Soni, Antal Száva, Slimane Thabet, Rodrigo A. Vargas-Hernández, Trevor Vincent, Nicola Vitucci, Maurice Weber, David Wierichs, Roeland Wiersema, Moritz Willmann, Vincent Wong, Shaoming Zhang, and Nathan Killoran. "PennyLane: Automatic differentiation of hybrid quantum-classical computations" (2022).

[4] Marin Bukov, Alexandre G. R. Day, Dries Sels, Phillip Weinberg, Anatoli Polkovnikov, and Pankaj Mehta. "Reinforcement Learning in Different Phases of Quantum Control". Physical Review X **8**, 031086 (2018).

[5] Xiao-Ming Zhang, Zezhu Wei, Raza Asad, Xu-Chen Yang, and Xin Wang. "When does reinforcement learning stand out in quantum con-

trol? A comparative study on state preparation". npj Quantum Information **5**, 85 (2019).

[6] Murphy Yuezhen Niu, Sergio Boixo, Vadim N. Smelyanskiy, and Hartmut Neven. "Universal quantum control through deep reinforcement learning". npj Quantum Information **5**, 33 (2019).

[7] Riccardo Porotti, Antoine Essig, Benjamin Huard, and Florian Marquardt. "Deep Reinforcement Learning for Quantum State Preparation with Weak Nonlinear Measurements". Quantum **6**, 747 (2022).

[8] Riccardo Porotti, Vittorio Peano, and Florian Marquardt. "Gradient-Ascent Pulse Engineering with Feedback". PRX Quantum **4**, 030305 (2023).

[9] Thomas Fösel, Petru Tighineanu, Talitha Weiss, and Florian Marquardt. "Reinforcement Learning with Neural Networks for Quantum Feedback". Physical Review X **8**, 031084 (2018).

[10] Valeria Cimini, Ilaria Gianani, Nicolò Spagnolo, Fabio Leccese, Fabio Sciarrino, and Marco Barbieri. "Calibration of Quantum Sensors by Neural Networks". Physical Review Letters **123**, 230502 (2019).

[11] Yue Ban, Javier Echanobe, Yongcheng Ding, Ricardo Puebla, and Jorge Casanova. "Neural-network-based parameter estimation for quantum detection". Quantum Science and Technology **6**, 045012 (2021).

[12] Samuel Nolan, Augusto Smerzi, and Luca Pezzè. "A machine learning approach to Bayesian parameter estimation". npj Quantum Information **7**, 169 (2021).

[13] Samuel P. Nolan, Luca Pezzè, and Augusto Smerzi. "Frequentist parameter estimation with supervised learning". AVS Quantum Science **3**, 034401 (2021).

[14] V. Nguyen, S. B. Orbell, D. T. Lennon, H. Moon, F. Vigneau, L. C. Camenzind, L. Yu, D. M. Zumbühl, G. A. D. Briggs, M. A. Osborne, D. Sejdinovic, and N. Ares. "Deep reinforcement learning for efficient measurement of quantum devices". npj Quantum Information **7**, 100 (2021).

[15] Adriano Macarone Palmieri, Egor Kovlakov, Federico Bianchi, Dmitry Yudin, Stanislav Straupe, Jacob D. Biamonte, and Sergei Kulik. "Experimental neural network enhanced quantum tomography". npj Quantum Information **6**, 20 (2020).

[16] Yihui Quek, Stanislav Fort, and Hui Khoon Ng. "Adaptive quantum state tomography with neural networks". npj Quantum Information **7**, 105 (2021).

[17] Hsien-Yi Hsieh, Jingyu Ning, Yi-Ru Chen, Hsun-Chung Wu, Hua Li Chen, Chien-Ming Wu, and Ray-Kuang Lee. "Direct Parameter Estimations from Machine Learning-Enhanced Quantum State Tomography". Symmetry **14**, 874 (2022).

[18] Florian Marquardt. "Machine learning and quantum devices". SciPost Physics Lecture NotesPage 29 (2021).

[19] Florian Marquardt. "Online Course: Advanced Machine Learning for Physics, Science, and Artificial Scientific Discovery" (2021).

[20] Mario Krenn, Jonas Landgraf, Thomas Foesel, and Florian Marquardt. "Artificial intelligence and machine learning for quantum technologies". Physical Review A **107**, 010101 (2023).

[21] Federico Belliardo, Fabio Zoratti, and Vittorio Giovannetti. "Applications of model-aware reinforcement learning in Bayesian quantum metrology" (2024). arXiv:2403.05706 [quant-ph].

[22] R. A. Fisher. "The design of experiments". The design of experiments. Oliver & Boyd. Oxford, England (1935).

[23] A. E. Foster. "Variational, Monte Carlo and policy-based approaches to Bayesian experimental design". http://purl.org/dc/dcmitype/Text. University of Oxford. (2021). url: https://ora.ox.ac.uk/objects/uuid:4a3e13ca-e6c6-4669-955e-f1a87e201228.

[24] Atılım Güneş Baydin, Kyle Cranmer, Pablo de Castro Manzano, Christophe Delaere, Denis Derkach, Julien Donini, Tommaso Dorigo, Andrea Giammanco, Jan Kieseler, Lukas Layer, Gilles Louppe, Fedor Ratnikov, Giles C. Strong, Mia Tosi, Andrey Ustyuzhanin, Pietro Vischia, and Hevjin Yarar. "Toward Machine Learning Optimization of Experimental Design". Nuclear Physics News **31**, 25–28 (2021).

[25] Zachary Ballard, Calvin Brown, Asad M. Madni, and Aydogan Ozcan. "Machine learning and computation-enabled intelligent sensor design". Nature Machine Intelligence **3**, 556–565 (2021).

[26] Desi R Ivanova, Adam Foster, Steven Kleinegesse, Michael U. Gutmann, and Thomas Rainforth. "Implicit Deep Adaptive Design: Policy-Based Experimental Design without Likelihoods". In Advances in

Neural Information Processing Systems. Volume 34, pages 25785–25798. Curran Associates, Inc. (2021). url: https://proceedings.neurips.cc/paper/2021/hash/d811406316b669ad3d370d78b51b1d2e-Abstract.html.

[27] Leopoldo Sarra and Florian Marquardt. "Deep Bayesian Experimental Design for Quantum Many-Body Systems" (2023).

[28] Adam Foster, Desi R. Ivanova, Ilyas Malik, and Tom Rainforth. "Deep Adaptive Design: Amortizing Sequential Bayesian Experimental Design". In Proceedings of the 38th International Conference on Machine Learning. Pages 3384–3395. PMLR (2021). url: https://proceedings.mlr.press/v139/foster21a.html.

[29] Federico Belliardo, Fabio Zoratti, and Vittorio Giovannetti. "Applications of model-aware reinforcement learning in bayesian quantum metrology". Phys. Rev. A **109**, 062609 (2024).

[30] Valeria Cimini, Mauro Valeri, Emanuele Polino, Simone Piacentini, Francesco Ceccarelli, Giacomo Corrielli, Nicolò Spagnolo, Roberto Osellame, and Fabio Sciarrino. "Deep reinforcement learning for quantum multiparameter estimation". Advanced Photonics **5**, 016005 (2023).

[31] Alessio Fallani, Matteo A. C. Rossi, Dario Tamascelli, and Marco G. Genoni. "Learning Feedback Control Strategies for Quantum Metrology". PRX Quantum **3**, 020310 (2022).

[32] Han Xu, Lingna Wang, Haidong Yuan, and Xin Wang. "Generalizable control for multiparameter quantum metrology". Physical Review A **103**, 042615 (2021).

[33] Tailong Xiao, Jianping Fan, and Guihua Zeng. "Parameter estimation in quantum sensing based on deep reinforcement learning". npj Quantum Information **8**, 1–12 (2022).

[34] Lukas J. Fiderer, Jonas Schuff, and Daniel Braun. "Neural-Network Heuristics for Adaptive Bayesian Quantum Estimation". PRX Quantum **2**, 020303 (2021).

[35] Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. "A unifying framework for reinforcement learning and planning". Frontiers in Artificial Intelligence**5** (2022).

[36] Johannes Jakob Meyer, Johannes Borregaard, and Jens Eisert. "A variational toolbox for quantum multi-parameter estimation". npj Quantum Information **7**, 1–5 (2021).

[37] Mao Zhang, Huai-Ming Yu, Haidong Yuan, Xiaoguang Wang, Rafał Demkowicz-Dobrzański, and Jing Liu. "QuanEstimation: An open-source toolkit for quantum parameter estimation". Physical Review Research **4**, 043057 (2022).

[38] Christopher Granade, Christopher Ferrie, Ian Hincks, Steven Casagrande, Thomas Alexander, Jonathan Gross, Michal Kononenko, and Yuval Sanders. "QInfer: Statistical inference software for quantum applications". Quantum **1**, 5 (2017).

[39] Robert D. McMichael, Sean M. Blakley, and Sergey Dushenko. "Optbayesexpt: Sequential Bayesian Experiment Design for Adaptive Measurements". Journal of Research of the National Institute of Standards and Technology **126**, 126002 (2021).

[40] Jessica Bavaresco, Patryk Lipka-Bartosik, Pavel Sekatski, and Mohammad Mehboudi. "Designing optimal protocols in Bayesian quantum parameter estimation with higher-order operations" (2023).

[41] Jing Liu and Haidong Yuan. "Quantum parameter estimation with optimal control". Physical Review A **96**, 012117 (2017).

[42] Han Xu, Junning Li, Liqiang Liu, Yu Wang, Haidong Yuan, and Xin Wang. "Generalizable control for quantum parameter estimation through reinforcement learning". npj Quantum Information **5**, 1–8 (2019).

[43] Phila Rembold, Nimba Oshnik, Matthias M. Müller, Simone Montangero, Tommaso Calarco, and Elke Neu. "Introduction to quantum optimal control for quantum sensing with nitrogen-vacancy centers in diamond". AVS Quantum Science **2**, 024701 (2020).

[44] Jonas Schuff, Lukas J Fiderer, and Daniel Braun. "Improving the dynamics of quantum sensors with reinforcement learning". New Journal of Physics **22**, 035001 (2020).

[45] Jing Liu, Mao Zhang, Hongzhen Chen, Lingna Wang, and Haidong Yuan. "Optimal Scheme for Quantum Metrology". Advanced Quantum Technologies **5**, 2100080 (2022).

[46] Yuxiang Qiu, Min Zhuang, Jiahao Huang, and Chaohong Lee. "Efficient and robust entanglement generation with deep reinforcement learning for quantum metrology". New Journal of Physics **24**, 083011 (2022).

[47] Seyed Shakib Vedaie, Archismita Dalal, Eduardo J. Páez, and Barry C. Sanders. "Framework for Learning and Control in the Classical and Quantum Domains" (2023).

[48] Valentin Gebhart, Raffaele Santagati, Antonio Andrea Gentile, Erik M. Gauger, David Craig, Natalia Ares, Leonardo Banchi, Florian Marquardt, Luca Pezzè, and Cristian Bonato. "Learning quantum systems". Nature Reviews Physics **5**, 141–156 (2023).

[49] Ziqi Ma, Pranav Gokhale, Tian-Xing Zheng, Sisi Zhou, Xiaofei Yu, Liang Jiang, Peter Maurer, and Frederic T. Chong. "Adaptive Circuit Learning for Quantum Metrology". In 2021 IEEE International Conference on Quantum Computing and Engineering (QCE). Pages 419–430. (2021).

[50] Chengyin Han, Zhu Ma, Yuxiang Qiu, Ruihuan Fang, Jiatao Wu, Chang Zhan, Maojie Li, Jiahao Huang, Bo Lu, and Chaohong Lee. "Atomic clock locking with bayesian quantum parameter estimation: Scheme and experiment". Phys. Rev. Appl. **22**, 044058 (2024).

[51] Raphael Kaubruegger, Denis V. Vasilyev, Marius Schulte, Klemens Hammerer, and Peter Zoller. "Quantum Variational Optimization of Ramsey Interferometry and Atomic Clocks". Physical Review X **11**, 041045 (2021).

[52] Christian D. Marciniak, Thomas Feldker, Ivan Pogorelov, Raphael Kaubruegger, Denis V. Vasilyev, Rick van Bijnen, Philipp Schindler, Peter Zoller, Rainer Blatt, and Thomas Monz. "Optimal metrology with programmable quantum sensors". Nature **603**, 604–609 (2022).

[53] Raphael Kaubruegger, Athreya Shankar, Denis V. Vasilyev, and Peter Zoller. "Optimal and variational multiparameter quantum metrology and vector-field sensing". PRX Quantum **4**, 020333 (2023).

[54] Lukas Gerster, Fernando Martínez-García, Pavel Hrmo, Martin W. van Mourik, Benjamin Wilhelm, Davide Vodola, Markus Müller, Rainer Blatt, Philipp Schindler, and Thomas Monz. "Experimental bayesian calibration of trapped-ion entangling operations". PRX Quantum **3**, 020350 (2022).

[55] Alec Cao, William J. Eckner, Theodor Lukin Yelin, Aaron W. Young, Sven Jandura, Lingfeng Yan, Kyungtae Kim, Guido Pupillo, Jun Ye, Nelson Darkwah Oppong, and Adam M. Kaufman. "Multi-qubit gates and schrödinger cat states in an optical clock". Nature **634**, 315–320 (2024).

[56] Emre Köse and Daniel Braun. "Superresolution imaging with multiparameter quantum metrology in passive remote sensing". Physical Review A **107**, 032607 (2023).

[57] A. Muñoz de las Heras, C. Tabares, J. T. Schneider, L. Tagliacozzo, D. Porras, and A. González-Tudela. "Photonic quantum metrology with variational quantum optical non-linearities" (2023).

[58] Jing Yang, Shengshi Pang, Zekai Chen, Andrew N. Jordan, and Adolfo del Campo. "Variational principle for optimal quantum controls in quantum metrology". Phys. Rev. Lett. **128**, 160505 (2022).

[59] Ming Chen, Chao Meng, Qi Zhang, Changkui Duan, Fazhan Shi, and Jiangfeng Du. "Quantum metrology with single spins in diamond under ambient conditions". National Science Review **5**, 346–355 (2018).

[60] Muhammad Junaid Arshad, Christiaan Bekker, Ben Haylock, Krzysztof Skrzypczak, Daniel White, Benjamin Griffiths, Joe Gore, Gavin W. Morley, Patrick Salter, Jason Smith, Inbar Zohar, Amit Finkler, Yoann Altmann, Erik M. Gauger, and Cristian Bonato. "Real-time adaptive estimation of decoherence timescales for a single qubit". Phys. Rev. Appl. **21**, 024026 (2024).

[61] Timo Joas, Simon Schmitt, Raffaele Santagati, Antonio Andrea Gentile, Cristian Bonato, Anthony Laing, Liam P. McGuinness, and Fedor Jelezko. "Online adaptive quantum characterization of a nuclear spin". npj Quantum Information **7**, 1–8 (2021).

[62] Fabio Zoratti, Nicola Dalla Pozza, Marco Fanizza, and Vittorio Giovannetti. "An agnostic-Dolinar receiver for coherent states classification". Physical Review A **104**, 042606 (2021).

[63] Pierre Del Moral. "Nonlinear filtering: Interacting particle resolution". Comptes Rendus de l'Académie des Sciences - Series I - Mathematics **325**, 653–658 (1997).

[64] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking". IEEE Transactions on Signal Processing **50**, 174–188 (2002).

[65] Jun S. Liu and Rong Chen. "Sequential Monte Carlo Methods for Dynamic Systems". Journal of the American Statistical Association **93**, 1032–1044 (1998).

[66] Tim De Ryck, Samuel Lanthaler, and Siddhartha Mishra. "On the approximation of functions by tanh neural networks". Neural Networks **143**, 732–750 (2021).

[67] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In Yoshua Bengio and Yann LeCun, editors, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. (2015). url: http://arxiv.org/abs/1412.6980.

[68] Peter Karkus, David Hsu, and Wee Sun Lee. "Particle Filter Networks with Application to Visual Localization". In Proceedings of The 2nd Conference on Robot Learning. Pages 169–178. PMLR (2018). url: https://proceedings.mlr.press/v87/karkus18a.html.

[69] Adam Ścibior and Frank Wood. "Differentiable Particle Filtering without Modifying the Forward Pass" (2021).

[70] Adam Gali. "Ab initio theory of the nitrogen-vacancy center in diamond". Nanophotonics **8**, 1907–1943 (2019).

[71] Marcus W. Doherty, Chunhui Rita Du, and Gregory D. Fuchs. "Quantum science and technology based on color centers with accessible spin". Journal of Applied Physics **131**, 010401 (2022).

[72] Jeronimo Maze. "Quantum manipulation of nitrogen-vacancy centers in diamond: From basic properties to applications". PhD thesis. Harvard University. (2010).

[73] John F Barry, Jennifer M Schloss, Erik Bauch, Matthew J Turner, Connor A Hart, Linh M Pham, and Ronald L Walsworth. "Sensitivity optimization for NV-diamond magnetometry". Rev. Mod. Phys. **92**, 68 (2020).

[74] Simon Schmitt, Tuvia Gefen, Daniel Louzon, Christian Osterkamp, Nicolas Staudenmaier, Johannes Lang, Matthew Markham, Alex Retzker, Liam P. McGuinness, and Fedor Jelezko. "Optimal frequency measurements with quantum probes". npj Quantum Information **7**, 55 (2021).

[75] Christopher Ferrie, Christopher E. Granade, and D. G. Cory. "How to best sample a periodic probability distribution, or on the accuracy of Hamiltonian finding strategies". Quantum Information Processing **12**, 611–623 (2013).

[76] Sergey Dushenko, Kapildeb Ambal, and Robert D. McMichael. "Sequential Bayesian Experiment Design for Optically Detected Magnetic Resonance of Nitrogen-Vacancy Centers". Physical Review Applied **14**, 054036 (2020).

[77] Robert D. McMichael, Sergey Dushenko, and Sean M. Blakley. "Sequential Bayesian experiment design for adaptive Ramsey sequence measurements". Journal of Applied Physics **130**, 144401 (2021).

[78] Christopher E. Granade, Christopher Ferrie, Nathan Wiebe, and D. G. Cory. "Robust online Hamiltonian learning". New Journal of Physics **14**, 103013 (2012).

[79] Nimba Oshnik, Phila Rembold, Tommaso Calarco, Simone Montangero, Elke Neu, and Matthias M. Müller. "Robust magnetometry with single nitrogen-vacancy centers via two-step optimization". Physical Review A **106**, 013107 (2022).

[80] K Craigie, E M Gauger, Y Altmann, and C Bonato. "Resource-efficient adaptive Bayesian tracking of magnetic fields with a quantum sensor". Journal of Physics: Condensed Matter **33**, 195801 (2021).

[81] C. Bonato, M. S. Blok, H. T. Dinani, D. W. Berry, M. L. Markham, D. J. Twitchen, and R. Hanson. "Optimized quantum sensing with a single electron spin using real-time adaptive measurements". Nature Nanotechnology **11**, 247–252 (2016).

[82] 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". Physical Review X **9**, 021019 (2019).

[83] I Zohar, B Haylock, Y Romach, M J Arshad, N Halay, N Drucker, R Stöhr, A Denisenko, Y Cohen, C Bonato, and A Finkler. "Real-time frequency estimation of a qubit without single-shot-readout". Quantum Science and Technology **8**, 035017 (2023).

[84] N. M. Nusran, M. Ummal Momeen, and M. V. Gurudev Dutt. "High-dynamic-range magnetometry with a single electronic spin in diamond". Nature Nanotechnology **7**, 109–113 (2012).

[85] Jianwei Wang, Stefano Paesani, Raffaele Santagati, Sebastian Knauer, Antonio A. Gentile, Nathan Wiebe, Maurangelo Petruzzella, Jeremy L. O'Brien, John G. Rarity, Anthony Laing, and Mark G. Thompson. "Experimental quantum Hamiltonian learning". Nature Physics **13**, 551–555 (2017).

[86] Hossein T. Dinani, Dominic W. Berry, Raul Gonzalez, Jeronimo R. Maze, and Cristian Bonato. "Bayesian estimation for quantum sensing in the absence of single-shot detection". Physical Review B **99**, 125413 (2019).

[87] Cristian Bonato and Dominic W. Berry. "Adaptive tracking of a time-varying field with a quantum sensor". Physical Review A **95**, 052348 (2017).

[88] Christopher Ferrie, Christopher E. Granade, and D. G. Cory. "Adaptive Hamiltonian estimation using Bayesian experimental design". AIP Conference Proceedings **1443**, 165–173 (2012).

[89] Genyue Liu, Mo Chen, Yi-Xiang Liu, David Layden, and Paola Cappellaro. "Repetitive readout enhanced by machine learning". Machine Learning: Science and Technology **1**, 015003 (2020).

[90] Moeta Tsukamoto, Shuji Ito, Kensuke Ogawa, Yuto Ashida, Kento Sasaki, and Kensuke Kobayashi. "Machine-learning-enhanced quantum sensors for accurate magnetic field imaging". Scientific Reports **12**, 13942 (2022).

[91] Nathan Wiebe, Christopher Granade, Christopher Ferrie, and D. G. Cory. "Hamiltonian Learning and Certification Using Quantum Resources". Physical Review Letters **112**, 190501 (2014).

[92] Jr. Dolinar, S. J. "Processing and transmission of information". Massachusetts Institute of Technology. Research Laboratory of Electronics. Quarterly Progress Report, no. 111 (1973).

[93] JM Geremia. "Distinguishing between optical coherent states with imperfect detection". Physical Review A **70**, 062303 (2004).

[94] Shuro Izumi, Masahiro Takeoka, Mikio Fujiwara, Nicola Dalla Pozza, Antonio Assalini, Kazuhiro Ema, and Masahide Sasaki. "Displacement receiver for phase-shift-keyed coherent states". Physical Review A **86**, 042328 (2012).

[95] Antonio Assalini, Nicola Dalla Pozza, and Gianfranco Pierobon. "Revisiting the Dolinar receiver through multiple-copy state discrimination theory". Physical Review A **84**, 022342 (2011).

[96] Robert L. Cook, Paul J. Martin, and J. M. Geremia. "Optical coherent state discrimination using a closed-loop quantum measurement". Nature **446**, 774–777 (2007).

[97] Nicola Dalla Pozza and Nicola Laurenti. "Adaptive discrimination scheme for quantum pulse-position-modulation signals". Physical Review A **89**, 012339 (2014).

[98] Masahiro Takeoka, Masahide Sasaki, Peter van Loock, and Norbert Lütkenhaus. "Implementation of projective measurements with linear optics and continuous photon counting". Physical Review A **71**, 022318 (2005).

[99] M. Bilkis, M. Rosati, R. Morral Yepes, and J. Calsamiglia. "Real-time calibration of coherent-state receivers: Learning by trial and error". Physical Review Research **2**, 033295 (2020).

[100] Chaohan Cui, William Horrocks, Shuhong Hao, Saikat Guha, Nasser Peyghambarian, Quntao Zhuang, and Zheshen Zhang. "Quantum receiver enhanced by adaptive learning". Light: Science & Applications **11**, 344 (2022).

[101] Gael Sentís, Mădălin Guţă, and Gerardo Adesso. "Quantum learning of coherent states". EPJ Quantum Technology **2**, 17 (2015).

[102] Carl W. Helstrom. "Quantum detection and estimation theory". Journal of Statistical Physics **1**, 231–252 (1969).

[103] A. S. Holevo. "Statistical problems in quantum physics". In G. Maruyama and Yu. V. Prokhorov, editors, Proceedings of the Second Japan-USSR Symposium on Probability Theory. Pages 104–119. Lecture Notes in Mathematics-Berlin, Heidelberg (1973). Springer.

[104] Masahito Hayashi. "Asymptotic Theory of Quantum Statistical Inference". WORLD SCIENTIFIC. (2005).

[105] Antonio A. Gentile, Brian Flynn, Sebastian Knauer, Nathan Wiebe, Stefano Paesani, Christopher E. Granade, John G. Rarity, Raffaele Santagati, and Anthony Laing. "Learning models of quantum systems from experiments". Nature Physics **17**, 837–843 (2021).

[106] Tiancheng Li, Miodrag Bolic, and Petar M. Djuric. "Resampling Methods for Particle Filtering: Classification, implementation, and strategies". IEEE Signal Processing Magazine **32**, 70–86 (2015).

[107] Michael Zhu, Kevin Murphy, and Rico Jonschkowski. "Towards Differentiable Resampling" (2020).

[108] Xiao Ma, Peter Karkus, and David Hsu. "Particle Filter Recurrent Neural Networks". Proceedings of the AAAI Conference on Artificial Intelligence **34**, 5101–5108 (2020).

[109] Alexander S. Holevo. "Probabilistic and Statistical Aspects of Quantum Theory". Monographs (Scuola Normale Superiore). Edizioni della Normale. (2011).

[110] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In Advances in Neural Information Processing Systems. Volume 12. MIT Press (1999). url: https://papers.nips.cc/paper/1999/hash/464d828b85b0bed98e80ade0a5c43b0f-Abstract.html.

[111] Gregory Farquhar, Shimon Whiteson, and Jakob Foerster. "Loaded DiCE: Trading off Bias and Variance in Any-Order Score Function Gradient Estimators for Reinforcement Learning". In Advances in Neural Information Processing Systems. Volume 32. Curran Associates, Inc. (2019). url: https://proceedings.neurips.cc/paper/2019/hash/6fd6b030c6afec018415662d0db43f9d-Abstract.html.

[112] Lex Weaver and Nigel Tao. "The optimal reward baseline for gradient-based reinforcement learning". In Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence. Pages 538–545. UAI'01San Francisco, CA, USA (2001). Morgan Kaufmann Publishers Inc.

[113] Valeria Cimini, Emanuele Polino, Federico Belliardo, Francesco Hoch, Bruno Piccirillo, Nicolò Spagnolo, Vittorio Giovannetti, and Fabio Sciarrino. "Experimental metrology beyond the standard quantum limit for a wide resources range". npj Quantum Information **9**, 1–9 (2023).

A Schematization of physical systems in qsensoropt

Encoding of the probe

Following the standard nomenclature in quantum metrology, we define a quantum *probe* as a quantum system initialized in a reference state ρ. This probe encodes the d-dimensional vector of parameters $\boldsymbol{\theta} \in \Theta$ of interest, undergoing a controllable evolution determined by the controls x, i.e., $\rho \to \rho_{x,\boldsymbol{\theta}} = \mathcal{E}_{x,\boldsymbol{\theta}}(\rho)$, where $\mathcal{E}_{x,\boldsymbol{\theta}}$ is a general linear completely positive trace-preserving (LCPT) map. A control is a tunable parameter that can be adjusted during the experiment, which may include factors such as measurement duration, laser frequency detuning in a cavity, or a tunable phase in an interferometer. In a figurative sense, control parameters encompass all the controls on the experimental electronics. A control can be continuous if it takes values within an interval, or discrete if it assumes only a finite set of values (e.g., on and off). The encoded parameters $\boldsymbol{\theta}$ may represent properties of the environment, such as a magnetic field acting on a spin in the NV center platform, or certain degrees of freedom of the probe's initial state, such as the parameter α of a coherent state of light $|\alpha\rangle$ in the agnostic Dolinar receiver.

This scheme can also be framed as a *communication protocol*, wherein Alice transmits the state $\rho_{\boldsymbol{\theta}}$ to Bob, who is tasked with decoding the vector $\boldsymbol{\theta}$. For the sake of generality, we will continue to refer to the quantum system as a probe in these contexts. The objective is to perform a measurement on $\rho_{x,\boldsymbol{\theta}}$ to extract information regarding $\boldsymbol{\theta}$. It is essential to distinguish that the term quantum parameter estimation in the literature pertains to situations where the encoded probe $\rho_{\boldsymbol{\theta}}$ is provided, meaning we start from this state and cannot act on the encoding. In contrast, quantum metrology provides access to the encoding process $\mathcal{E}_{x,\boldsymbol{\theta}}$, not merely to the final result. The implicit assumption in parameter estimation [104] is that the encoding occurs externally to the observed framework.

In both metrology and parameter estimation, we assume that the encoding $\mathcal{E}_{x,\boldsymbol{\theta}}$ is applied multiple times or that we are given numerous copies of $\rho_{\boldsymbol{\theta}}$, allowing us to gather statistically relevant data by measuring all copies from which we infer the value of $\boldsymbol{\theta}$. Quantum metrology represents a more general setting than parameter estimation, and since the techniques developed here apply to quantum metrology, they are also applicable to parameter estimation. An example of parameter estimation is receiving radiation emitted by a distribution of currents on a plane, which depends on the properties of the source, such as temperature [56]. In this scenario, the quantum probe is the radiation. As the radiation is assumed to be emitted from a distant and inaccessible region, we lack direct access to the quantum channel performing the encoding, receiving only the encoded states, which is the state of the radiated field upon detection.

An example of a quantum metrological task is the estimation of an environmental magnetic field using a spin, where we can select the initial state and the interaction duration. Parameters can be continuous or discrete. Continuous parameters include the magnetic field and temperature, while examples of discrete parameters encompass signal sign and the type of interaction between two quantum systems [105]. When discrete parameters are involved, we operate within the realm of value discrimination. In a metrological task, we may encounter a mixture of continuous and discrete parameters, as observed in the agnostic Dolinar receiver in Section 3. A parameter can also be classified as a nuisance parameter, which is an unknown variable that requires estimation, though we do not evaluate the precision of the procedure with respect to it, as we are not

directly interested in its value. An example of a nuisance parameter is the fluctuating optical visibility of an interferometer when our primary interest lies in the phase. Estimating nuisance parameters is often necessary and beneficial for estimating parameters of interest.

Measurement on the probe

To obtain information on $\boldsymbol{\theta}$, it is necessary to perform a measurement on the encoded probe $\rho_{x,\boldsymbol{\theta}}$, represented by a positive operator-valued measure (POVM) $\mathcal{M} := \{M_y^x\}$, where x denotes the control parameters and y signifies the measurement outcome. For simplicity, we will use the notation x to refer to both the controls of the evolution and those of the measurement. The probability of obtaining outcome y can be computed using the Born rule, yielding

$$p(y|\boldsymbol{\theta}, x) := \mathrm{tr}\left(M_y^x \rho_{\boldsymbol{\theta},x}\right) \ . \tag{15}$$

If the measurement is projective, the system transitions to a known state, thereby extracting the maximum possible amount of information from $\rho_{x,\boldsymbol{\theta}}$. The probe is subsequently reinitialized in the reference state ρ, encoded, and measured again, with the outcome probability given by the same expression in Eq. (15). In the case of weak measurement (non-projective), information about $\boldsymbol{\theta}$ remains encoded in the probe state, and reinitialization is not performed. The probe may or may not undergo evolution $\mathcal{E}_{x',\boldsymbol{\theta}}$ again, potentially with different controls x'. Following this, the probe is measured again using a different POVM $\mathcal{M}' := \{M_{y'}^{x'}\}$, leading to the outcome y'. This procedure can be iterated multiple times until a projective measurement is performed on the probe, at which point its state is reinitialized. For weak measurements, the Born rule prescribes an outcome probability that depends on the entire trajectory of previous controls and measurement outcomes. We denote by $\boldsymbol{x}_t := (x_0, x_1, \ldots, x_t)$ and $\boldsymbol{y}_t := (y_0, y_1, \ldots, y_t)$ the tuples containing the controls and outcomes up to the t-th iteration, respectively. The probability of obtaining y_{t+1} at the $(t+1)$-th step is given by

$$p(y_{t+1}|\boldsymbol{x}_{t+1}, \boldsymbol{y}_t, \boldsymbol{\theta}) := \mathrm{tr}\left(M_{y_{t+1}}^{x_{t+1}} \rho_{\boldsymbol{x}_t, \boldsymbol{y}_t, \boldsymbol{\theta}}\right) \ . \tag{16}$$

The case of continuous measurement can be simulated by taking the appropriate limits, though it is beyond the scope of this work.

B Implementation of the particle filter

B.1 Bayesian update

If we perform a projective measurement on the probe at each step, the probability of observing outcome y, given the control x and the true value $\boldsymbol{\theta}$ of the unknown parameters, is reported in Eq. (4). To recover the value of $\boldsymbol{\theta}$, we apply the principles of Bayesian estimation. Starting from the prior distribution $\pi(\boldsymbol{\theta})$ on $\boldsymbol{\theta}$, we calculate the posterior probability distribution using Bayes' rule, expressed as

$$P(\boldsymbol{\theta}|x, y) = \frac{p(y|x, \boldsymbol{\theta})\pi(\boldsymbol{\theta})}{P(y)} = \frac{p(y|x, \boldsymbol{\theta})\pi(\boldsymbol{\theta})}{\int p(y|x, \boldsymbol{\theta})\pi(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}} \ . \tag{17}$$

The denominator serves as the normalization required for $P(\boldsymbol{\theta}|x, y)$ to qualify as a probability density. For a series of measurements, we apply Bayes' rule iteratively, using the posterior computed at the previous step as the prior for the next one. Given the tuples of controls $\boldsymbol{x}_{t+1}$ and outcomes $\boldsymbol{y}_{t+1}$, we compute the posterior at the $(t+1)$-th step from the posterior at the t-th step using the formula

$$P(\boldsymbol{\theta}|\boldsymbol{x}_{t+1}, \boldsymbol{y}_{t+1}) = \frac{p(y_{t+1}|x_{t+1}, \boldsymbol{\theta})P(\boldsymbol{\theta}|\boldsymbol{x}_t, \boldsymbol{y}_t)}{\int p(y_{t+1}|x_{t+1}, \boldsymbol{\theta})P(\boldsymbol{\theta}|\boldsymbol{x}_t, \boldsymbol{y}_t)\, \mathrm{d}\boldsymbol{\theta}} \ . \tag{18}$$

It is important to note that for each measurement, the probability of obtaining y_{t+1} is independent of the outcomes and controls up to that point and depends solely on x_{t+1}. This independence arises from the reinitialization of the probe following projective measurements. To efficiently perform the Bayesian update on a computer, we utilize the particle filter method (PF). This method represents the posterior distribution using a discrete set of points in the parameter space Θ, each associated with a weight. Essentially, this approximates the posterior distribution as a sum of δ-functions, expressed as

$$P(\boldsymbol{\theta}|\boldsymbol{x}_t, \boldsymbol{y}_t) \simeq \sum_{j=1}^{N} w_j^t \delta(\boldsymbol{\theta} - \boldsymbol{\theta}_j) \ , \tag{19}$$

where the values $\{\boldsymbol{\theta}_j\}_{j=1}^N$ are referred to as particles and $\{w_j^t\}_{j=1}^N$ are the weights at step t. The values of the particles are sampled from the initial prior $\pi(\boldsymbol{\theta})$, while the weights are initialized uniformly across all particles, i.e., $\omega_j^0 := \frac{1}{N}$. The weights vary with each step due to the Bayesian update of the posterior in Eq. (18), which corresponds to the transformation

$$w_j^{t+1} = \frac{p(y_{t+1}|x_{t+1}, \boldsymbol{\theta}_j) w_j^t}{\sum_{j=1}^N p(y_{t+1}|x_{t+1}, \boldsymbol{\theta}_j) w_j^t} \,. \tag{20}$$

The particles $\{\boldsymbol{\theta}_j\}_{j=1}^N$ may also vary with the step t. We will introduce a resampling procedure that, when triggered, generates a new set of particles, meaning they do not necessarily remain constant throughout the estimation process. For notational simplicity, we will not include a time index on $\boldsymbol{\theta}_j$. We denote the set of particles and their weights as $\mathfrak{p}_t := \{\boldsymbol{\theta}_j, w_j^t\}_{j=1}^N$, referred to as the PF ensemble.

B.2 Moments of the posterior

Computing the first moments of the posterior, namely the mean value and the covariance matrix, corresponds to straightforward linear algebra operations on the PF ensemble, expressed as

$$\widehat{\boldsymbol{\theta}}_t := \int \boldsymbol{\theta} P(\boldsymbol{\theta}|\boldsymbol{x}_t, \boldsymbol{y}_t)\, \mathrm{d}\boldsymbol{\theta} \simeq \sum_{j=1}^N w_j^t \boldsymbol{\theta}_j \,, \tag{21}$$

and

$$\Sigma_t := \int (\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_t)(\boldsymbol{\theta} - \widehat{\boldsymbol{\theta}}_t)^\intercal P(\boldsymbol{\theta}|\boldsymbol{x}_t, \boldsymbol{y}_t)\, \mathrm{d}\boldsymbol{\theta} \simeq \sum_{j=1}^N w_j^t (\boldsymbol{\theta}_j - \widehat{\boldsymbol{\theta}}_t)(\boldsymbol{\theta}_j - \widehat{\boldsymbol{\theta}}_t)^\intercal \,. \tag{22}$$

The mean value of the posterior $\widehat{\boldsymbol{\theta}}$ serves as our estimator for all continuous parameters throughout this paper. As the estimation progresses, the weights typically concentrate on a limited number of particles, while the remaining particles become negligible for the estimation and may only consume memory. The precision of the estimation is limited by the average distance between the points $\boldsymbol{\theta}_j$, which depends on the prior distribution $\pi(\boldsymbol{\theta})$ and the number of particles N. The next section will demonstrate how the introduction of a resampling scheme can address this issue by generating a new set of particles $\{\boldsymbol{\theta}'_j\}_{j=1}^N$, which should be situated in regions where the posterior distribution is concentrated. This leads to an increased density of particles in that region, enhancing the resolution in distinguishing close values of $\boldsymbol{\theta}$. Throughout the paper, we utilize the same symbols for the "theoretical" moments of the posterior (which are not directly accessible) and the approximations of these quantities computed from the PF. The context will clarify which quantities are being referenced at any given time.

B.3 Resampling scheme

While for a small number of unknown parameters we could still obtain good performances even if no resampling procedure is performed, it is essential for larger dimensions. Indeed the density of particles, i.e. the resolution in $\boldsymbol{\theta}$, after the initialization, is inversely proportional to the volume of the parameter space, which grows exponentially in the number d of dimension of the Θ space. To solve this problem it is typical to perform during the estimation a resampling of the particles according to the posterior distribution, which is triggered by the condition

$$N_{\text{eff}} := \frac{1}{\sum_{j=1}^N (w_j^i)^2} < r_t N \,, \tag{23}$$

where r_t is the resampling threshold that is kept fixed at $r_t = 0.5$ in all the simulations of the paper. The left-hand side of Eq. (23) is sometimes called the effective number of particles N_{eff}.

Soft resampling

The simplest resampling scheme prescribes the extraction of N samples with repetitions from the set of indexes $J = \{1, \cdots, N\}$, each weighted with the corresponding w_j, $j \in J$. We will call $\phi(j) : J \to J$ the map that gives the outcome of the j-th extraction event. The indexes $j \in J$ corresponding to the particles $\boldsymbol{\theta}_j$ that have large weights are extracted more frequently, while the particles with small weights tend to disappear. In our implementation, we considered a slightly more general version of this procedure which goes under the name

of soft resampling [68], that is, we mix the probability distribution represented by the weights $\{w_j\}_{j=1}^N$ with a uniform distribution on $\{\boldsymbol{\theta}_j\}_{j=1}^N$ by constructing the soft-weights q_j defined as

$$q_j := \alpha w_j + (1-\alpha)\frac{1}{N} , \tag{24}$$

where $\alpha \in [0,1]$ is a parameter characterizing the effectiveness of the resampling. With $\alpha = 1$ we have the traditional procedure, while with $\alpha = 0$ no actual resampling is performed, because we extract the new particles from a uniform distribution, just like at the beginning. With $\alpha = 0$ the particles with low weights are not cut away from the ensemble but persist after the process. With an intermediate value of α (by default we set $\alpha = 0.5$) only a fraction α of the particles are effective for the resampling, because the other fraction $(1-\alpha)$ is expected to be distributed uniformly. We call $\boldsymbol{\theta}'_j$ the new particles extracted from q_j, i.e.

$$\boldsymbol{\theta}'_j = \boldsymbol{\theta}_{\phi(j)} . \tag{25}$$

Their corresponding weights are chosen, so that the ensemble of the PF represents the same distribution as before the resampling. These are

$$w'_j \propto \frac{w_{\phi(j)}}{q_{\phi(j)}} = \frac{w_{\phi(j)}}{\alpha w_{\phi(j)} + (1-\alpha)\frac{1}{N}} , \tag{26}$$

that still need to be normalized. With this choice for w'_j the PF represents the correct posterior even though the particles have been sampled from a different distribution. The probability density function represented by the PF is, roughly speaking, proportional to the product of the weights w'_j and the density of particles at the position $\boldsymbol{\theta}'_j$, i.e. $q_{\phi(j)}$, which with our choice for w'_j is exactly $w_{\phi(j)}$, i.e. the weight of the particle $\boldsymbol{\theta}'_j$ prior to the resampling step. In the next section, we detail this relation. The reader that is interested in the successive steps of the resampling can however skip it. The soft resampling scheme, which is based on importance sampling [106], will be crucial in making the PF differentiable [107, 108]. We might want, in general, to perform a subsampling of the particles, that is, we sample from the distribution in Eq. (24) not N but γN particles, with $0 < \gamma \leq 1$. We will later in the resampling routine propose $(1-\gamma)N$ new particles that will help us in representing the posterior better, so that we have in total after the resampling step N particles again. In this case the weights in Eq. (26) will be normalized as $w'_j \to \frac{w'_j}{\mathcal{C}}$, where $\mathcal{C}$ is such that

$$\frac{1}{\mathcal{C}}\sum_{j=1}^{\gamma N} w_j = \gamma , \tag{27}$$

By default we set $\gamma = 0.99$, that is, only 1% of the particles after the resampling are new.

Particle filters and importance sampling

In this section we review the core ideas underlying the functioning of a particle filter and the principle of importance sampling, as it is applied in our implementation of the soft resampling. Consider a distribution $P(\boldsymbol{\theta})$, from which we sample N particles $\boldsymbol{\theta}_j$ with $j = 1, \cdots, N$. Let us define an hypercube $\mathcal{C}$ of volume $\mathrm{d}\boldsymbol{\theta}$ centred around $\boldsymbol{\theta}$, and let us call $n(\boldsymbol{\theta}, d^N\boldsymbol{\theta})$ the number of particles in the said hypercube, i.e.

$$n(\boldsymbol{\theta}, \mathrm{d}\boldsymbol{\theta}) := \sum_{j=1}^{N} \chi_{\mathcal{C}}(\boldsymbol{\theta}_j) \tag{28}$$

with $\chi_{\mathcal{C}}$ being the characteristic function of the hypercube. We can write

$$\frac{1}{N} n(\boldsymbol{\theta}, d^N\boldsymbol{\theta}) \to P(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} \quad \text{for } N \to \infty , \tag{29}$$

that is in the limit of large N the fraction of particles in the hypercube tends to the probability in such volume element. In a PF we associate to each particle $\boldsymbol{\theta}_i$ a weight w_i and we can define the total weight in the hypercube $\mathcal{C}$ as

$$P(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} \simeq w(\boldsymbol{\theta}, d^N\boldsymbol{\theta}) := \sum_{j=1}^{N} w_j \chi_{\mathcal{C}}(\boldsymbol{\theta}_j) . \tag{30}$$

This total weight is the probability distribution actually represented by the PF. In the limit of large N, for a smooth distribution, we can consider the weight a function of the point $w(\boldsymbol{\theta})$, which varies smoothly in space

and is approximatively constant in the hypercube $\mathcal{C}$. This leads us to write

$$P(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} \simeq w(\boldsymbol{\theta})\sum_{j=1}^{N}\chi_{\mathcal{C}}(\boldsymbol{\theta}_j) = w(\boldsymbol{\theta})n(\boldsymbol{\theta},\mathrm{d}\boldsymbol{\theta})\ . \tag{31}$$

This means that the distribution represented by the particle filter is proportional to the product of the weights and the density of the particles. This is however not the only way to represent $P(\boldsymbol{\theta})$. Suppose that for whatever reason we sample the particles $\boldsymbol{\theta}_j$ from $Q(\boldsymbol{\theta})$, but that we actually want to represent the distribution $P(\boldsymbol{\theta})$. Then we can multiply the weights $w_j = 1/N$ of each particle $\boldsymbol{\theta}_j$ with the corrective factor $P(\boldsymbol{\theta}_j)/Q(\boldsymbol{\theta}_j)$, which remains approximately constant inside the region $\mathcal{C}$, i.e.

$$w(\boldsymbol{\theta}, d^N\boldsymbol{\theta}) = \frac{1}{N}\sum_{j=1}^{N}\frac{P(\boldsymbol{\theta}_j)}{Q(\boldsymbol{\theta}_j)}\chi_{\mathcal{C}}(\boldsymbol{\theta}_j) \simeq \frac{P(\boldsymbol{\theta})}{Q(\boldsymbol{\theta})}\frac{1}{N}\sum_{j=1}^{N}\chi_{\mathcal{C}}(\boldsymbol{\theta}_j) = \frac{P(\boldsymbol{\theta})}{Q(\boldsymbol{\theta})}\frac{n(\boldsymbol{\theta}, d^N\boldsymbol{\theta})}{N}\ , \tag{32}$$

since now the particles are distributed spacially according to $Q(\boldsymbol{\theta})$ the density of particles will tend to $Q(\boldsymbol{\theta})$ for large N that, according to Eq. (29), gives $w(\boldsymbol{\theta}, d^N\boldsymbol{\theta}) \to Q(\boldsymbol{\theta})$, therefore we have $w(\boldsymbol{\theta}, d^N\boldsymbol{\theta}) \simeq P(\boldsymbol{\theta})$. In the case of soft resampling the distribution $Q(\boldsymbol{\theta})$ is constructed from $P(\boldsymbol{\theta})$ as

$$Q(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} := \left[\alpha w(\boldsymbol{\theta}) + (1-\alpha)\frac{1}{N}\right] n(\boldsymbol{\theta},\mathrm{d}\boldsymbol{\theta})\ , \tag{33}$$

and the factor that multiplies the weights is $P(\boldsymbol{\theta})/Q(\boldsymbol{\theta}) = w(\boldsymbol{\theta})/\left[\alpha w(\boldsymbol{\theta}) + (1-\alpha)\frac{1}{N}\right]$. The factor used in Eq. (26) for the particle at $\boldsymbol{\theta}'_j$ contains $w_\phi(j)$, which is the weight at this point in the original distribution $P(\boldsymbol{\theta})$.

Gaussian perturbation

After the soft resampling, we add a perturbation to the particles as proposed in [78], that is, we define

$$\boldsymbol{\theta}''_j := \beta\boldsymbol{\theta}'_j + (1-\beta)\widehat{\boldsymbol{\theta}}_t + \boldsymbol{\delta}\ , \tag{34}$$

where $\beta \in (0,1]$, $\widehat{\boldsymbol{\theta}}_t$ is the mean of the posterior approximated in Eq. (21) and $\boldsymbol{\delta}$ is a random variable distributed according to

$$\boldsymbol{\delta} \sim \mathcal{N}(0, (1-\beta^2)\Sigma_t)\ . \tag{35}$$

With this expression we move the particles toward the mean of the posterior, which is our estimator for $\boldsymbol{\theta}$ and at the same time we lift the degeneracy of the $\boldsymbol{\theta}'_j$, that comes about because the particle $\boldsymbol{\theta}_j$ with high weights appear many times in the new particles ensemble. Were the degeneracy not removed, all these copies of the same particle wouldn't contribute much to improving the resolution of the PF. This holds true unless they are perturbed, at which point they can encode the small scale behavior of the posterior. Because of the perturbation in Eq. (34) the PF does not represent anymore the posterior $P(\boldsymbol{\theta}|\boldsymbol{x}_t, \boldsymbol{y}_t)$ exactly. We now compute the probability distribution for $\boldsymbol{\theta}''_j$ after the perturbation step. The particles are distributed in the space according to the q_j weights in Eq. (24) and we call this distribution $Q(\boldsymbol{\theta}')$. Let us write Eq. (34) as $\boldsymbol{\theta}''_j = \beta\boldsymbol{\theta}'_j + \boldsymbol{\delta}'$ with

$$\boldsymbol{\delta}' \sim \mathcal{N}((1-\beta)\widehat{\boldsymbol{\theta}}_t, (1-\beta^2)\Sigma_t)\ , \tag{36}$$

being a perturbation with non-null mean value. Then the probability density for $\boldsymbol{\theta}''_j$ is the convolution of the probability of a particle being at position $\boldsymbol{\theta}'$ and the probability of the noise causing a displacement $\boldsymbol{\delta}' = \boldsymbol{\theta}'' - \beta\boldsymbol{\theta}'$, i.e.

$$\widetilde{Q}(\boldsymbol{\theta}'') = \int Q(\boldsymbol{\theta}')g_\beta(\boldsymbol{\theta}'' - \beta\boldsymbol{\theta}')\,\mathrm{d}\boldsymbol{\theta}' = \sum_{j=1}^{\gamma N} q_{\phi(j)}g_\beta(\boldsymbol{\theta}'' - \beta\boldsymbol{\theta}'_j)\ , \tag{37}$$

where g_β is the Gaussian probability density associated to $\boldsymbol{\delta}'$, i.e.

$$g_\beta(\boldsymbol{\theta}) := (2\pi)^{-\frac{\mathrm{d}}{2}}(1-\beta^2)^{-\frac{1}{2}}\det(\Sigma_t)^{-\frac{1}{2}}\exp\left[-\frac{1}{2(1-\beta^2)}\left(\boldsymbol{\theta} - (1-\beta)\widehat{\boldsymbol{\theta}}_t\right)^{\intercal}\Sigma_t^{-1}\left(\boldsymbol{\theta} - (1-\beta)\widehat{\boldsymbol{\theta}}_t\right)\right]\ . \tag{38}$$

In Eq. (37) we also substituted the integral with a summation being the probability $Q(\boldsymbol{\theta}')$ discrete. According to the principles of importance sampling the distribution represented by a PF is the product of the weights and the density of particles, which reads

$$\widetilde{P}(\boldsymbol{\theta}_j) \propto \frac{P(\boldsymbol{\theta})}{Q(\boldsymbol{\theta})}\widetilde{Q}(\boldsymbol{\theta}_j) \simeq P(\boldsymbol{\theta})\ , \tag{39}$$

In principle, we could correct the distribution for this perturbation by computing exactly Eq. (37) and accounting for it in the weights w'_j, in our implementation we don't do it however, since it would be very small anyway.

New particles proposal

We still need to produce $(1-\gamma)N$ new particles and we do it by extracting them from the Gaussian distribution with the same two first moments of the PF ensemble, i.e.

$$\boldsymbol{\theta}_j'' \sim \mathcal{N}\left(\widehat{\boldsymbol{\theta}}, \Sigma\right) , \tag{40}$$

for $j = \gamma N, \cdots, N$. The mean and the covariance matrix are defined in Eq. (21) and Eq. (22) respectively. This is done again to increase the density of particles in the region of high probability, but it works properly only for unimodal distributions. The weights of these new particles are set to $w_j' = \frac{1}{N}$, so that their normalization is

$$\sum_{j=\gamma N}^{N} w_j' = 1 - \gamma . \tag{41}$$

This extra particles and weights are concatenated directly to $\{\boldsymbol{\theta}_j''\}_{j=1}^{\gamma N}$ and $\{w_j'\}_{j=1}^{\gamma N}$. We then rename the new weights and particles, i.e. $w_j' \to w_j$ and $\boldsymbol{\theta}_j'' \to \boldsymbol{\theta}_j$, and with that the resampling procedure is concluded. In doing the last step of proposing new particles we are mixing the distribution represented by the PF $P(\boldsymbol{\theta})$ as it comes out of the perturbation step in Eq. (39) with the distribution g_0 in Eq. (38). At the end the PF ensemble represents the distribution

$$P'(\boldsymbol{\theta}) = \gamma P(\boldsymbol{\theta}) + (1-\gamma) g_0(\boldsymbol{\theta}) . \tag{42}$$

Again we do not correct for this distortion, which could be done by modifying the weights properly.

Resampling of the batch

In order to compute the precision of the estimation, we need the results of many runs of the simulation, possibly executed in parallel on a GPU. In these circumstances the resampling is performed on all the instances of the estimation as soon as the condition Eq. (40) holds true for at least a fraction f of the estimations in the batch, which by default is set to $f = 0.98$. The premature resampling of an estimation run will have a quite strong detrimental effect on the goodness of the posterior represented by the PF, on the contrary a late resampling is much less probable to distort the distribution, this is the reason why we set f so close to one, that is, we want to limit as much as possible the number of simulations that are prematurely resampled. With the current implementation at each step either all the simulations is the batch are resampled or none. An improvement to the PF would be to resample selectively only those runs that are in need of resampling, and leave the other untouched until they satisfy Eq. (40), so that whatever number of runs could be resampled at each step. The complete resampling cycle, including the extraction and the new particle and the importance sampling is represented in Fig. 7.

B.4 State particle filter

In this section we describe what happens when we are acting with weak (non-projective) measurements on the probe. In this case the probability to observe the outcome y_{t+1} at the step $t+1$ depends on all the string of previous outcomes and controls, that is on the whole trajectory $\tau := (\boldsymbol{x}_t, \boldsymbol{y}_t)$, as well as on the current control x_{t+1}. This means we must substitute $p(y_{t+1}|x_{t+1}, vtheta)$ with $p(y_{t+1}|\boldsymbol{\theta}, \boldsymbol{x}_{t+1}, \boldsymbol{y}_t)$ in Eq. (20). Since we avoid the reinitialization of the probe, its state depends on all the evolution history. With this change in the outcome probability all the formulas of the previous section remain valid. To compute $p(y_{t+1}|\boldsymbol{\theta}, \boldsymbol{x}_{t+1}, \boldsymbol{y}_t)$, we need to keep track of the state of the probe. In order to do so we introduce the state particle filter. In this data structure we save for each particle $\boldsymbol{\theta}_j$ the state of probe had the system evolved under the action of $\mathcal{E}_{\boldsymbol{\theta}_j, x}$, with the controls and the outcomes being the ones actually applied/observed in the evolution, we indicate this state with $\rho_{\boldsymbol{\theta}_j, \tau_t}$. To this state we associate the weight w_j of the particle $\boldsymbol{\theta}_j$. The state particle filter represents the posterior probability distribution for the state of the probe conditioned on the trajectory τ_t. The expression for $\rho_{\boldsymbol{\theta}_j, \tau_t}$ reads

$$\rho_{\boldsymbol{\theta}_j, \tau_t} = \mathcal{M}_{y_t}^{x_t} \circ \mathcal{E}_{\boldsymbol{\theta}_j, x_t} \circ \mathcal{M}_{y_{t-1}}^{x_{t-1}} \circ \mathcal{E}_{\boldsymbol{\theta}_j, x_{t-1}} \circ \cdots \circ \mathcal{M}_{y_1}^{x_1} \circ \mathcal{E}_{\boldsymbol{\theta}_j, x_1}(\rho) , \tag{43}$$

where

$$\mathcal{M}_{y_t}^{x_t}(\rho) := \frac{M_{y_t}^{x_t} \rho M_{y_t}^{x_t \dagger}}{\mathrm{tr}\left[M_{y_t}^{x_t} \rho M_{y_t}^{x_t \dagger}\right]} , \tag{44}$$

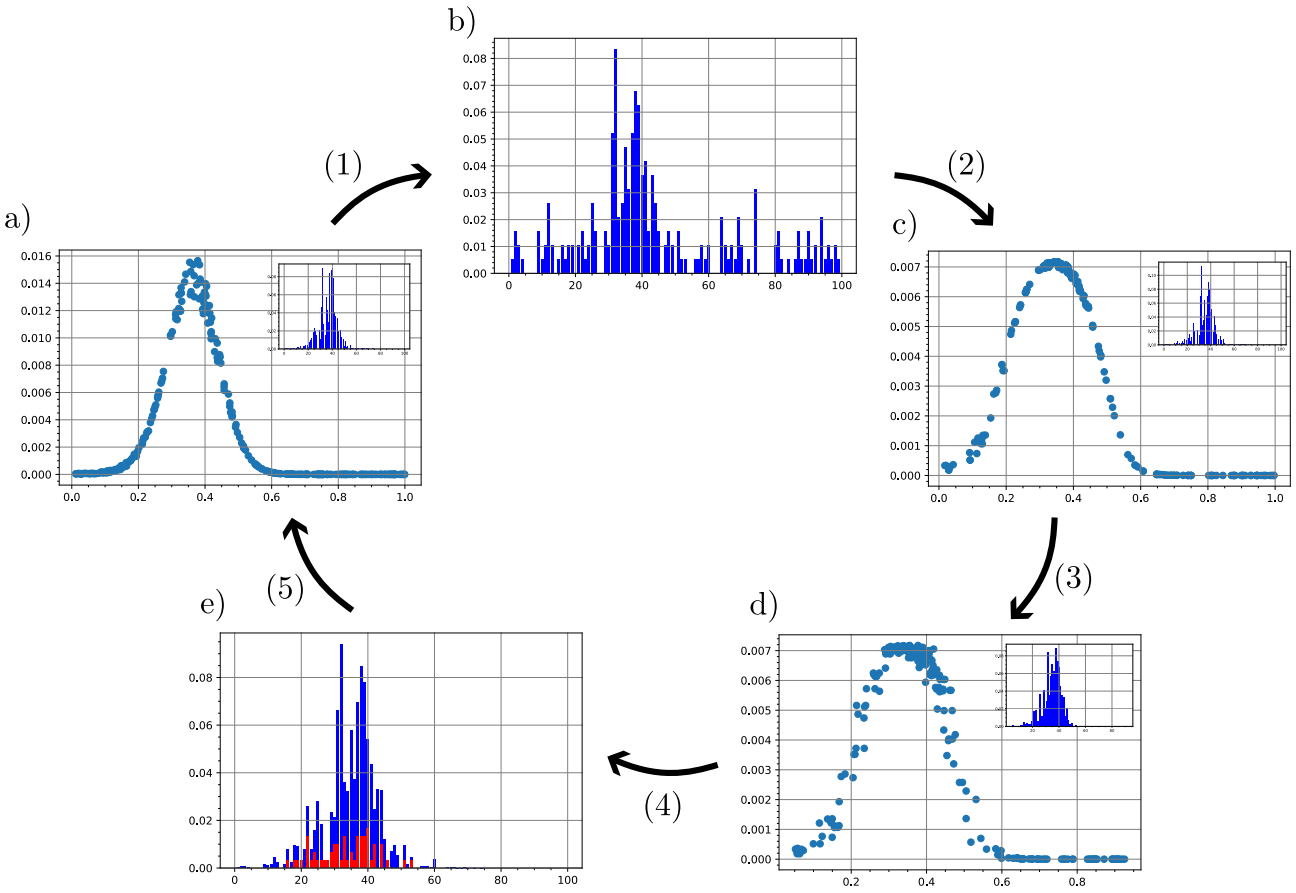


Figure 7: The ensemble of the PF before the resampling is represented in a), the scatter plot are the points $(\boldsymbol{\theta}_j, w_j)$. This plot doesn't represent directly the posterior, because it doesn't take into account the density of particles. In all the plots, the inserted histogram is the actual posterior represented by the PF. Once the condition Eq. (23) is satisfy the first step of the resampling is executed, which is the transformation (1) and corresponds to sampling with repetitions from Eq. (24), the plot b) is the spatial density of particles after this action. The scatter plot c) is the distribution of the PF after the weights have been corrected according to Eq. (26) (transformation (2)). (3) is the application of the Gaussian noise in Eq. (34) and (4) is the sampling of the extra proposed particles in Eq. (40). Plot e) is the distribution represented by the PF when the resampling routine is complete, where in red the contribution of the new particles is highlighted. At last (5) is the repeated application of the Bayes rule Eq. (20), following some new measurement on the probe, that leads to the ensemble of the PF being again in need of resampling. To emphasise the effect of each transformation we have set $\alpha = 0.5, \beta = 0.9, \gamma = 0.8$. The total number of particles was $N = 10^3$ and the effective number of particles in a), that is before the resampling, was $N_{\text{eff}} = 93.8$. In the histograms the interval $[0, 1]$ of the scatter plots has been mapped to $[0, 100]$.

is the backreaction of the measurement on the state of the probe. The estimator for the probe state at the step t is

$$\widehat{\rho}_{\tau_t} := \sum_{j=1}^{N} w_j \rho_{\boldsymbol{\theta}_j, \tau_t} \ , \tag{45}$$

that is, the mean of the state on the posterior distribution for the parameters. The estimator $\widehat{\rho}_{\tau_t}$ can then be fed to the agent, to contribute to the computation of the next control. When the resampling is performed on the PF ensemble we get a new set of particles $\boldsymbol{\theta}'_j$ and their corresponding states must be also updated. This means we have to keep track of the vectors $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ in the simulations and recompute the evolution of the whole state particle filter from the beginning, so that we get $\rho_{\boldsymbol{\theta}'_j, \tau_t}$. From the computational point of view, the fact that we need these rather memory intensive structures of the PF and the state PF tells us that the optimization loop presented here can be applied only to rather small and simple quantum sensors.

B.5 Multimodal posterior distributions

The resampling procedure presented in the previous section has some limitation in dealing with multimodal distributions. In this case the mean of the posterior may lay in a region of relatively low probability between

two peaks and the accumulation of particles in this region after a resampling would be detrimental to the precision of the estimation. From its own design it would be difficult to modify the PF so that it accounts for multiple maxima. The informations that we can easily extract from the PF are its moments and from them the actual positions of the maxima are not straightforward to obtain. Multimodal posterior distributions are however common in quantum metrology. For example in multiphase estimation, like the measurement of the hyperfine interaction in NV-^{13}C. To promote the preservation of secondary features in the posterior distribution we can use multiple particle filter at once. In this situation a set of PFs, with different priors, are updated in parallel and only together represent the full Bayesian posterior. To reduce the memory requirement of such approach we could consider simple Gaussian distributions instead of full PFs. We start by approximating the prior distribution $\pi(\boldsymbol{\theta})$ with as a sum of L Gaussians:

$$\pi(\boldsymbol{\theta}) \simeq \sum_{l=1}^{L} w_l \mathcal{N}(\boldsymbol{\mu}_l, \boldsymbol{\sigma}_l) \, . \tag{46}$$

If the parameters $\boldsymbol{\mu}_l, \boldsymbol{\sigma}_l$ are fixed then the Bayesian update step can be done by solving a linear regression problem to find the best new values for $\{w_l\}_{l=1}^{L}$ that represent the posterior. In this way the PF has however a limited resolution, determined by the initial Gaussian. If we also let $\boldsymbol{\mu}_l, \boldsymbol{\sigma}_l$ change during the Bayesian update step, then we solve the problem of having limited resolution, but we now have to deal with a non-linear regression problem.

C Differentiability of the particle filter

In this section we discuss what happens when the resampling routine of the particle filter is switched on, and, in particular, what we need to do to assure that the gradient produced by the automatic differentiation is correct.

C.1 Differentiable PF through reparametrization and soft resampling

Differentiability of the soft resampling

As seen in Appendix D.3, the gradient can't be propagated through randomly extracted variables, therefore when the categorical resampling is executed, the particles $\boldsymbol{\theta}'_j$ in Eq. (25) don't have any connection with the controls, i.e.

$$\frac{\mathrm{d}\boldsymbol{\theta}'_j}{\mathrm{d}\boldsymbol{\lambda}} = 0 \, . \tag{47}$$

Similarly, the weights are reinitialized and loose every dependence on the history of the estimation. At the moment of taking the gradient we won't be able to account for anything that happened before the last resampling. This means that the training routine optimizes the agent only for the later steps, although what has been learnt in this context may be useful also for the earlier measurements. The soft resampling with $\alpha < 1$, introduced in Appendix B, is able to partially remove this obstacle. With this trick the dependence on $\boldsymbol{\lambda}$ is passed from the old weights to the new ones through Eq. (26). However, the gradient doesn't backpropagate entirely but it is attenuated by the factor $1 - \alpha$. The price to pay for propagating the gradient is that the N particles are not all fully effective for the resampling, instead only a fraction α of them participate to it. As discussed in Appendix D.3, since we are extracting stochastic variables from the distribution in Eq. (24), we should also add the corresponding log-likelihood terms $\sum_j \log q_{\phi(j)}$ to the loss. However adding so many terms would increase too much the variance of the gradient. Either we don't account for these log-likelihoods and we accept the gradient to have a bias, or we use the correction introduced in [69], that prescribes to substitute the definition of the new weights w'_j with an appropriate surrogate expressions. Introducing this correction is the default behaviour of our code but it can be applied only if the loss $\ell(\widehat{\boldsymbol{\theta}}, \boldsymbol{\theta})$ is of a certain form. It can be proved that for the MSE defined in Eq. (7) this gives the correct gradient. See Appendix C.2 for a complete discussion.

Differentiability of the perturbation

The next transformation on the particles is the perturbation of Eq. (34). Again we would be unable to propagate the gradient through the perturbation $\boldsymbol{\delta}'$, if we extracted it directly from the Gaussian $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$, with $\boldsymbol{\mu} = (1-\beta)\widehat{\boldsymbol{\theta}}_t$ and $\Sigma = (1-\beta^2)\Sigma_t$. For this reason we apply the reparametrization trick and write

$$\boldsymbol{\delta}'(\boldsymbol{y}_t, \boldsymbol{\lambda}) = \Sigma(\boldsymbol{y}_t, \boldsymbol{\lambda})\boldsymbol{u} + \boldsymbol{\mu}(\boldsymbol{y}_t, \boldsymbol{\lambda}) \, , \tag{48}$$

where $\boldsymbol{u}$ is extracted from the multivariate standard Gaussian $\boldsymbol{u} \sim \mathcal{N}(0, \mathbb{1})$. The perturbation is now a differentiable functions of $\boldsymbol{\lambda}$. For the extraction of $\boldsymbol{u}$ we do not need to add a corresponding log-likelihood term, as discussed in Appendix D.3, because its probability density function doesn't depend on $\boldsymbol{\lambda}$.

Differentiability of the proposed particles

For the last step of the resampling, which consists in proposing new particles $\boldsymbol{\theta}''_j$, extracted from $\mathcal{N}(\widehat{\boldsymbol{\theta}}_t, \Sigma_t)$, we again exploit the reparametrization trick and write

$$\boldsymbol{\theta}''_j(\boldsymbol{y}_t, \boldsymbol{\lambda}) = \Sigma_t(\boldsymbol{y}_t, \boldsymbol{\lambda})\boldsymbol{u}_j + \widehat{\boldsymbol{\theta}}_t(\boldsymbol{y}_t, \boldsymbol{\lambda}) \,, \tag{49}$$

which is again differentiable and doesn't require a log-likelihood term.

Differentiability of the state particle filter

Regarding the differentiability of the state particle filter discussed in Appendix B.4, we observe that its elements are functions of $\boldsymbol{\lambda}$ through the trajectory $\tau_t(\boldsymbol{\lambda}) = (\boldsymbol{x}_t(\boldsymbol{y}_t, \boldsymbol{\lambda}), \boldsymbol{y}_t)$:

$$\rho_j(\boldsymbol{\lambda}) := \rho_{\boldsymbol{\theta}_j, \tau_t(\boldsymbol{\lambda})} \,. \tag{50}$$

Under the assumption that the encoding and the measurements appearing in Eq. (43) are differentiable we can propagate the gradient through the evolution of the probe in Eq. (43) . When the particles are resampled and the new states are computed, the dependence of the new state $\rho_j(\boldsymbol{\lambda})$ on the old weights of the PF, and therefore on $\boldsymbol{\lambda}$, persists through the measurement backreaction operators $\mathcal{M}^{y_t}_{x_t(\boldsymbol{\lambda})}$. A new dependence on $\boldsymbol{\lambda}$ appears in the evolution map $\mathcal{E}_{\boldsymbol{\theta}_j(\boldsymbol{\lambda}), x_t(\boldsymbol{\lambda})}$, coming from the new particles $\boldsymbol{\theta}_j(\boldsymbol{\lambda})$, so that we can write

$$\rho'_j(\boldsymbol{\lambda}) := \rho_{\boldsymbol{\theta}_j(\boldsymbol{\lambda}), \tau_t(\boldsymbol{\lambda})} \,. \tag{51}$$

C.2 Differentiable PF through the correction of Ścibior and Wood

In [69] a correction was introduced to make the resampling procedure in a PF differentiable. This can be implemented in place of the soft resampling, or alongside with it. The default behaviour of our software is to perform the soft resampling with $\alpha = 0.5$ alongside the Ścibor and Wood correction. With this choice only half of the gradient is backpropagated through soft resampling, the other half is done by the Ścibor and Wood correction. The former prescribes to modify the normalized weights w'_j of Eq. (26) to

$$\widetilde{w}'_j \leftarrow w'_j \frac{q_{\phi(j)}}{\mathrm{sg}\left[q_{\phi(j)}\right]} \,, \tag{52}$$

where the meaning of the symbols is that of Appendix B.3. In this formula we are using the stop gradient operator $\mathrm{sg}\,[\cdot]$, which is an instruction that tells the automatic differentiation frameworks not to compute the derivatives of the expression inside the operator. This correction has no effects in the forward pass, but produces additional gradient terms in the backward pass. We see in this section, that for a MSE loss the extra terms in the gradient appearing because of this surrogate expression are exactly the log-likelihoods that we would have to insert following the conclusions of Appendix D.3, although this observation can't be extended to a generic loss. Let us start from the expression for the MSE, when one and only one resampling is performed in the whole experiment, at step t, i.e.

$$\Delta^2\widehat{\boldsymbol{\theta}} = \int \ell(\widehat{\boldsymbol{\theta}}, \boldsymbol{\theta}) P(\tau_{t+1:M-1}|\boldsymbol{\theta}'_j, \boldsymbol{\theta}) \left(\prod_{j=1}^{N} P(\boldsymbol{\theta}'_j|\tau_{0:t})\right) P(\tau_{0:t}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})\, \mathrm{d}\tau_{M-1} \left(\prod_{j=1}^{N} \mathrm{d}\boldsymbol{\theta}'_j\right) \mathrm{d}\boldsymbol{\theta} \,. \tag{53}$$

We assume for clarity that the perturbation and the extraction of the extra particles, in Eq. (48) and Eq. (49) respectively, are turned off, i.e. $\beta = \gamma = 1$. The object $\tau_{\alpha:\beta}$ with α, β integers in $[0, M-1]$ is the trajectory between the steps α and β (extrema included), i.e. $\tau_{\alpha:\beta} = (\boldsymbol{x}_{\alpha:\beta}, \boldsymbol{y}_{\alpha:\beta})$ with $\boldsymbol{x}_{\alpha:\beta} = (x_\alpha, x_{\alpha+1}, \ldots, x_\beta)$ and $\boldsymbol{y}_{\alpha:\beta} = (y_\alpha, y_{\alpha+1}, \ldots, y_\beta)$. Reading Eq. (53) from right to left we encounter the probability densities for all the random variable extractions in chronological order. First the extraction of the true values $\boldsymbol{\theta}$ for the simulation instance, then the trajectory up to the resampling point, then the extraction of the new particles $\boldsymbol{\theta}'_j$, and finally the measurements after the resampling, i.e. the trajectory after the t-th step until the end. This last probability depends also on the values of the new particles, through the posterior distribution momenta, that are passed to the agent that decides the next control. We now insert the expression for $\ell(\widehat{\boldsymbol{\theta}}, \boldsymbol{\theta})$ found in Eq. (7) in Eq. (53).

We postpone the computation of the trace to the end and expand the error matrix for the estimator in Eq. (21), i.e.

$$(\widehat{\boldsymbol{\theta}}-\boldsymbol{\theta})(\widehat{\boldsymbol{\theta}}-\boldsymbol{\theta})^{\intercal}=\sum_{i,j=1}^{N}w'_iw'_j\boldsymbol{\theta}'_i\boldsymbol{\theta}'^{\intercal}_j-\boldsymbol{\theta}\left(\sum_{j=1}^{N}w'_j\boldsymbol{\theta}'^{\intercal}_j\right)-\left(\sum_{j=1}^{N}w'_j\boldsymbol{\theta}'_j\right)\boldsymbol{\theta}^{\intercal}+\boldsymbol{\theta}\boldsymbol{\theta}^{\intercal}\ . \tag{54}$$

Each term in the first summation gives a contribution to $\Delta^2\widehat{\boldsymbol{\theta}}$ equal to

$$\int w'_iw'_j\boldsymbol{\theta}'_i\boldsymbol{\theta}'^{\intercal}_jP(\tau_{t+1:M-1}|\boldsymbol{\theta})P(\boldsymbol{\theta}'_i|\tau_{0:t})P(\boldsymbol{\theta}'_j|\tau_{0:t})P(\tau_{0:t}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})\,\mathrm{d}\tau_{M-1}\,\mathrm{d}\boldsymbol{\theta}'_i\,\mathrm{d}\boldsymbol{\theta}'_j\,\mathrm{d}\boldsymbol{\theta}\ , \tag{55}$$

where we neglect all the integrals on the variables $\boldsymbol{\theta}'_\alpha$ with index $\alpha\neq i,j$, because they do not appear in the integrand. The gradient of this term with respect to $\boldsymbol{\lambda}$ gives rise to the usual likelihood terms for the measurement plus the following extra terms coming from the resampling:

$$\sum_{i,j}^{N}w'_iw'_j\boldsymbol{\theta}'_i\boldsymbol{\theta}'^{\intercal}_j\left(\frac{\mathrm{d}\log P(\boldsymbol{\theta}'_i|\tau_{0:t})}{\mathrm{d}\boldsymbol{\lambda}}+\frac{\mathrm{d}\log P(\boldsymbol{\theta}'_j|\tau_{0:t})}{\mathrm{d}\boldsymbol{\lambda}}\right)\ , \tag{56}$$

for $i,j=1,\dots,N$. Similarly the linear terms in Eq. (54) give the following likelihood terms:

$$-\sum_{j=1}^{N}w'_j\left(\boldsymbol{\theta}'_j\boldsymbol{\theta}^{\intercal}+\boldsymbol{\theta}\boldsymbol{\theta}'^{\intercal}_j\right)\frac{\mathrm{d}\log P(\boldsymbol{\theta}'_j|\tau_{0:t})}{\mathrm{d}\boldsymbol{\lambda}}\ , \tag{57}$$

plus the same terms with $\boldsymbol{\theta}'^{\intercal}_j$. There is no likelihood terms associated to the constant $\boldsymbol{\theta}\boldsymbol{\theta}^{\intercal}$ in Eq. (54). Now we shall see that deriving the surrogate expression gives the same terms in the gradient. Let us write the total derivative of the error matrix:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}(\widehat{\boldsymbol{\theta}}-\boldsymbol{\theta})(\widehat{\boldsymbol{\theta}}-\boldsymbol{\theta})^{\intercal}&=\frac{\mathrm{d}\widehat{\boldsymbol{\theta}}}{\mathrm{d}\boldsymbol{\lambda}}(\widehat{\boldsymbol{\theta}}-\boldsymbol{\theta})^{\intercal}+(\widehat{\boldsymbol{\theta}}-\boldsymbol{\theta})\frac{d\widehat{\boldsymbol{\theta}}^{\intercal}}{\mathrm{d}\boldsymbol{\lambda}}\\&=\left(\sum_{i=1}^{N}\frac{\mathrm{d}\widetilde{w}'_i}{\mathrm{d}\boldsymbol{\lambda}}\boldsymbol{\theta}'_i\right)\left(\sum_{j=1}^{N}w'_j\boldsymbol{\theta}'_j-\boldsymbol{\theta}\right)^{\intercal}+\left(\sum_{i=1}^{N}w'_i\boldsymbol{\theta}'_i-\boldsymbol{\theta}\right)\left(\sum_{j=1}^{N}\frac{\mathrm{d}\widetilde{w}'_j}{\mathrm{d}\boldsymbol{\lambda}}\boldsymbol{\theta}'^{\intercal}_j\right)\ .\end{aligned}$$

Where the derivative doesn't act the weights $\widetilde{w}'_j$ become w'_j. From the definition of $\widetilde{w}'_j$ in Eq. (52) we compute the derivative of the surrogate expression, that is

$$\frac{\mathrm{d}\widetilde{w}'_j}{\mathrm{d}\boldsymbol{\lambda}}=\frac{dw'_j}{\mathrm{d}\boldsymbol{\lambda}}+w'_j\frac{\mathrm{d}\log q_{\phi(j)}}{\mathrm{d}\boldsymbol{\lambda}}\ . \tag{58}$$

We know keep track only of the extra likelihood terms coming from the surrogate part of the weights $\widetilde{w}'_j$ and organize the terms according to the order in $\boldsymbol{\theta}'_j$. We have the second order terms:

$$\sum_{i,j}^{N}w'_iw'_j\boldsymbol{\theta}'_i\boldsymbol{\theta}'^{\intercal}_j\left(\frac{\mathrm{d}\log q_{\phi(i)}}{\mathrm{d}\boldsymbol{\lambda}}+\frac{\mathrm{d}\log q_{\phi(j)}}{\mathrm{d}\boldsymbol{\lambda}}\right)\ , \tag{59}$$

and the first order ones:

$$\sum_{j=1}^{N}w'_j\left(\boldsymbol{\theta}'_j\boldsymbol{\theta}^{\intercal}+\boldsymbol{\theta}\boldsymbol{\theta}'^{\intercal}_j\right)\frac{\mathrm{d}\log q_{\phi(j)}}{\mathrm{d}\boldsymbol{\lambda}}\ , \tag{60}$$

which correspond respectively to Eq. (56) and Eq. (57), once we realize that $P(\boldsymbol{\theta}'_j|\tau_{0:t})=q_{\phi(j)}$. One of the advantages of this approach is the reduce variance of the gradient estimator, which would explode, where we to insert all the likelihood terms for the new particle extractions at the end of the estimation. The correction, however, doesn't produce always the correct gradient for the loss, but only when $\ell(\widehat{\boldsymbol{\theta}},\boldsymbol{\theta})$ is a polynomial function of the weights $\widetilde{w}'_j$. Consider the estimation of single parameter $\theta\in[0,2\pi)$, we might want to use a loss functions $l(\widehat{\theta},\theta)$ that respect the circular nature of the parameter, like

$$l(\widehat{\theta},\theta):=\sin(\widehat{\theta}-\theta)^2\ . \tag{61}$$

The gradient with respect to $\boldsymbol{\lambda}$, when the correction is implemented, is

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}} l(\widehat{\theta},\theta) = 2\sin(\widehat{\theta}-\theta)\cos(\widehat{\theta}-\theta)\frac{\mathrm{d}\widehat{\theta}}{\mathrm{d}\boldsymbol{\lambda}} \tag{62}$$

$$= \sin(2\widehat{\theta}-2\theta)\sum_{j=1}^{N}\frac{\mathrm{d}\widetilde{w}'_j}{\mathrm{d}\boldsymbol{\lambda}}\theta'_j \tag{63}$$

$$= \sin(2\widehat{\theta}-2\theta)\sum_{j=1}^{N}\left(\frac{dw'_j}{\mathrm{d}\boldsymbol{\lambda}}\theta'_j + w'_j\theta'_j\frac{\mathrm{d}\log q_{\phi(j)}}{\mathrm{d}\boldsymbol{\lambda}}\right) , \tag{64}$$

so that the likelihood term in the gradient is

$$\sin(2\widehat{\theta}-2\theta)\sum_{j=1}^{N} w'_j\theta'_j\frac{\mathrm{d}\log q_{\phi(j)}}{\mathrm{d}\boldsymbol{\lambda}} , \tag{65}$$

while it should be

$$\sin(\widehat{\theta}-\theta)^2\sum_{j=1}^{N}\frac{\mathrm{d}\log q_{\phi(j)}}{\mathrm{d}\boldsymbol{\lambda}} . \tag{66}$$

Another example where the correction fails in the loss of Eq. (8). Let us take $\widehat{\boldsymbol{\theta}}$ to be the maximum likelihood estimator, then, since a small perturbation in the posterior distribution won't change it we have

$$\frac{\mathrm{d}\widehat{\boldsymbol{\theta}}}{\mathrm{d}\boldsymbol{\lambda}} = \frac{\mathrm{d}\widehat{\boldsymbol{\theta}}}{\mathrm{d}w_j}\frac{\mathrm{d}w_j}{\mathrm{d}\lambda} = 0 , \tag{67}$$

therefore the correction is useless to backpropagate the gradient through the resampling step and we must rely only on the importance sampling. Incidentally we notice that the loss for Eq. (8) is a pure-likelihood expression, analogous to the loss in regular Policy Gradient RL.

D Computation and differentiation of the loss function

As in most optimization problems, the trainable variables of the agent are updated with a version of the stochastic gradient descent. In this section, we define the loss function for this training, compute its gradient, and comment on the computational resources required by the training.

D.1 Definition of the loss function

The two scalar losses that we used in this work are the MSE, defined in Eq. (7), used for continuous parameters, and the discrimination loss of Eq. (8), used for discrete parameters, that converges to the error probability when averaged. If the parameter to be estimated is a phase we might want to take as loss the circular variance [109]. In the following section we adopt the symbol $\ell(\widehat{\boldsymbol{\theta}},\boldsymbol{\theta})$ for the loss and keep the discussion completely general. We mention that this analysis would apply also to a more general class of losses, being functions of the of the PF ensemble, i.e. $\ell(\mathfrak{p},\boldsymbol{\theta})$, provided they are well-behaved as functions. The expected value of the loss on the trajectory is

$$\Delta^2\widehat{\boldsymbol{\theta}}_\tau := \int \ell(\widehat{\boldsymbol{\theta}},\boldsymbol{\theta})P(\widehat{\boldsymbol{\theta}}|\tau_{M-1},\boldsymbol{\theta})\,\mathrm{d}\widehat{\boldsymbol{\theta}} , \tag{68}$$

with $\tau_{M-1} := (\boldsymbol{x}_{M-1},\boldsymbol{y}_{M-1})$ indicating the complete trajectory. This definition presumes a stochastic dependence of the estimator $\widehat{\boldsymbol{\theta}}$ computed from the PF on the outcomes and the controls of the measurements, collectively denominated τ_{M-1}. This is codified by the probability density $P(\widehat{\boldsymbol{\theta}}|\tau_{M-1},\boldsymbol{\theta})$. This stochasticity can be due to the resampling routine or, in general, to the construction of the estimator $\widehat{\boldsymbol{\theta}}$, which could entail the sampling from a distribution, which is however never the case in our examples. The quantity $\Delta^2\widehat{\boldsymbol{\theta}}_\tau$ refers to a single trajectory of the PF, the one indicated with τ_{M-1}. We wish however to consider the average of the MSE over all the possible trajectories τ_{M-1} weighted appropriately. The expectation value over τ_{M-1} is expressed by the following operator

$$\mathbb{E}_\tau[\cdot] := \int \cdot\, P(\boldsymbol{x}_{M-1},\boldsymbol{y}_{M-1}|\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{x}_{M-1}\,\mathrm{d}\boldsymbol{y}_{M-1} , \tag{69}$$

which applied to Eq. (68) gives

$$\mathbb{E}_\tau\left[\Delta^2\widehat{\boldsymbol{\theta}}_\tau\right] = \int \ell(\widehat{\boldsymbol{\theta}},\boldsymbol{\theta})P(\widehat{\boldsymbol{\theta}}|\boldsymbol{\theta})\,\mathrm{d}\widehat{\boldsymbol{\theta}}\,, \tag{70}$$

where we have defined

$$P(\widehat{\boldsymbol{\theta}}|\boldsymbol{\theta}) := \int P(\widehat{\boldsymbol{\theta}}|\tau_{M-1},\boldsymbol{\theta})P(\tau_{M-1}|\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{x}_{M-1}\,\mathrm{d}\boldsymbol{y}_{M-1}\,. \tag{71}$$

We also want to take the expectation value of $\widehat{\boldsymbol{\theta}}$ on the prior $\pi(\boldsymbol{\theta})$ through the operator

$$\mathbb{E}_{\boldsymbol{\theta}}[\cdot] := \int \cdot\,\pi(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}\,, \tag{72}$$

which applied to $\mathbb{E}_\tau\left[\Delta^2\widehat{\boldsymbol{\theta}}_\tau\right]$ gives the figure of merit for the error

$$\Delta^2\widehat{\boldsymbol{\theta}} := \mathbb{E}_{\boldsymbol{\theta}}\left[\mathbb{E}_\tau\left[\Delta^2\widehat{\boldsymbol{\theta}}_\tau\right]\right] = \int \ell(\widehat{\boldsymbol{\theta}},\boldsymbol{\theta})P(\widehat{\boldsymbol{\theta}})\,\mathrm{d}\widehat{\boldsymbol{\theta}}\,, \tag{73}$$

with

$$P(\widehat{\boldsymbol{\theta}}) := \int P(\widehat{\boldsymbol{\theta}}|\tau_{M-1})P(\tau_{M-1}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{x}_{M-1}\,\mathrm{d}\boldsymbol{y}_{M-1}\,\mathrm{d}\boldsymbol{\theta}\,. \tag{74}$$

This is the probability density for the final estimator $\widehat{\boldsymbol{\theta}}$, given that the true value $\boldsymbol{\theta}$ is extracted from the prior $\pi(\boldsymbol{\theta})$ at the beginning and we average over the trajectory τ_{M-1} that is stochastically generated in the simulation, through the actions of the agent and the measurements. The expression in Eq. (73) suggests us a straightforward way to approximate the error from the numerical simulation (*à la* Monte Carlo), i.e.

$$\Delta^2\widehat{\boldsymbol{\theta}} \simeq \frac{1}{B}\sum_{k=1}^{B}\ell(\widehat{\boldsymbol{\theta}}_k,\boldsymbol{\theta}_k)\,, \tag{75}$$

where $\boldsymbol{\theta}_k$ is the true value of the parameters in the k-th simulation and $\widehat{\boldsymbol{\theta}}_k$ is the corresponding final estimator. By carrying out the complete estimation in a batch of B simulated experiments, with each $\boldsymbol{\theta}_k$ extracted from $\pi(\boldsymbol{\theta})$, we are effectively sampling $\widehat{\boldsymbol{\theta}}$ from $P(\widehat{\boldsymbol{\theta}})$ so that by the law of large number we can approximate the expectation value of the loss function in Eq. (73) with the empirical mean on the batch. Each simulation in the batch follows its particular trajectory, which will be different from the ones of the other simulations, because the randomly extracted measurement outcomes are different. Notice that in distinction with the notation of the previous sections the subscript in Eq. (75) doesn't refer to the step of the measurement cycle, but to the index of the simulation in the batch: the estimators $\widehat{\boldsymbol{\theta}}_k$ are always evaluated at the last step $t = M-1$. We call B the *batchsize* of the simulation. The right-hand side of Eq. (75) will be the loss to be minimized by the training procedure. A natural question that arises here, is why aren't we using the covariance matrix as estimated from the PF in the computation of the MSE? The answer is that the PF may be imprecise for the evaluation of the variance, in particular, it tends to underestimate it, because some tails of the distribution $P(\widehat{\boldsymbol{\theta}}|\tau_{M-1})$ may not be very well represented. We prefer to estimate the MSE empirically from the sampled $\widehat{\boldsymbol{\theta}}_k$, extracted from the true distribution $P(\widehat{\boldsymbol{\theta}})$, in order to avoid biases. The loss of Eq. (75) is the closest it can be to the precision we would observe in an experiment.

Definition of the loss for limited resources

In the previous paragraph we have implicitly assumed that the stopping condition of the estimation was based on the number of measurement M, i.e. we had a fixed number of measurement in each instance of the estimation. If, however, the resources are not simply related to the number of measurement steps, since each estimation in the batch follows its own trajectory, we may have different termination times, which correspond to the sensor employing a different number of measurement steps to consume all the available resources. In this section we introduce the notation $\widehat{\boldsymbol{\theta}}_{k,t}$, where the first subscript k is the index in the batch, and the second t is the measurement step. Whatever the nature of the resource chosen, to avoid having infinite loops we always fix a maximum number of measurement steps M in the simulation, that should to be much larger than the expected number of iterations before the resources run out. At each step only the PF ensemble of those

estimations which haven't terminated yet are updated with the Bayes rule, all the others, which have already consumed the available amount of resources, remain "freezed", since no measurement is performed and therefore no update is applied. Nevertheless all the quantities computed from the PF ensemble, e.g. $\widehat{\boldsymbol{\theta}}_{k,t}$ and $\Sigma_{k,t}$ are defined potentially for all the estimation steps $t = 0, \cdots, M-1$. To put it simply if $t_k^\star$ was the index of the last measurement for the k-th estimation in the batch before it running out of resources, then $\widehat{\boldsymbol{\theta}}_{k,t} = \widehat{\boldsymbol{\theta}}_{t_k^\star,k}$, $\Sigma_{k,t} = \Sigma_{t_k^\star,k}$ for $t \geq t_k^\star$. In general, the PF ensemble remains the same if no new measurement outcomes are incorporated, i.e. $\mathfrak{p}_{k,t} = \mathfrak{p}_{t_k^\star,k}$ for $t \geq t_k^\star$. The simplest stopping condition for the measurement cycle is now that all the B estimations in the batch have concluded, but to reduce the simulation time we only ask for at least a fraction $\nu = 0.98$ of estimations to have terminated. These would exclude those simulations that are taking too long to terminate. We define M' the realized number of iterations in the measurement loop determined by this condition, so that the loss in Eq. (75) becomes

$$\Delta^2\widehat{\boldsymbol{\theta}} \simeq \frac{1}{B'}\sum_{k=1}^{B'} \ell(\widehat{\boldsymbol{\theta}}_{k,M'}, \boldsymbol{\theta}_k) \,, \tag{76}$$

where the summation is taken only on those $B' = \lceil \nu B \rceil$ estimations in the batch that have terminated.

D.2 Dependence of the loss on the trainable variables

We go on by deriving from Eq. (73) an expression for the MSE, that is more directly related to the quantities simulated, under the hypotheses that the resampling has been turned off, i.e. $r_t = 0$, and that the computation of $\widehat{\boldsymbol{\theta}}$ from the ensemble of the PF doesn't require any stochastic operation. These are working hypotheses, which will allow to make useful observations and generalizations, whose domain of applications is however not limited by the said hypotheses. We begin observing that the controls $\boldsymbol{x}_{M-1}$ produced by the agent are deterministic functions of the ensemble of the PF, for example through the mean and the covariance matrix. Therefore, the weights of the PF are in turn deterministic functions of the measurement outcomes, as they are computed with Eq. (20), so that we can write the controls $\boldsymbol{x}_t$ and the estimator $\widehat{\boldsymbol{\theta}}_t$ at step t as

$$\boldsymbol{x}_t = g_1(\boldsymbol{y}_{t-1}, \boldsymbol{\lambda}) \quad \text{and} \quad \widehat{\boldsymbol{\theta}}_t = g_2(\boldsymbol{y}_t, \boldsymbol{\lambda}) \,, \tag{77}$$

for two appropriate functions g_1 and g_2. Beside the outcomes both the controls and the estimators depend on the trainable variables of the agent, indicated with $\boldsymbol{\lambda}$, for the aforementioned reasons. Under these hypotheses the probabilities appearing in Eq. (74) can be rewritten as

$$P(\boldsymbol{x}_{M-1}, \boldsymbol{y}_{M-1}|\boldsymbol{\theta}) = \delta(\boldsymbol{x}_{M-1} - g_1(\boldsymbol{y}_{M-2}, \boldsymbol{\lambda}))p(\boldsymbol{y}_{M-1}|\boldsymbol{\theta}, \boldsymbol{\lambda}) \,, \tag{78}$$

$$P(\widehat{\boldsymbol{\theta}}|\boldsymbol{x}_{M-1}, \boldsymbol{y}_{M-1}) = \delta(\widehat{\boldsymbol{\theta}} - g_2(\boldsymbol{y}_{M-1}, \boldsymbol{\lambda})) \,. \tag{79}$$

Solving the integrals in $\mathrm{d}\boldsymbol{x}_{M-1}$ and in $\mathrm{d}\widehat{\boldsymbol{\theta}}$ in Eq. (73), we get the following expression for the MSE

$$\Delta^2\widehat{\boldsymbol{\theta}} = \int \ell(\widehat{\boldsymbol{\theta}}(\boldsymbol{y}_{M-1}, \boldsymbol{\lambda}), \boldsymbol{\theta}) p(\boldsymbol{y}_{M-1}|\boldsymbol{\theta}, \boldsymbol{\lambda}) \pi(\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{y}_{M-1} \, \mathrm{d}\boldsymbol{\theta} \,. \tag{80}$$

This is an expectation value on the probability distribution of the tuple of outcomes $\boldsymbol{y}_{M-1}$. We introduce $\omega := (\boldsymbol{y}_{M-1}, \boldsymbol{\theta})$ and redefine the loss for the next sections as

$$\ell(\omega, \boldsymbol{\lambda}) := \ell(\widehat{\boldsymbol{\theta}}(\boldsymbol{y}_{M-1}, \boldsymbol{\lambda}), \boldsymbol{\lambda}) \,. \tag{81}$$

The object ω contains all the variables that depend on the specific instance of the simulation so that the empirical approximation of $\Delta^2\widehat{\boldsymbol{\theta}}$ from Eq. (80) is

$$\mathcal{L}(\boldsymbol{\lambda}) := \frac{1}{B}\sum_{k=1}^{B} \ell(\omega_k, \boldsymbol{\lambda}) \,, \tag{82}$$

with $\omega_k := (\boldsymbol{y}_{k,M-1}, \boldsymbol{\theta}_k)$. The true values $\boldsymbol{\theta}_k$ are sampled from $\pi(\boldsymbol{\theta})$ at the beginning of the run. In case the agent is a NN the trainable variables are the weights and the biases, while for the non-adaptive strategy the variables are directly the tuple of all the controls, i.e. $\boldsymbol{\lambda} = (x_1, x_2, \ldots, x_{M-1})$. The average loss in Eq. (82) will be also named the scalar loss, in contrast to $\ell(\omega_k, \boldsymbol{\lambda})$, which is the individual loss or the vector loss, since it has a free index k.

D.3 Gradient of the loss

The simulation of the quantum sensor, the particle filter, and the evaluation of the NN are implemented in the chosen automatic differentiation (AD) environment, i.e. TensorFlow (TF), so that at the end of the simulation we can take the gradient of the loss in Eq. (82) with respect to $\boldsymbol{\lambda}$ with no effort and obtain

$$\frac{\mathrm{d}\mathcal{L}(\boldsymbol{\lambda})}{\mathrm{d}\boldsymbol{\lambda}} = \frac{1}{B}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}\sum_{k=1}^{B}\ell(\omega_k,\boldsymbol{\lambda})\ . \tag{83}$$

The automatic differentiation framework does all the derivatives automatically, that we would need to evaluate analytically or numerically otherwise. Even if the outcomes $\boldsymbol{y}_{k,M-1}$ are extracted from a probability distribution that depends on $\boldsymbol{\lambda}$, as it is because each of them is sampled from $p(y_{k,t+1}|\boldsymbol{x}_{k,t+1},\boldsymbol{y}_t,\boldsymbol{\theta}_k)$ and the controls $\boldsymbol{x}_{k,t+1}$ depend on $\boldsymbol{\lambda}$, in TF and other similar frameworks their derivatives with respect to $\boldsymbol{\lambda}$ are always null by construction, i.e.

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}y_{k,t+1} = 0\ , \tag{84}$$

in other words, the gradient cannot propagate through the extraction of random variables. This is a consistent behaviour of automatic differentiation frameworks, and has to do with the fact that the sampled variables are considered constant tensors in the construction of the graph, on the same level as other numerical constants fixed by the programmer. We will show now, that much like in [8], the gradient of the loss produced by AD in Eq. (83) is not correct and will lead to a suboptimal training routine. Another term must be added that keeps track of the sampled variables during the evolution. To understand why this is so, let us start from the theoretical definition of $\Delta^2\widehat{\boldsymbol{\theta}}$ in Eq. (80) and take its gradient with respect to $\boldsymbol{\lambda}$. The two terms $p(\boldsymbol{y}_{M-1}|\boldsymbol{\theta},\boldsymbol{\lambda})$ and $\widehat{\boldsymbol{\theta}}(\boldsymbol{y}_{M-1},\boldsymbol{\lambda})$ both depend on $\boldsymbol{\lambda}$. The first one can be expanded as follows:

$$p(\boldsymbol{y}_{M-1}|\boldsymbol{\theta},\boldsymbol{\lambda}) = \prod_{t=0}^{M-1}p(y_t|\boldsymbol{x}_t,\boldsymbol{y}_{t-1},\boldsymbol{\theta})\ , \tag{85}$$

and the dependence on the controls $\boldsymbol{x}_t$ is a dependence on the trainable variables of the agent $\boldsymbol{\lambda}$. The second term $\widehat{\boldsymbol{\theta}}(\boldsymbol{y}_{M-1},\boldsymbol{\lambda})$ depends on $\boldsymbol{\lambda}$ through the PF weights, which are updated with the Bayes rule Eq. (20), that features the term $p(y_{t+1}|\boldsymbol{x}_{t+1},\boldsymbol{y}_t,\boldsymbol{\theta})$, where again the controls $\boldsymbol{x}_{t+1}$ are $\boldsymbol{\lambda}$-dependent. The complete gradient of the right-hand term of Eq. (80) reads therefore

$$\int\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}\ell(\widehat{\boldsymbol{\theta}}(\boldsymbol{y}_{M-1},\boldsymbol{\lambda}),\boldsymbol{\theta})p(\boldsymbol{y}_{M-1}|\boldsymbol{\theta},\boldsymbol{\lambda})\,\mathrm{d}\boldsymbol{y}_{M-1} + \int\ell(\widehat{\boldsymbol{\theta}}(\boldsymbol{y}_{M-1},\boldsymbol{\lambda}),\boldsymbol{\theta})\frac{\mathrm{d}p(\boldsymbol{y}_{M-1}|\boldsymbol{\theta},\boldsymbol{\lambda})}{\mathrm{d}\boldsymbol{\lambda}}\,\mathrm{d}\boldsymbol{y}_{M-1}\ . \tag{86}$$

The first term is in the form of an expectation value and can be straightforwardly approximated in a Monte Carlo simulation. It corresponds exactly to the naïve gradient of the loss in Eq. (83) computed by the AD framework. The second term can be written as

$$\int\ell(\widehat{\boldsymbol{\theta}}(\boldsymbol{y}_{M-1},\boldsymbol{\lambda}),\boldsymbol{\theta})\frac{\mathrm{d}\log p(\boldsymbol{y}_{M-1}|\boldsymbol{\theta},\boldsymbol{\lambda})}{\mathrm{d}\boldsymbol{\lambda}}p(\boldsymbol{y}_{M-1}|\boldsymbol{\theta},\boldsymbol{\lambda})\,\mathrm{d}\boldsymbol{y}_{M-1}\ , \tag{87}$$

which is now in the form of an expectation value on the trajectories of the simulation and can be evaluated simultaneously with the first term, provided we keep track of $\log p(\boldsymbol{y}_{M-1}|\boldsymbol{\theta},\boldsymbol{\lambda})$. This second contribution to the gradient can be approximated as

$$\frac{1}{B}\sum_{k=1}^{B}\ell(\omega_k,\boldsymbol{\lambda})\frac{\mathrm{d}\log p(\boldsymbol{y}_{k,M-1}|\boldsymbol{\theta}_k,\boldsymbol{\lambda})}{\mathrm{d}\boldsymbol{\lambda}}\ , \tag{88}$$

on a batch of B simulations. The term $\log p(\boldsymbol{y}_{k,M-1}|\boldsymbol{\theta}_k,\boldsymbol{\lambda})$ is the sum

$$\log p(\boldsymbol{y}_{k,M-1}|\boldsymbol{\theta}_k,\boldsymbol{\lambda}) = \sum_{t=0}^{M-1}\log p(y_{k,t}|\boldsymbol{y}_{k,t-1},\boldsymbol{\theta}_k,\boldsymbol{\lambda})\ , \tag{89}$$

where we exchanged the dependence on $\boldsymbol{x}_{k,t}$ of the factors $p(y_{k,t}|\boldsymbol{x}_{k,t},\boldsymbol{y}_{k,t-1},\boldsymbol{\theta}_k)$ for the dependence on $\boldsymbol{\lambda}$. This logarithm can be accumulated step by step in the simulation, after the extraction of each measurement

outcome. Notice that for B simulations in the batch, we have to compute B cumulated probabilities, because each trajectory is different. In conclusion, the total gradient is

$$\frac{1}{B}\sum_{k=1}^{B}\left[\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}\ell(\omega_k,\boldsymbol{\lambda})+\ell(\omega_k,\boldsymbol{\lambda})\frac{\mathrm{d}\log p(\boldsymbol{y}_{k,M-1}|\boldsymbol{\theta}_k,\boldsymbol{\lambda})}{\mathrm{d}\boldsymbol{\lambda}}\right] . \tag{90}$$

By introducing the *stop gradient* operation we can write it in the convenient form

$$\frac{1}{B}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}\sum_{k=1}^{B}\{\ell(\omega_k,\boldsymbol{\lambda})+\mathrm{sg}\left[\ell(\omega_k,\boldsymbol{\lambda}),\boldsymbol{\theta}\right]\log p(\boldsymbol{y}_{k,M-1}|\boldsymbol{\theta}_k,\boldsymbol{\lambda})\} , \tag{91}$$

that requires only one gradient, which makes it more straightforward to implement in the AD framework. In this formula we are using the stop gradient operator $\mathrm{sg}\left[\cdot\right]$, which is an instruction that tells the automatic differentiation frameworks not to compute the derivatives of the expression inside the operator. We are therefore naturally led to introducing the modified loss $\widetilde{\mathcal{L}}(\boldsymbol{\lambda})$, i.e.

$$\widetilde{\mathcal{L}}(\boldsymbol{\lambda}) := \frac{1}{B}\sum_{k=1}^{B}\widetilde{\ell}(\omega_k,\boldsymbol{\lambda}) , \tag{92}$$

with

$$\widetilde{\ell}(\omega_k,\boldsymbol{\lambda}) := \ell(\omega_k,\boldsymbol{\lambda})+\mathrm{sg}\left[\ell(\omega_k,\boldsymbol{\lambda})\right]\log p(\boldsymbol{y}_{k,M-1}|\boldsymbol{\theta}_k,\boldsymbol{\lambda}) . \tag{93}$$

which is the correct function to be minimized. For a resource limited estimation the modified loss is the average of the B' simulations that have terminated. In Appendix E.1 we comment on the gradient descent process, on the choice of the hyper-parameters, and the typical behaviour of the loss in the training. The second term of the gradient in Eq. (91) is similar to the loss of the Policy Gradient method in reinforcement learning [110], where the probabilities arise because of the stochastic extraction of the policy, while here the NN produces directly the action and the stochasticity comes from the measurement outcome extraction. The necessity of introducing such terms when dealing with the gradient of expressions involving non-reparametrizable random variables has been known in the machine learning literature for a while [111] and the expressions involving stop-gradient operators go under the name of surrogate expressions. However, the first time this has appeared in the physics literature is in [8], applied to quantum feedback. Had we neglected the log-likelihood term of Eq. (91) we would have introduced a bias in the gradient. Not adding the log-likelihood terms means not only a slower convergence in the training but possibly also converging to a worse minimum.

Log-likelihood terms in the loss

We can generalize and say that whatever extraction of random variables we perform during the simulation, we need to add a corresponding log-likelihood term, but only if the probability distribution from which they have been extracted depends on $\boldsymbol{\lambda}$, implicitly or explicitly. With growing number of extracted variables the variance of the gradient grows and if the batchsize is too small this can severely affect the training, then we might loose convergence and end up in a bad local minimum. If possible we advice to reparametrize the random variables and account for the backpropagation of the gradient in a more direct way. Depending on the quantum sensor we are simulating we might be able to implement the extraction of the measurement outcomes through a differentiable reparametrization. If we can write the measurement outcome y_t as

$$y_t = g(u_t, x_t, \boldsymbol{\theta}) , \tag{94}$$

where u_t is a random variable extracted from a probability distribution independent on $\boldsymbol{\lambda}$, i.e. $\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}p(u_t)=0$, then we can omit the corresponding log-likelihood term in Eq. (91). The gradient propagates now directly through the measurement outcome, i.e.

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}y_t \neq 0 , \tag{95}$$

and we can differentiate the loss in Eq. (82) as it is. The log-likelihood term would be $\log p(u_t)$, which is independent on $\boldsymbol{\lambda}$. The reparametrization can however be applied only to continuous variables, see [8] for more details. In Appendix C we apply this technique in the resampling step of the particle filter.

Adding a baseline

We can also try to add a baseline to Eq. (91), as suggested in [112] for RL with Policy Gradient. This means modifying the gradient to

$$\frac{1}{B}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}\sum_{k=1}^{B}\left[\ell(\omega_k,\boldsymbol{\lambda})+\mathrm{sg}\left[\ell(\omega_k,\boldsymbol{\lambda})-\mathcal{B}\right]\log p(\boldsymbol{y}_{k,M-1}|\boldsymbol{\theta}_k,\boldsymbol{\lambda})\right]\ , \tag{96}$$

with the standard choice for the baseline $\mathcal{B}$ being

$$\mathcal{B}:=\frac{1}{B}\sum_{k=1}^{B}\ell(\omega_k,\boldsymbol{\lambda})\ , \tag{97}$$

that is, inside the stop gradient we subtract to each loss in the batch the mean value of the loss on the batch. It is important for $\mathcal{B}$ to be a constant across the simulations indexed by k. We briefly see in the following that the introduction of $\mathcal{B}$ doesn't change the expected value gradient, while it can be proved that it reduces the variance of the gradient [112]. Consider the following chain of equalities

$$0=\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}\int p(\boldsymbol{y}_{M-1}|\boldsymbol{\theta},\boldsymbol{\lambda})\,\mathrm{d}\boldsymbol{y}_{M-1}=\int\frac{1}{p(\boldsymbol{y}_{M-1}|\boldsymbol{\theta},\boldsymbol{\lambda})}\frac{\mathrm{d}p(\boldsymbol{y}_{M-1}|\boldsymbol{\theta},\boldsymbol{\lambda})}{\mathrm{d}\boldsymbol{\lambda}}p(\boldsymbol{y}_{M-1}|\boldsymbol{\theta},\boldsymbol{\lambda})\,\mathrm{d}\boldsymbol{y}_{M-1}\ , \tag{98}$$

where the first one comes from the normalization of $p(\boldsymbol{y}_{M-1}|\boldsymbol{\theta}_k,\boldsymbol{\lambda})$ and in the second we divided and multiplied for $p(\boldsymbol{y}_{M-1}|\boldsymbol{\theta}_k,\boldsymbol{\lambda})$ upon swapping the integral and the derivate. The rightmost term of Eq. (98) is now in the form of an expectation value, that can be approximated by the following summation in the simulation:

$$\int\frac{1}{p(\boldsymbol{y}_{M-1}|\boldsymbol{\theta},\boldsymbol{\lambda})}\frac{\mathrm{d}p(\boldsymbol{y}_{M-1}|\boldsymbol{\theta},\boldsymbol{\lambda})}{\mathrm{d}\boldsymbol{\lambda}}p(\boldsymbol{y}_{M-1}|\boldsymbol{\theta},\boldsymbol{\lambda})\,\mathrm{d}\boldsymbol{y}_{M-1}\simeq\frac{1}{B}\sum_{k=1}^{B}\frac{1}{p(\boldsymbol{y}_{k,M-1}|\boldsymbol{\theta}_k,\boldsymbol{\lambda})}\frac{\mathrm{d}p(\boldsymbol{y}_{k,M-1}|\boldsymbol{\theta}_k,\boldsymbol{\lambda})}{\mathrm{d}\boldsymbol{\lambda}}\ . \tag{99}$$

Where the term in the right-hand summation is the derivative of the log-likelihood, so that we expect

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}\sum_{k=1}^{B}\log p(\boldsymbol{y}_{k,M-1}|\boldsymbol{\theta}_k,\boldsymbol{\lambda})\simeq 0\ , \tag{100}$$

for large B. Adding the baseline in Eq. (97), means adding a terms proportional to the derivative of the log-likelihood, with the proportionality constant being $\mathcal{B}$, which has null expectation value.

Discrete control space

In this section we briefly comment on the case in which the control space is discrete, that is, x_t can be chosen only among finitely many elements, i.e $x_t\in\chi=\{x_1,x_2,\dots,x_R\}$. This happens for example in the experiment presented in [113], where the control parameter was the topological charge of the q-plate. In this case the agent produces a probability distribution on the set χ as outcome, just like in Policy Learning, and a random x_t is extracted from this categorical distribution. In this scenario we need to revisit the Eq. (78) and Eq. (79), that now need to accommodate also for the probability of extracting a particular x_t:

$$P(\boldsymbol{x}_{M-1},\boldsymbol{y}_{M-1}|\boldsymbol{\theta})=\prod_{t=0}^{M-1}p(y_t|\boldsymbol{x}_t,\boldsymbol{y}_{t-1},\boldsymbol{\theta})g(x_t|\boldsymbol{y}_{t-1},\boldsymbol{x}_{t-1},\boldsymbol{\lambda}) \tag{101}$$

$$P(\widehat{\boldsymbol{\theta}}|\boldsymbol{x}_{M-1},\boldsymbol{y}_{M-1})=\delta(\widehat{\boldsymbol{\theta}}-g_2(\boldsymbol{y}_{M-1},\boldsymbol{x}_{M-1}))\ . \tag{102}$$

Substituting this expressions in Eq. (73) we get

$$\Delta^2\widehat{\boldsymbol{\theta}}=\int\ell(\widehat{\boldsymbol{\theta}}(\boldsymbol{y}_{M-1},\boldsymbol{x}_{M-1}),\boldsymbol{\theta})\prod_{t=0}^{M-1}p(y_t|\boldsymbol{x}_t,\boldsymbol{y}_{t-1},\boldsymbol{\theta})g(x_t|\boldsymbol{y}_{t-1},\boldsymbol{x}_{t-1},\boldsymbol{\lambda})\,\mathrm{d}\boldsymbol{x}_{M-1}\,\mathrm{d}\boldsymbol{y}_{M-1}\ . \tag{103}$$

By repeating the derivation of the loss, considering that $p(y_t|\boldsymbol{x}_t,\boldsymbol{y}_{t-1},\boldsymbol{\theta})$ doesn't depend on $\boldsymbol{\lambda}$ anymore, the log-likelihood term of Eq. (91) becomes

$$\sum_{t=0}^{M-1}\mathrm{sg}\left[\ell(\omega_k,\boldsymbol{\lambda})\right]\log g(x_{k,t}|\boldsymbol{y}_{t-1,k},\boldsymbol{x}_{t-1,k},\boldsymbol{\lambda})\ . \tag{104}$$

Stochastic estimator

In all the applications of this work the estimator $\widehat{\boldsymbol{\theta}}$ is always a deterministic function of the PF ensemble. Even for the case of values discrimination, this is computed as the most likely hypothesis at the end of the experiment, which doesn't require any sampling. A perfectly valid estimator for θ would be one sample extracted from the Bayesian posterior $P(\boldsymbol{\theta}|\boldsymbol{y}_{M-1}, \boldsymbol{\lambda})$. Doing so requires adding a term to the log-likelihood term in the loss in Eq. (91), which now becomes

$$\sum_{t=0}^{M-1} \mathrm{sg}\left[\ell(\omega_k, \boldsymbol{\lambda})\right] \left[\log p(\boldsymbol{y}_{k,M-1}|\boldsymbol{\theta}_k, \boldsymbol{\lambda}) + \log P(\widehat{\boldsymbol{\theta}}_k|\boldsymbol{y}_{k,M-1}, \boldsymbol{\lambda})\right] . \tag{105}$$

A differentiable expression for the posterior distribution at $\widehat{\boldsymbol{\theta}}$ might not be accessible if the parameters are continuous, but if they are discrete, the term $P(\widehat{\boldsymbol{\theta}}|\boldsymbol{y}_{M-1}, \boldsymbol{\lambda})$ is just the weight corresponding to the discrete values $\widehat{\boldsymbol{\theta}}$.

D.4 Definition of the cumulative and logarithmic losses

When doing a simulation for a certain M, if we want the result of the training to give us the optimal strategy also for $M_2 < M$ we can introduce the cumulative loss, that also takes into account the loss at intermediate steps. A naïve approach is to extend the MSE to all steps between $t = 0$ and $t = M - 1$, and write

$$\mathcal{L}_{\mathrm{cum}}(\boldsymbol{\lambda}) := \frac{1}{MB} \sum_{t=0}^{M-1} \sum_{k=1}^{B} \ell(\widehat{\boldsymbol{\theta}}_{k,t}, \boldsymbol{\theta}_k) , \tag{106}$$

where $\widehat{\boldsymbol{\theta}}_{k,t}$ is the estimator at step t of the k-th simulation. With this loss the agent is incentivized to make the estimator $\widehat{\boldsymbol{\theta}}$ converge to the value $\boldsymbol{\theta}$ as soon as possible. However the error on the first time steps of the estimation dominates the later errors in the summation, and this puts pressure on the agent to optimize the first steps of the procedure at the expense of the later precision. To solve this problem we divide each terms in the sum Eq. (106) by a function $\eta(\boldsymbol{\theta}, t)$, i.e.

$$\mathcal{L}_{\mathrm{cum}}(\boldsymbol{\lambda}) := \frac{1}{MB} \sum_{t=0}^{M-1} \sum_{k=1}^{B} \frac{\ell(\widehat{\boldsymbol{\theta}}_{k,t}, \boldsymbol{\theta}_k)}{\eta(\boldsymbol{\theta}_k, t)} , \tag{107}$$

where $\eta(\boldsymbol{\theta}_k, t)$ is the expected precision of the estimation at step t given the true value $\boldsymbol{\theta}_k$, or an approximation to it, in the form of a lower bound for example, like the Cramér-Rao bound. This new loss measures the relative variation of the error from the reference value. Even if $\eta(\boldsymbol{\theta}_k, t)$ is a rigorous lower bound on the MSE we can't expect the inequality

$$\ell(\widehat{\boldsymbol{\theta}}_{k,t}, \boldsymbol{\theta}_k) \geq \eta(\boldsymbol{\theta}_k, t) , \tag{108}$$

to hold exactly for every t and k, as there will be fluctuations due to the finite batchsize. From the practical point of view this means that it is possible for the loss of some training steps to be $\mathcal{L}(\boldsymbol{\lambda}) < 1$, which doesn't necessarily point toward a bug in the implementation of the training. With Eq. (107) we still incentivise the agent to be as fast as possible in reaching a good precision, and not wait until the end, because then it will be rewarded by the reduced loss for all the duration of the experiment. Another possibility to account fairly for the MSE at intermediate times is to take the logarithm of the mean error on the batch and write the cumulative loss as

$$\mathcal{L}_{\log}(\boldsymbol{\lambda}) := \frac{1}{M} \sum_{t=0}^{M-1} \log\left[\frac{1}{B} \sum_{k=1}^{B} \ell(\widehat{\boldsymbol{\theta}}_{k,t}, \boldsymbol{\theta}_k)\right] . \tag{109}$$

The advantage of this approach is that it doesn't require any prior known reference value for the error. Notice that this loss is not in the form of an expectation value of $\ell(\widehat{\boldsymbol{\theta}}, \boldsymbol{\theta})$ over a batch.

D.5 Cumulative and logarithmic losses for a resource limited estimation

In this section we comment on the form taken by the cumulative and logarithmic losses in the case of a limited number of resources. Given M' the realized number of iterations of the measurement loop Eq. (107) becomes

$$\mathcal{L}_{\mathrm{cum}}(\boldsymbol{\lambda}) := \frac{1}{M'B} \sum_{t=0}^{M'-1} \sum_{k=1}^{B} \frac{\ell(\widehat{\boldsymbol{\theta}}_{k,t}, \boldsymbol{\theta}_k)}{\eta(\boldsymbol{\theta}_k, t)} , \tag{110}$$

notice, that at difference with Eq. (76) all the simulations in the batch are considerer, not only those B' that were already ended as the measurement loop stopped. If one estimation in the batch is ended prematurely with respect to all the other, with all the resources consumed to obtain a bad estimator for $\boldsymbol{\theta}$ this will have a huge weight in the loss, since the squared error will appear multiple times, until all the other estimations are ended. This means that an unwise use of the resources, which are consumed early to reach a poor result will be strongly penalized. One may think, that since the number of iterations M' is stochastic then teh cumulative loss is a form of "existential loss" which would put pressure on th agent to terminate with the smallest number of measurement step possible, this would be at odd with the actual goal of optimizing with fixed resources irrespective of the number of measurements, but indeed the loss is normalized according to M', so that having a short or a long cycle doesn't matter for the computation of $\mathcal{L}(\boldsymbol{\lambda})$. Similarly to the cumulative loss, the logarithmic loss for an estimation with a limited number of resources can be expressed as

$$\mathcal{L}_{\log}(\boldsymbol{\lambda}) := \frac{1}{M'} \sum_{t=0}^{M'-1} \log \left[\frac{1}{B} \sum_{k=1}^{B} \ell(\widehat{\boldsymbol{\theta}}_{k,t}, \boldsymbol{\theta}_k) \right] , \tag{111}$$

where again M' is the actual number of executed iterations of the loop.

D.6 Gradients of the cumulative and logarithmic losses

In this section we comment on the expression of the gradient of the cumulative and logarithmic losses, and of the role of the log-likelihood terms that we had inserted in Eq. (93). The modified cumulative loss, from which the AD framework can directly compute the gradient, reads

$$\widetilde{\mathcal{L}}_{\mathrm{cum}}(\boldsymbol{\lambda}) := \frac{1}{MB} \sum_{k=1}^{B} \Big\{ \sum_{t=0}^{M-1} \ell(\widehat{\boldsymbol{\theta}}_{k,t}, \boldsymbol{\theta}_k) + \mathrm{sg} \left[\sum_{t=0}^{M-1} \ell(\widehat{\boldsymbol{\theta}}_{k,t}, \boldsymbol{\theta}_k) \right] \sum_{t=0}^{M-1} \log p(y_{k,t} | \boldsymbol{\theta}_k, \boldsymbol{y}_{t-1,k}, \boldsymbol{\lambda}) \Big\} . \tag{112}$$

Given that the stop gradient operator is linear, we now make the important observation that the gradient of the log-likelihood terms in the form

$$\mathrm{sg} \left[\ell(\widehat{\boldsymbol{\theta}}_{\alpha,k}, \boldsymbol{\theta}_k) \right] \log p(y_{\beta,k} | \boldsymbol{\theta}_k, \boldsymbol{y}_{\beta-1,k}, \boldsymbol{\lambda}) . \tag{113}$$

with $\beta > \alpha$ have null expectation value on the batch of simulations, that is

$$\frac{1}{B} \sum_{k=1}^{B} \ell(\widehat{\boldsymbol{\theta}}_{\alpha,k}, \boldsymbol{\theta}_k) \frac{\mathrm{d} \log p(y_{\beta,k} | \boldsymbol{\theta}_k, \boldsymbol{y}_{\beta-1,k}, \boldsymbol{\lambda})}{\mathrm{d}\boldsymbol{\lambda}} \simeq 0 , \tag{114}$$

The expression in Eq. (114) is an approximation of the true expectation value

$$\int \ell(\widehat{\boldsymbol{\theta}}_{\alpha}, \boldsymbol{\theta}) \frac{\mathrm{d} \log p(y_{\beta} | \boldsymbol{\theta}, \boldsymbol{y}_{\beta-1}, \boldsymbol{\lambda})}{\mathrm{d}\boldsymbol{\lambda}} \pi(\boldsymbol{\theta}) d\boldsymbol{\theta} \prod_{t=0}^{\beta} p(y_t | \boldsymbol{\theta}, \boldsymbol{y}_{t-1}, \boldsymbol{\lambda}) \, \mathrm{d}y_t . \tag{115}$$

All the integral for y_t for $t > \beta$ can be simplified in the above formula, since the integrand doesn't depend on these variables. Let us first solve the integral for $\mathrm{d}y_\beta$. The loss term doesn't depend on this variable, so that we can pull it out of the integral and write

$$\ell(\widehat{\boldsymbol{\theta}}_{\alpha}, \boldsymbol{\theta}) \int \frac{\mathrm{d} \log p(y_{\beta} | \boldsymbol{\theta}_k, \boldsymbol{y}_{\beta-1}, \boldsymbol{\lambda})}{\mathrm{d}\boldsymbol{\lambda}} p(y_{\beta} | \boldsymbol{\theta}, \boldsymbol{y}_{t-1}, \boldsymbol{\lambda}) \, \mathrm{d}y_{\beta} , \tag{116}$$

which is equal to

$$\int \frac{\mathrm{d}p(y_{\beta} | \boldsymbol{\theta}, \boldsymbol{y}_{t-1}, \boldsymbol{\lambda})}{\mathrm{d}\boldsymbol{\lambda}} \, \mathrm{d}y_{\beta} = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}} \int p(y_{\beta} | \boldsymbol{\theta}, \boldsymbol{y}_{t-1}, \boldsymbol{\lambda}) \, \mathrm{d}y_{\beta} = 0 . \tag{117}$$

Since the summation Eq. (114) tends to zero for large B, we can write the loss as following

$$\widetilde{\mathcal{L}}_{\mathrm{cum}}(\boldsymbol{\lambda}) := \frac{1}{MB} \sum_{k=1}^{B} \Big\{ \sum_{t=0}^{M-1} \ell(\widehat{\boldsymbol{\theta}}_{k,t}, \boldsymbol{\theta}_k) + \sum_{t=0}^{M-1} \mathrm{sg} \left[\ell(\widehat{\boldsymbol{\theta}}_{k,t}, \boldsymbol{\theta}_k) \right] \log p(\boldsymbol{y}_{k,t} | \boldsymbol{\theta}_k, \boldsymbol{\lambda}) \Big\} . \tag{118}$$

which is the expression implemented in the library. Since we have removed some of the stochastic terms in the loss, which average to zero, but nevertheless contribute to the fluctuations, using expression Eq. (118) we

expect to have reduced the variance of the gradient, just like we did with the correction of Ścibior and Wood for the particle resampling. From this derivation we learn that in general the log-likelihood terms of variables extracted in the future with respect to the terms they multiply can be simplified. Notice that in this derivation we haven't assumed projective measurements, that would have meant $p(y_t|\boldsymbol{\theta}_k, \boldsymbol{y}_{t-1}, \boldsymbol{\lambda}) = p(y_t|\boldsymbol{\theta}_k, \boldsymbol{\lambda})$, instead our derivation works in the most general case of a weakly measured probe. We now turn to the gradient of the logarithm loss of Eq. (111). This is somewhat different from the previous cases since now the mean on the batch is inside the logarithm. The expectation value of the loss is

$$\frac{1}{M}\sum_{t=0}^{M-1}\log\left[\int \ell(\widehat{\boldsymbol{\theta}}_t, \boldsymbol{\theta}) p(\boldsymbol{y}_t|\boldsymbol{\theta}, \boldsymbol{\lambda})\,\mathrm{d}\boldsymbol{y}_t\right] , \tag{119}$$

which has the following gradient

$$\frac{1}{M}\sum_{t=0}^{M-1}\frac{\int \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}\ell(\widehat{\boldsymbol{\theta}}_t, \boldsymbol{\theta}) p(\boldsymbol{y}_t|\boldsymbol{\theta}, \boldsymbol{\lambda})\,\mathrm{d}\boldsymbol{y}_t + \int \ell(\widehat{\boldsymbol{\theta}}_t, \boldsymbol{\theta})\frac{\mathrm{d}\log p(\boldsymbol{y}_t|\boldsymbol{\theta},\boldsymbol{\lambda})}{\mathrm{d}\boldsymbol{\lambda}} p(\boldsymbol{y}_t|\boldsymbol{\theta}, \boldsymbol{\lambda})\,\mathrm{d}\boldsymbol{y}_t}{\int \ell(\widehat{\boldsymbol{\theta}}_t, \boldsymbol{\theta}) p(\boldsymbol{y}_t|\boldsymbol{\theta}, \boldsymbol{\lambda})\,\mathrm{d}y_t} . \tag{120}$$

This expression can be obtained on the batch of simulations with the modified loss

$$\widetilde{\mathcal{L}}_{\log}(\boldsymbol{\lambda}) := \frac{1}{M}\sum_{t=0}^{M-1}\frac{\sum_{k=1}^{B}\ell(\widehat{\boldsymbol{\theta}}_{k,t}, \boldsymbol{\theta}_k) + \sum_{k=1}^{B}\mathrm{sg}\left[\ell(\widehat{\boldsymbol{\theta}}_{k,t}, \boldsymbol{\theta}_k)\right]\log p(\boldsymbol{y}_{k,t}|\boldsymbol{\theta}_k, \boldsymbol{\lambda})}{\mathrm{sg}\left[\sum_{k=1}^{B}\ell(\widehat{\boldsymbol{\theta}}_{k,t}, \boldsymbol{\theta}_k)\right]} . \tag{121}$$

To get the results for the resources limited estimation we substitute M with M' in the whole section.

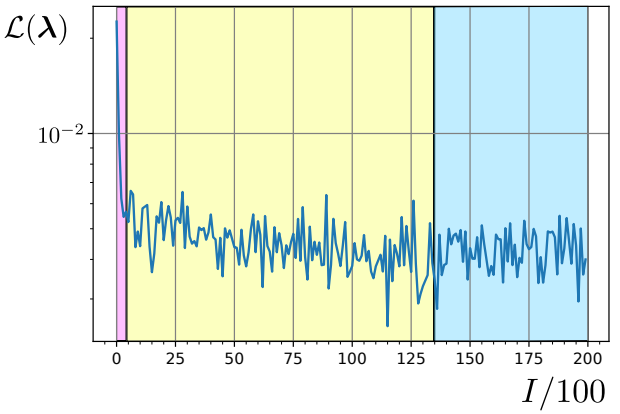
E Details on the simulations

E.1 Tuning of the hyperparameters

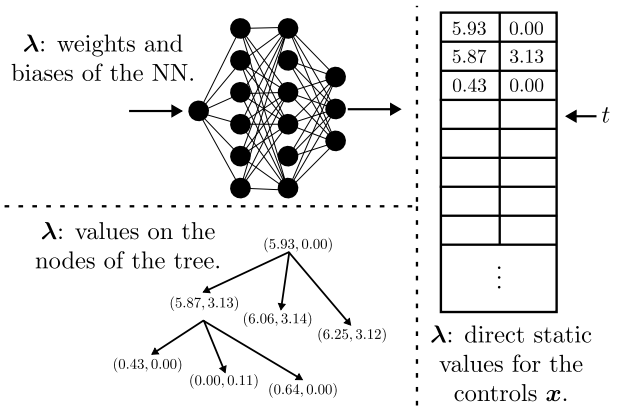
We mentioned in the main text that the update of the agent's trainable variables is not actually done through Eq. (12), but via a more sophisticated optimizator called Adam. We observed empirically that a decaying learning rate is beneficial when using the Adam optimizer. This is because the agent first learns the rough features of the optimal solution with a relatively large update step for the variables. Subsequently, with a smaller learning rate, the solution is fine tuned. The Adam optimized, however, already has an internal adaptive update step that is different for every variables, therefore the learning rate should be really only understood as a broad indication of the training speed given to the optimizer. In the original Adam paper [67] the authors consider a learning rate decaying with the inverse square root of the number of update steps. This was also our choice. Let us define $i = 1, 2, \cdots, I$ the index of the update step in the training process, then the learning rate at the i-th iteration of the gradient descent is

$$\alpha_i := \frac{\alpha_0}{\sqrt{i}} . \tag{122}$$

We observe empirically, that the initial value of the learning rate α_0 for a NN should depend on the batchsize B. For $B \sim \mathcal{O}(10^3)$ we use $\alpha_0 \sim \mathcal{O}(10^{-2})$, while for $B \sim \mathcal{O}(10^2)$ a value of $\alpha_0 \sim \mathcal{O}(10^{-3})$ is more appropriate. For the non-adaptive strategy we use an initial learning rate that is one order of magnitude larger than the one used for the NN at equal batchsize. The minimum number of training steps I depend strongly on the application, but we observed in all our examples that it should of order $I \sim \mathcal{O}(10^3 - 10^4)$ to reach convergence. We observed some universal feature in the behaviour of the loss as the training proceeds, which can be associated to three different phases in the training, see Fig. 8. First we have an initial phase of fast learning, which is the shortest one, coloured in pink, followed by the fine tuning phase in yellow and the plateau at the end, with the loss remaining on average constant. As a final note, we mention that when the resampling routine is active we might expect a slow-down of the simulation speed as the training session proceeds. This happens because as the agent is perfected and the loss is reduced, it is more probable that a resampling event is triggered (because the increasing precision means also more concentrated weights in the PF ensemble), which slows down the simulation. In other words, the amount of code that has to be executed in a run is not fixed *a priori*, but depends dynamically on the resampling condition that is checked at run-time. To end the section we briefly recap the three possible implementations that the trainable variables $\boldsymbol{\lambda}$ have taken in this work, see Fig. 8. In the case the agent is a NN $\boldsymbol{\lambda}$ are the weights and the biases of the network When the agent is a decision tree, the controls x_t are associated to each node of the tree, and they are the $\boldsymbol{\lambda}$ variables. Finally, for a non-adaptive strategy, there is no adaptivity and the controls $\boldsymbol{x}$ are codified in a list, indexed by the measurement step t. In this case the controls and the training variables coincide, i.e. $\boldsymbol{\lambda} = \boldsymbol{x}_{M-1}$.

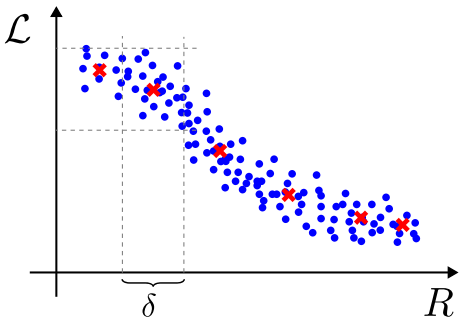


(a) Three phases of the training.

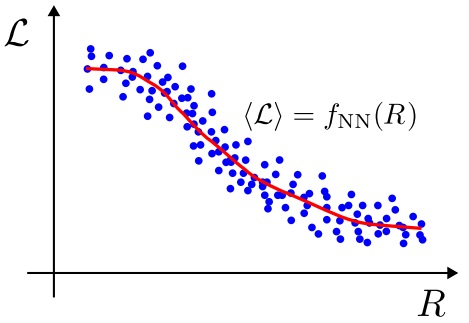


(b) Implementations of the trainable variables.

Figure 8: On the left picture the three phases of the learning process explained in Appendix E.1 are plotted. One tick on the x-axis corresponds to 500 update steps. The agent first learns the rough form of the optimal strategy and later the fine details, before converging. On the right the three agents used in this work are represented, i.e a NN, a ternary decision tree, and a table containing the values of the controls $\boldsymbol{x}$, indexed by the measurement step t.



(a) Clustering of the precision point cloud.



(b) Fit of the precision point cloud.

Figure 9: On the left side we represent the precision plot where the cloud of points has been clustered to obtain the red crosses. On the right we use a NN to interpolate and get the average loss at a given value of the resources.

E.2 Fit of the precision

In this section we briefly comment on the way the precision plot are realized throughout the work. The definition of the resources doesn't only impact the stopping condition of the measurement loop, but it defines how the performances of an agent are visualized, since by default we plot the mean loss as a function of the consumed resources. After having trained the agent we simulate many times the estimation and we keep track of the tuples $\mathcal{S} := \{(R_j, \mathcal{L}_j)\}_{j=1}^{S}$ after each measurement step, containing the consumed resources R_t and the loss $\mathcal{L}$. Since the experiment is a stochastic process we will collect a cloud of points from which a simple relation between the expected precision and the resources must be obtained, see Fig. 9. The first possibility is to divide the x-axis of the resources in intervals of size δ, and compute the barycenters of all the points $(R_j, \mathcal{L}_j)$ falling in this interval, these would be the red crosses of Fig. 9. The second possibility is to fit this cloud of points with a NN. We set the training loss to be the MSE, i.e.

$$\mathcal{L}_{\mathrm{fit}} := \frac{1}{S}\sum_{j=1}^{S}\left[\mathcal{L}_j - f_{\mathrm{NN}}(R_j)\right]^2 \ , \tag{123}$$

which, for a single value of the resource, i.e. $R_j = R\ \forall j$, would converge to $f_{\mathrm{NN}}(R) = \frac{1}{S}\sum_{j=1}^{S}\mathcal{L}_j$, that is a NN will approximate the mean loss. This won't be exactly true for a cloud of points, but with Eq. (123) we incentivise the NN to converge toward the average loss for every value of the resources. All the plots in this paper have been produced with the first method, choosing an appropriate δ, except for those plots on the NV center platform with $T_2^\star = \infty$ and referring to the time-limited estimation. In the PGH line of the first plot in Fig. 2 there is a non-monotonicity for small T, that is a defect in the plot and an artifact on this way of interpolating with a NN.

E.3 Scaling of the time and memory requirements

Since the B estimations in a batch can be performed in parallel we will benefit from the use of a GPU or a TPU (Tensor Processing Unit) in the training of the agent. The main difference between the CPU and the GPU is that a CPU has fewer ($\sim \mathcal{O}(10)$) faster cores, while a GPU has many ($\sim \mathcal{O}(10^3)$) slower cores. With a large batchsize the use of hardware acceleration through a GPU will turn out to be essential and we will first examine the resource requirements of model-aware RL assuming that everything that can be parallelized has been is indeed executed in parallel. In this case the time requirement of the simulation is mainly influenced linearly by the number of measurements M in the training loop, that have to be executed necessarily sequentially. The update of the PF and the computation of the distributions moments all require $\mathcal{O}(N)$ multiplications each but can be done in parallel, where N is the number of particles. The memory requirement depends on the batchsize B and the number of particles N in the PF. Nevertheless because of the construction of the gradient, for which we need to keep in memory the results of all the intermediate computations, the number of measurements M has also an almost linear influence on the required memory. Finally, the total time used in the training is also proportional to the number of update steps I. Each update step comprises the complete run of an estimation batch together with the evaluation of the gradient and the update of the controls. The size of the NN has little impact on the training time and memory. We can summarise the above considerations as

$$\text{Memory} \sim \mathcal{O}(BMN)\,, \qquad \text{Time}_{\text{Par}} \sim \mathcal{O}(IM)\,. \tag{124}$$

Assuming that nothing can be parallelized (we have a single core) and therefore everything is sequential, if, as usual, the computational time in the CPU is dominated by the number of floating point multiplications, we instead have the time scaling

$$\text{Time}_{\text{Seq}} \sim \mathcal{O}(IBMN)\,, \tag{125}$$

while the memory requirement is unchanged. Neither a GPU nor a CPU will perfectly reproduce these theoretical scalings, because the GPU has a limited number of cores, but there is a tendency for a GPU to follow the scaling of Time_{Par} and for a CPU Time_{Seq}. If the batchsize B is very large (or the GPU not very powerful) the simulations in the batch can't all be executed in parallel and B starts to affect also the time requirements. If B and N are small a CPU may complete the training before a GPU, because of the smaller proportionality factor for the time requirement in Eq. (125) with respect to Eq. (124), due to the faster cores of the CPU. This analysis applies also to the training of a non-adaptive strategy, which is not resource-saving compared to the training of the NN. In the applications we expect our agent to run on a small controller near the sensor, where most definitely we won't have access to a GPU and lots of memory, which anyway are required only in the training phase. In this situation we have no batch and only one iteration, i.e. $I = B = 1$. Furthermore there is no extra M in the memory requirement appearing because of the automatic differentiation, so that the resource scaling in the application will be

$$\text{Memory} \sim \mathcal{O}(dN + N)\,, \qquad \text{Time}_{\text{Seq}} \sim \mathcal{O}(MN)\,, \tag{126}$$

where d is the number of parameters. For an estimation limited by the resources instead of the number of measurements, M must be intended as the average number of measurements employed for a fixed amount of resources. In general thanks to the resampling routine we can keep the number of particles fixed while increasing the precision arbitrarily. It is a good practise although to choose N proportional to the number of parameters to estimate, i.e. $N \sim \mathcal{O}(d)$. In the applications the memory requirement of the NN, and the multiplications needed to evaluate it at each step contribute respectively with additional $\mathcal{O}(n_l n_u)$ memory and $\mathcal{O}(n_l n_u^2)$ time (per step), where n_l is the number of layers and n_u the number of units per layer, so that we have

$$\text{Memory} \sim \mathcal{O}(dN + N + n_l n_u)\,, \qquad \text{Time}_{\text{Seq}} \sim \mathcal{O}(MN + M n_l n_u^2)\,. \tag{127}$$

If the control is non-adaptive we don't need this extra computations, and if the PF is removed from the picture (because we don't need real time feedback) we have that the memory and time requirements trivialize, i.e they scale respectively as $\mathcal{O}(1)$ and $\mathcal{O}(M)$. Of course the total time of estimation would be influenced also by the time it takes to perform the physical measurement on the probe, but here we are referring only to the computational time.

F Optimal strategies for frequentist optimization

The qsensoropt library can also optimize the Cramér-Rao bound (based on the Fisher information matrix) for the local estimation of the parameters $\boldsymbol{\theta}$. This is frequentist inference instead of Bayesian inference, this

last being the main topic of this work. The multiparameter Cramér-Rao bound (CR bound) is a lower bound on the Mean Square Error matrix of the frequentist estimator $\widehat{\boldsymbol{\theta}}$ at the position $\boldsymbol{\theta}$, expressed in terms of the FI matrix, i.e.

$$K := \mathbb{E}\left[(\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta})(\widehat{\boldsymbol{\theta}} - \boldsymbol{\theta})^{\intercal}\right] \geq F^{-1}(\boldsymbol{\theta}) \,, \tag{128}$$

with

$$F_{ij}(\boldsymbol{\theta}) := \mathbb{E}_y\left[\frac{\partial \log p_{\boldsymbol{\theta}}(y)}{\partial \theta_i}\frac{\partial \log p_{\boldsymbol{\theta}}(y)}{\partial \theta_j}\right] \,. \tag{129}$$

This result sets the maximum achievable precision of an estimator around $\boldsymbol{\theta}$, in other words, it limits the ability to distinguish reliably two close values $\boldsymbol{\theta}$ and $\boldsymbol{\theta} + \delta\boldsymbol{\theta}$. The expectation value is taken on many realizations of the experiment, i.e. on the probability distribution of the trajectories for the outcomes and the controls. Let us introduce the tuple $\boldsymbol{x}$ and $\boldsymbol{y}$ containing respectively the entirety of the controls and the outcomes of the measurements done in the experiment, then the FI matrix has the following expression

$$F_{ij}(\boldsymbol{\theta}) := \mathbb{E}_{\boldsymbol{y}}\left[\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})}{\partial \theta_i}\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})}{\partial \theta_j}\right] \,, \tag{130}$$

being $p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})$ the probability of the observed trajectory of outcomes at the point $\boldsymbol{\theta}$. Notice that the expectation value is taken on the whole trajectory of the experiment. By contracting Eq. (128) with the weight matrix $G \geq 0$ we get the scalar version of the CR bound, i.e.

$$\mathrm{tr}(G \cdot K) \geq \mathrm{tr}(G \cdot F^{-1}) := \mathcal{L}(\boldsymbol{\lambda}) \,, \tag{131}$$

where we have defined the loss to be optimized in the training, as a function of the trainable variables of the agent $\boldsymbol{\lambda}$. The gradient of the loss is

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}} = \mathrm{tr}\left(F^{-1}GF^{-1} \cdot \frac{\partial F}{\partial \boldsymbol{\lambda}}\right) \,. \tag{132}$$

The expectation value in the definition of F is approximated in the simulation by averaging the product of the log-likelihoods derivatives on a batch of estimations, i.e.

$$F \simeq \widehat{F} = \frac{1}{B}\sum_{k=1}^{B}\frac{\partial \log p(\boldsymbol{y}_k|\boldsymbol{x}_k,\boldsymbol{\theta})}{\partial \theta_i}\frac{\partial \log p(\boldsymbol{y}_k|\boldsymbol{x}_k,\boldsymbol{\theta})}{\partial \theta_j} = \frac{1}{B}\sum_{k=1}^{B} f_k \,. \tag{133}$$

where $(\boldsymbol{x}_k, \boldsymbol{y}_k)$ is the trajectory of a particular realization of the experiment in the batch of simulations and f_k is called the observed Fisher information. The unbiased gradient of the loss, that takes into account also the gradient of the probability distribution in the expectation value, can be computed as following

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\lambda}} \simeq \frac{1}{B}\frac{\partial}{\partial \boldsymbol{\lambda}}\mathrm{tr}\Big\{\mathrm{sg}\left(\widehat{F}^{-1}G\widehat{F}^{-1}\right)\sum_{k=1}^{B}\left[f_k + \mathrm{sg}(f_k)\log p(\boldsymbol{x}_k,\boldsymbol{y}_k|\boldsymbol{\theta})\right]\Big\} \,. \tag{134}$$

The $\mathrm{sg}(\cdot)$ is the stop gradient operator, and the probability $p(\boldsymbol{x}_k,\boldsymbol{y}_k|\boldsymbol{\theta})$ is the likelihood of the particular trajectory, that contains both the probability of the stochastic outcome and that of the control (in case it is stochastically generated). This is the gradient used in the update step of the stochastic gradient descent procedure for the optimization of frequentist estimation. We can also introduce the logarithmic loss, i.e.

$$\mathcal{L}_{\log}(\boldsymbol{\lambda}) := \log \mathrm{tr}(G \cdot F^{-1}) \,, \tag{135}$$

which is particularly useful to stabilize the training when the FI spans multiple orders of magnitude. If we have a broad prior on $\boldsymbol{\theta}$, but we are stile interested in local estimation, we can introduce the average FI, i.e.

$$\mathcal{F} := \int F(\boldsymbol{\theta})\pi(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} \,, \tag{136}$$

and optimize the loss

$$\mathcal{L}(\boldsymbol{\lambda}) := \mathrm{tr}\left[G \cdot \mathcal{F}^{-1}\right] \leq \int \mathrm{tr}\left[G \cdot F^{-1}(\boldsymbol{\theta})\right]\mathrm{d}\boldsymbol{\theta} \,, \tag{137}$$

which is a lower bound on the expectation value of the CR bound, because of the Jensen inequality applied to the matrix inverse. In the case of a single parameter the training would maximize the expected value of the Fisher information on the prior $\pi(\theta)$. It is possible to use a custom distribution $\widetilde{p}(y|x,\boldsymbol{\theta})$ for extracting

the measurements outcomes instead of $p(y|x,\boldsymbol{\theta})$ by using importance sampling. In this case the FI matrix is computed as

$$F_{ij}(\boldsymbol{\theta}) = \mathbb{E}_{\widetilde{p}}\left[\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})}{\partial\theta_i}\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})}{\partial\theta_j}\cdot\frac{p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})}{\widetilde{p}(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta},)}\right] , \tag{138}$$

which can be approximated on a batch as

$$F \simeq \frac{1}{B}\sum_{k=1}^{B} f_k \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k,\boldsymbol{\theta})}{\widetilde{p}(\boldsymbol{y}_k|\boldsymbol{x}_k,\boldsymbol{\theta})} , \tag{139}$$

with f_k defined in Eq. (133). Also the gradient of F is changed accordingly. Typically the distribution $\widetilde{p}$ is some perturbation of p, for example it can be obtained by mixing p with a uniform distribution on the outcomes. The importance sampling is useful when some trajectories have vanishingly small probability of occurring according to the model p but contribute significantly to the Fisher information. If these trajectories have some probability of occurring sampling with $\widetilde{p}$, then the estimator of the FI might be more accurate. The drawback is that the perturbed probability of the complete trajectory $\widetilde{p}(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})$ might be too different from the real probability (because of the accumulation of the perturbation at each step), so that the simulation might entirely miss the region in the space of trajectories in which the system state moves, thus delivering a bad estimate of the FI and a bad control strategy, upon completion of the training. Whether or not the importance sampling can be beneficial to the optimization should be checked case by case. The derivative with respect to $\boldsymbol{\theta}$ in the definition of f_k in Eq. (133) are computed through automatic differentiation. This means there there are two nested automatic differentiation environments, one for the parameter and one for the training variables of the agent.

G Precision lower bounds of the examples

In this section we apply the Bayesian Cramér-Rao bound to the estimation of various parameters on the NV center platform. This bounds will be based on the Fisher information [22], which we briefly define in the following. Refer to the literature for more details. Consider a stochastic variable y, which is extracted from a probability distribution $p_\theta(y)$, where θ is a parameter we want to estimate. This is a model for an experiment leading to a stochastic outcome. The information on θ available from y can be measured by the Fisher information (FI), defined as

$$I(\theta) := \mathbb{E}_y\left[\left(\frac{\partial \log p_\theta(y)}{\partial\theta}\right)^2\right] , \tag{140}$$

where the expectation value is taken over the distribution $p_\theta(y)$. If the experiment allows to be controlled through the parameter x, then the outcome probability is $p_\theta(y|x)$ and the FI inherits such dependence, i.e. we write $I(\theta|x)$. If the control parameter x is computed from a strategy h, which could be the Particle Guess Heuristic the a neural network, then we indicate it explicitly in the control x_h.

G.1 Bayesian Cramér-Rao bound

Given θ a single parameter to estimate, we call $I(\theta|\boldsymbol{x}_h)$ the Fisher information of a sequence of measurements with controls $\boldsymbol{x}_h = (x_0^h, x_1^h, \cdots, x_{M-1}^h)$, which are computed from a strategy h. The quantity $I(\theta|\boldsymbol{x}_h)$, together with the Fisher information of the prior $\pi(\theta)$, i.e. $I(\pi)$, defines a lower bound on the precision $\Delta^2\widehat{\theta}$ of whatever estimator $\widehat{\theta}$, that contains the expectation value of $I(\theta|\boldsymbol{x}_h)$ on $\pi(\theta)$, and is optimized on the strategy h. This lower bounds reads

$$\Delta^2\widehat{\theta} \geq \frac{1}{\sup_h \mathbb{E}_\theta\left[I(\theta|\boldsymbol{x}_h)\right] + I(\pi)} . \tag{141}$$

This definition appears in the work of Fiderer *et al.* [34]. For the NV centers the controls are the evolution time τ and the phase φ, this last however doesn't play any role in the computation of the lower bound, and it will be omitted in the following. The Fisher information of a sequence of measurements is always additive, even if the quantum probe is only measured weakly, but in dealing with projective measurements, as it is the case for NV center, the advantage is that the measurements are uncorrelated, and the same expression for the Fisher information applies to all of them, independently on the results of the previous measurements, i.e.

$$I(\theta|\boldsymbol{\tau}) = \sum_{t=1}^{M} I(\theta|\tau_t) \leq M \sup_\tau I(\theta|\tau) , \tag{142}$$

where M is the total number of measurements. The optimization of the single measurement FI gives directly the precision bound for the measurement-limited estimation:

$$\Delta^2\widehat{\theta} \geq \frac{1}{\sup_h \mathbb{E}_\theta\left[I(\theta|\boldsymbol{\tau}_h)\right] + I(\pi)} \geq \frac{1}{M\,\mathbb{E}_\theta\left[\sup_\tau I(\theta|\tau)\right] + I(\pi)}\,. \tag{143}$$

If the total evolution time is the limiting resource, then, the expression for the total FI is

$$I(\theta|\boldsymbol{\tau}) = T\sum_{t=1}^{M}\frac{\tau_t}{T}\left[\frac{I(\theta|\tau_t)}{\tau_t}\right] \leq T\sup_\tau \frac{I(\theta|\tau)}{\tau}\,, \tag{144}$$

with $\sum_{t=1}^{M}\tau_t = T$. In this expression the total FI is the weighted sum of the renormalized FI of each measurement, i.e. $\frac{I(\theta|\tau_t)}{\tau_t}$, and can be manifestly upper bounded by concentrating all the weights on the supremum of the renormalized FI. This gives the lower bound for the precision of the time-limited estimation:

$$\Delta^2\widehat{\theta} \geq \frac{1}{\sup_h \mathbb{E}_\theta\left[I(\theta|\boldsymbol{\tau}_h)\right] + I(\pi)} \geq \frac{1}{T\,\mathbb{E}_\theta\left[\sup_\tau \frac{I(\theta|\tau)}{\tau}\right] + I(\pi)}\,. \tag{145}$$

In the following we will apply this general observations to the derivation of the numerical bounds for DC magnetometry.

G.2 Evaluation of the Fisher information

Since the measurement outcome in the NV center is binary, we can compute the Fisher information for a parameter θ, given the control τ, as

$$I(\theta|\tau) = \mathbb{E}\left[\left(\frac{\partial \log p(\pm 1|\theta,\tau)}{\partial\omega}\right)^2\right] = \frac{\left(\frac{\partial p}{\partial\theta}\right)^2}{p(1-p)} = \frac{\left(\frac{\partial p}{\partial\theta}\right)^2}{\frac{1}{4}-(p-\frac{1}{2})^2}\,, \tag{146}$$

where we have used the definition in Eq. (140), and where $p := p(+1|\theta,\tau)$. For example, for a decoherence free estimation of the precession frequency ω we have $p := \cos^2\left(\frac{\omega\tau}{2}\right)$, from which $\frac{\partial p}{\partial\theta} = \tau\sin(\frac{\omega\tau}{2})\cos(\frac{\omega\tau}{2})$, and finally $I(\omega|\tau) = \tau^2$.

G.3 DC magnetometry

The lower bounds on the estimation of the frequency ω are reported in Table 1, and are represented in Fig. 2 of Section 3 as the shaded grey area. The left column of this table contains the bounds for a finite number of measurements M, while right column refers to the estimation with a fixed total evolution time T. The first row refers to the estimation of ω with perfect coherence while the second row refers to the estimation of ω with a finite and know $T_2^\star$. The symbol $I(\omega)$ indicate the FI of the prior for the precession frequency ω. The numerical

	Measurement		Time	
$T_2 = \infty$	$\frac{2^{-2(M+1)}}{3}$	147	$\frac{1}{T^2+I(\omega)}$	148
$T_2 < \infty$	$\max\left\{\frac{1}{\mu M(T_2^\star)^2+I(\omega)}, \frac{2^{-2(M+1)}}{3}\right\}$	149	$\frac{1}{0.5\,TT_2^\star+I(\omega)}$	150

Table 1: Lower bounds for the precision of the frequency estimation in DC magnetometry on an NV center.

values of the quantities appearing in Table 1, for $\omega \in (0,1)\,\mathrm{MHz}$ are: $\mu = 0.1619$, $I(\omega) = 12\,\mu\mathrm{s}^2$. In the following we derive these four bounds.

- The Fisher information for the decoherence-free precession frequency ω is given by $I(\omega|\tau) = \tau^2$, so that $\sup_\tau I(\omega|\tau) = \infty$ and the analysis based on the Cramèr-Rao bound doesn't gives a useful bound. Eq. (147) can be found by observing that each measurement gives at most one bit of information about the value of ω, because the measurement has a binary outcomes [34]. This bound is applied also for $T_2^\star < \infty$ in addiction to the one coming from the Fisher information.

- With a finite decoherence time $T_2^\star < \infty$ the FI for the frequency ω is

$$I(\omega|\tau, T_2^\star) = \frac{\tau^2 e^{-\frac{2\tau}{T_2^\star}} \cos^2\left(\frac{\omega\tau}{2}\right) \sin^2\left(\frac{\omega\tau}{2}\right)}{\left[e^{-\frac{\tau}{T_2^\star}} \cos^2\left(\frac{\omega\tau}{2}\right) + \frac{1-e^{-\frac{\tau}{T_2^\star}}}{2}\right]\left[e^{-\frac{\tau}{T_2^\star}} \sin^2\left(\frac{\omega\tau}{2}\right) + \frac{1-e^{-\frac{\tau}{T_2^\star}}}{2}\right]} , \tag{151}$$

  which, by defining $C := \cos^2\left(\frac{\omega\tau}{2}\right)$, can be bounded in the following way

$$I(\omega|\tau, T_2^\star) = \frac{\tau^2 e^{-\frac{2\tau}{T_2^\star}} C(1-C)}{\left[\frac{1}{4} - e^{-\frac{2\tau}{T_2^\star}}(C-\frac{1}{2})^2\right]} \leq \frac{\tau^2 e^{-\frac{2\tau}{T_2^\star}}}{1-e^{-\frac{2\tau}{T_2^\star}}} = (T_2^\star)^2 \frac{x^2 e^{-2x}}{1-e^{-2x}} , \tag{152}$$

  where $x = \frac{\tau}{T_2^\star}$. The maximization in $x \in \mathbb{R}_+$ gives $\sup_\tau I(\omega|\tau, T_2^\star) = \mu(T_2^\star)^2$ with $\mu = 0.1619$. Inserting this expression in Eq. (143) gives the first term in the maximum of Eq. (149), the second term was explained in the previous point.

- Regarding the time-constrained lower bounds, for $T_2^\star = \infty$, the total FI is maximized by performing a single measurement of time duration $\tau = T$, which gives Eq. (148), through the application of Eq. (145).

- For $T_2^\star < \infty$ we have to maximize the normalized FI in $x \in \mathbb{R}_+$, i.e.

$$\frac{I(\omega|\tau, T_2^\star)}{\tau} \leq \frac{\tau e^{-\frac{2\tau}{T_2^\star}}}{1-e^{-\frac{2\tau}{T_2^\star}}} \leq T_2^\star \frac{x e^{-2x}}{1-e^{-2x}} \leq \frac{T T_2^\star}{2} , \tag{153}$$

  from which Eq. (150) follows from Eq. (145).

H Backward recursion method for the optimization of the strategy

In this section we set the stage to understand what function the agent must approximate by formulating the problem in terms of a backward recursion. In this section we will see how the optimal control could theoretically be derived in other ways, so that the training will appear less as an unintelligible black-box and more as the solution to a well-posed problem (though we won't probably have uniqueness). The output of the agent at the $t+1$-th steps is x_{t+1}, that is, the control of the current evolutions and measurements. In the following we will define formally the function that the agent must learn to approximate, in doing this we will assume that the control x of the quantum sensor is a continuos real variable. Consider an estimation with M measurements, i.e. $t = 0, 1, \ldots, M-1$. Let us focus on the last one only and recall the definition of the particle filter ensemble before after the last measure $M-1$, i.e. $\mathfrak{p}_M := \{\boldsymbol{\theta}_j^{M-1}, w_j^{M-1}\}_{j=1}^N$. Then we can write the ensemble of the PF at the final step $t = M-1$ as

$$\mathfrak{p}_M = \mathfrak{B}(\mathfrak{p}_{M-1}, x_{M-1}, y_{M-1}) , \tag{154}$$

where $\mathfrak{B}$ encodes the application of the Bayes rule in Eq. (20). The ensemble $\mathfrak{p}_M$ inherits the stochasticity from the random measurement outcome y_{M-1}. Per definition $\mathfrak{p}_0$ is the initial PF ensemble that represents the prior $\pi(\boldsymbol{\theta})$. In Appendix D.1 we mentioned that the final loss is a scalar function $\ell(\mathfrak{p}_M, \boldsymbol{\theta})$ of the final PF ensemble and of the true value $\boldsymbol{\theta}$, like the squared derivation of the estimator from the true value. The final loss can be expressed as $\ell(\mathfrak{B}(\mathfrak{p}_{M-1}, x_{M-1}, y_{M-1}), \boldsymbol{\theta})$ and it's expectation value on y_{M-1} (the expected loss), computed with the density in Eq. (15), reads

$$\overline{\ell}(\mathfrak{p}_{M-1}, x_{M-1}, \boldsymbol{\theta}) := \int \ell(\mathfrak{B}(\mathfrak{p}_{M-1}, x_{M-1}, y_{M-1}), \boldsymbol{\theta}) p(y_{M-1}|x_{M-1}, \boldsymbol{\theta}) dy_{M-1} . \tag{155}$$

If the outcome probability, the prior and the loss are continuos functions we can also expect $\overline{\ell}(\mathfrak{p}_{M-1}, x_{M-1}, \boldsymbol{\theta})$ to be continuous in its parameters. Without aiming at full rigour, we say that the regularity properties of the probability densities are passed down to the expectation value. Now we look for the minimum of this function, which is realized by solving

$$\frac{\mathrm{d}\overline{\ell}(\mathfrak{p}_{M-1}, x_{M-1}^\star, \boldsymbol{\theta})}{\mathrm{d}x_{M-1}} = 0 . \tag{156}$$

This equation defines implicitly the optimal control $x_{M-1}^\star := r_{M-1}(\mathfrak{p}_{M-1}, \boldsymbol{\theta})$, where $x_{M-1}^\star$ realizes the absolute minimum of the expected loss. r_{M-1} inherits some regularity property (at least locally) from $\overline{\ell}(\mathfrak{p}_{M-1}, x_{M-1}, \boldsymbol{\theta})$

thanks to the implicit function theorem. The control $x^\star_{M-1}$ can still have discontinuities in $\mathfrak{p}_{M-1}$ if the expected loss has multiple competing minima. The dependence on $\boldsymbol{\theta}$ is rather inconvenient, because it is unknown, but we can think of substituting $\boldsymbol{\theta}$ with its estimator $\widehat{\boldsymbol{\theta}}_{M-2}$ to get $x^\star_{M-1} = r_{M-1}(\mathfrak{p}_{M-1}, \widehat{\boldsymbol{\theta}}_{M-2})$. We will however never do explicit optimizations with this approach, the introduction of machine learning in quantum metrology serves precisely to avoid these cumbersome computations. Until now we have only optimized the last control, but fortunately all these operations can be repeated with minor changes for the $t = M-2$ measurement step. Let us start from $\mathfrak{p}_{M-1}$ expressed as function of the ensemble $\mathfrak{p}_{M-2}$:

$$\mathfrak{p}_{M-1} = \mathfrak{B}(\mathfrak{p}_{M-2}, x_{M-2}, y_{M-2}) \,, \tag{157}$$

we insert this expression in Eq. (154) to get

$$\mathfrak{p}_M = \mathfrak{B}(\mathfrak{B}(\mathfrak{p}_{M-2}, x_{M-2}, y_{M-2}), x_{M-1}, y_{M-1}) \,. \tag{158}$$

By substituting this in the loss $\ell(\mathfrak{p}_M, \boldsymbol{\theta})$ we get $\ell(\mathfrak{B}(\mathfrak{B}(\mathfrak{p}_{M-2}, x_{M-2}, y_{M-2}), x_{M-1}, y_{M-1}), \boldsymbol{\theta})$, but the optimal x_{M-1} has been already computed as a function of the PF ensemble. Whatever control we suggest for the step $t = M-2$ it doesn't change the optimal control at the following step. In other words whatever control x_{M-2} gets applied, the optimal action for the last time step is always $x^\star_{M-1}$, we can therefore insert it in the loss at the step $M-2$, i.e.

$$\ell(\mathfrak{p}_{M-2}, x_{M-2}, y_{M-2}, y_{M-1}, \boldsymbol{\theta}) := \ell(\mathfrak{B}(\mathfrak{B}(\mathfrak{p}_{M-2}, x_{M-2}, y_{M-2}), r_{M-1}(\mathfrak{B}(\mathfrak{p}_{M-2}, x_{M-2}, y_{M-2}), \boldsymbol{\theta}), y_{M-1}), \boldsymbol{\theta}) \,, \tag{159}$$

where we have also used the expression for $\mathfrak{p}_{M-1}$ of Eq. (157) in the definition of $x^\star_{M-1}$, and we have redefined the parameters of ℓ. Again we take the expectation value on the measurement outcomes, i.e.

$$\overline{\ell}(\mathfrak{p}_{M-2}, x_{M-2}, \boldsymbol{\theta}) := \int \ell(\mathfrak{p}_{M-2}, x_{M-2}, y_{M-2}, y_{M-1}, \boldsymbol{\theta}) p(y_{M-1}|\boldsymbol{\theta}, x^\star_{M-1}) p(y_{M-2}|\boldsymbol{\theta}, x_{M-2}) dy_{M-1} dy_{M-2} \,. \tag{160}$$

By taking the derivative of this expected loss with respect to x_{M-2}, we define implicitly the optimal control $x^\star_{M-2} = r_{M-2}(\mathfrak{p}_{M-2}, \boldsymbol{\theta})$ as the solution of

$$\frac{\mathrm{d}\overline{\ell}(\mathfrak{p}_{M-2}, x^\star_{M-2}, \boldsymbol{\theta})}{\mathrm{d}x_{M-2}} = 0 \,, \tag{161}$$

where $x^\star_{M-2}$ realizes the absolute minimum of $\overline{\ell}$. Notice that this is not a greedy optimization: the value of x_{M-2} is not chosen to optimize the loss one step head in the future but the final loss, knowing what the strategy in the next step will be. We could treat all the previous measurements in the same manner, if it wasn't for the expectation values of the loss, that become more and more complicated. In this way we can inductively proceed in reverse to the start of the estimation $t = 0$ and find in the process the family of functions $r_t(\mathfrak{p}_t, \boldsymbol{\theta}_{t-1})$ that express the optimal controls $x^\star_t$. As discussed previously we can redefine $r_t(\mathfrak{p}_t) = r_t(\mathfrak{p}_t, \widehat{\boldsymbol{\theta}}_{t-1})$, in order to get rid of the dependence on the unknown value of the parameters $\boldsymbol{\theta}$. We then introduce $r(\mathfrak{p}_t, t) = r_t(\mathfrak{p}_t)$, which is the function that the agent is trying to approximate in the training, i.e. the map that spits out the optimal control at a given measurement step t, provided the ensemble PF at the previous step. Heuristically we expect the optimal control to be inhomogeneous in time because like in many application of RL a good strategy encompasses a phase of "exploration" followed by a phase of "exploitation" of what has been learned [18, 19].

I Backpropagation of the gradient

With the help of an automatic differentiation framework we compute the gradient of the modified loss in Eq. (92) in order to perform the training. Let us for the moment neglect the log-likelihood terms in this expression and concentrate only on the first part. If the loss is computed only from the ensemble at the last measurement step, then it takes the form

$$\ell_{\boldsymbol{\theta}} \circ \mathfrak{B}_{x_{M-1}, y_{M-1}} \circ \mathfrak{B}_{x_{M-1}, y_{M-1}} \circ \cdots \circ \mathfrak{B}_{x_0, y_0}(\mathfrak{p}_0) \,. \tag{162}$$

where $\ell_{\boldsymbol{\theta}}(\mathfrak{p}) := \ell(\mathfrak{p}, \boldsymbol{\theta})$ acts on the ensemble PF and is the individual loss of each simulation, e.g. the square error. The function $\mathfrak{B}_{x_t, y_t}(\mathfrak{p}) := \mathfrak{B}(\mathfrak{p}, x_t, y_t)$ applies the Bayesian update to the PF, which depends on the outcome of the measurement y_t and on the control x_t. The initial ensemble of the PF is named $\mathfrak{p}_0$. From a theoretical point of view differentiating this expression means applying repeatedly the rule for the derivation

of the composite function and propagating the derivative through the various stages of the estimation, this is called backpropagation. Let us compute explicitly the derivative of the loss for the last two steps of the estimation. We define $\mathfrak{p}_M = \mathfrak{B}_{x_{M-1},y_{M-1}}(\mathfrak{p}_{M-1})$ and $\mathfrak{p}_{M-1} = \mathfrak{B}_{x_{M-2},y_{M-2}}(\mathfrak{p}_{M-2})$, and apply the chain rule. We will indicate with the partial derivative symbol the derivatives with respect to the parameters of a function (which can also appear in the subscript), while the symbol $\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}$ is reserved to the total derivative with respect to $\boldsymbol{\lambda}$.

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}\ell_{\boldsymbol{\theta}}(\mathfrak{p}_M) = \frac{\partial \ell_{\boldsymbol{\theta}}(\mathfrak{p}_M)}{\partial \mathfrak{p}}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}\mathfrak{B}_{x_{M-1},y_{M-1}}(\mathfrak{p}_{M-1})\,. \tag{163}$$

The total derivate in the right-hand term an be expanded again with the chain rule:

$$\frac{\partial \mathfrak{B}_{x_{M-1},y_{M-1}}(\mathfrak{p}_{M-1})}{\partial x_{M-1}}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}x_{M-1}(\boldsymbol{\lambda},\mathfrak{p}_{M-1}) + \underline{\frac{\partial \mathfrak{B}_{x_{M-1},y_{M-1}}(\mathfrak{p}_{M-1})}{\partial \mathfrak{p}}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}\mathfrak{B}_{x_{M-2},y_{M-2}}(\mathfrak{p}_{M-2})} \quad \text{a)}\,. \tag{164}$$

Only the parameters x_{M-1} and $\mathfrak{p}_{M-1}$ can carry a dependence on $\boldsymbol{\lambda}$. Regarding the measurement outcomes y_{M-1} we already discussed their independence on $\boldsymbol{\lambda}$, expressed by Eq. (84). The control x_{M-1} has an explicit dependence on $\boldsymbol{\lambda}$ because it has been produced by the agents, but also has a dependence on $\boldsymbol{\lambda}$ through the PF ensemble, so that we can expand the total derivative in the following way

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}x_{M-1}(\boldsymbol{\lambda},\mathfrak{p}_{M-1}) = \underline{\frac{\partial x_{M-1}(\boldsymbol{\lambda},\mathfrak{p}_{M-1})}{\partial \boldsymbol{\lambda}}} \quad \text{b)} \tag{165}$$

$$+ \underline{\frac{\partial x_{M-1}(\boldsymbol{\lambda},\mathfrak{p}_{M-1})}{\partial \mathfrak{p}}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}\mathfrak{B}_{x_{M-2},y_{M-2}}(\mathfrak{p}_{M-2})} \quad \text{c)}\,. \tag{166}$$

The first piece of the derivative is the dependence on $\boldsymbol{\lambda}$ that comes from the last application of the agent, while the second piece represent the backpropagation through the input of the agent and the Bayesian update of the PF ensemble. In general, the gradient is backpropagated through all the applications of the agent until it reaches the beginning. We notice that we can write the total derivative of $\mathfrak{p}_M$ as a function of the total derivative of $\mathfrak{p}_{M-1}$, i.e.

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}\mathfrak{B}_{x_{M-1},y_{M-1}}(\mathfrak{p}_{M-1}) = Q_{M-1} + H_{M-1}\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}\mathfrak{B}_{x_{M-2},y_{M-2}}(\mathfrak{p}_{M-2}) \tag{167}$$

where

$$Q_{M-1} := \frac{\partial \mathfrak{B}_{x_{M-1},y_{M-1}}(\mathfrak{p}_{M-1})}{\partial x_{M-1}}\frac{\partial x_{M-1}(\boldsymbol{\lambda},\mathfrak{p}_{M-1})}{\partial \boldsymbol{\lambda}}\,, \tag{168}$$

$$H_{M-1} := \frac{\partial \mathfrak{B}_{x_{M-1},y_{M-1}}(\mathfrak{p}_{M-1})}{\partial x_{M-1}}\frac{\partial x_{M-1}(\boldsymbol{\lambda},\mathfrak{p}_{M-1})}{\partial \mathfrak{p}} + \frac{\partial \mathfrak{B}_{x_{M-1},y_{M-1}}(\mathfrak{p}_{M-1})}{\partial \mathfrak{p}}\,. \tag{169}$$

We have arrived to a family of recurrence equations, which have the solution

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\lambda}}\mathfrak{B}_{x_{M-1},y_{M-1}}(\mathfrak{p}_{M-1}) = \sum_{t=0}^{M-1} Q_t \prod_{m=t+1}^{M-1} H_m\,, \tag{170}$$

where Q_t and H_m are defined analogously to Q_{M-1} and H_{M-1}. The H_m terms have each two summand, and when multiplied together the number of terms in the gradient grows exponentially in the number of measurement M, and generates gradient terms corresponding to multiple repeated backpropagations through the agents. This are a kind of "higher order" gradient terms. In our implementation of the training we simplify the gradient by introducing a stop gradient operation before the input of the agent, so that Eq. (2) is actually implemented as

$$x_{t+1} = \mathcal{F}\{\mathrm{sg}\,[P(\boldsymbol{\theta}|\boldsymbol{x}_t,\boldsymbol{y}_t); R_t; t]\}\,, \tag{171}$$

This modification doesn't change the forward pass, that is, the results of simulations are the same, but it affects the backpropagation of the gradient, in particular it makes the first term of H_m disappear, because

$$\frac{\partial x_m\left(\boldsymbol{\lambda}, \mathrm{sg}\,[\mathfrak{p}_m]\right)}{\partial \mathfrak{p}} = 0\,. \tag{172}$$

Such simplification reduces considerably the training time and doesn't really affect, at least from a theoretical point of view, the ability of the agent to learn the optimal strategy, on the contrary it might even improve it. Before we back up this last assertions we want to recapitulate our analysis of the backpropagation with

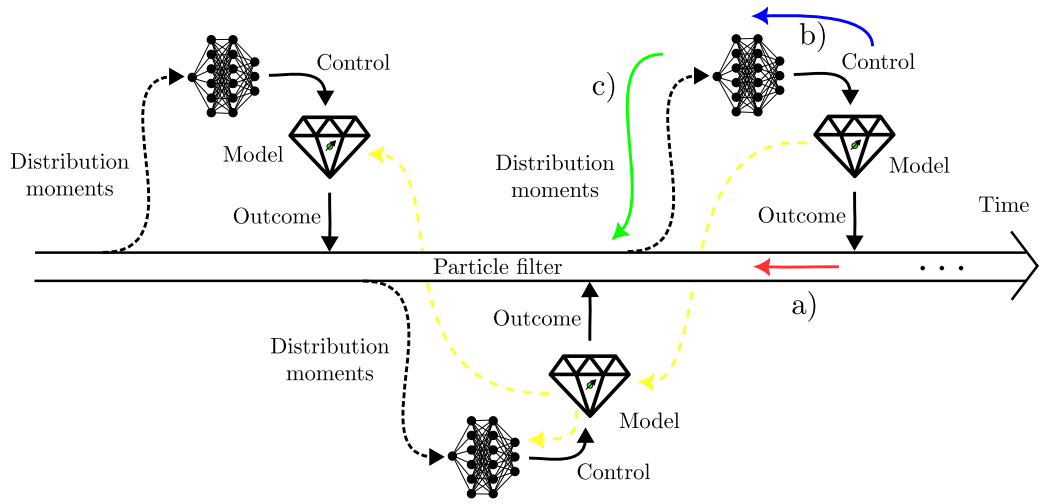


Figure 10: The fat empty arrow in the middle of the picture represents the PF, which is updated via the Bayes rule, indicated with the function $\mathfrak{B}$ in the text, after it receive an outcome from the measurement on the probe. The second term of Eq. (164), underlined in red and labelled with a), corresponds to the backpropagation of the gradient along the history of the PF ensemble, whose weights and particles inherits a dependence on $\boldsymbol{\lambda}$ from the actions of the agent. Visually this corresponds to backpropagation along the red arrow labelled with a). Through the match between the underlined terms in Eq. (166) and Eq. (164) and the arrows in the figure can visualize the origin of each term of the gradient. The blue term of Eq. (166) labelled with b) accounts for the dependence on $\boldsymbol{\lambda}$ that comes from the controls computed by the agent and propagated through the application of the Bayes rule and the computation of x_t. The green term of Eq. (166) labelled with c) is the gradient propagating from the input of the agent to the previous PF ensemble. This term is responsible for the "higher-order" terms of the gradient, that propagate multiple times through the agent. Inserting the stop gradient in Eq. (172) means cutting the green line. The dashed yellow line represents the propagation of the gradient through the state of the probe, when this is not reinitialized between the measurements.

the help of Fig. 10. A control strategy is called myopic if it optimizes the information gained from the next measurement only, while it is non-myopic if it optimizes many steps ahead in the future. It might seem that by cutting the gradient propagation through the green arrows we limit the optimization to the class of myopic strategies, but this is not true because the optimization is always done for the final precision. The input of the agent is basically a constant now and the t-th term in the summation of Eq. (170) pulls the weights of the NN to minimize the final loss given the PF ensemble at the $t-1$-th step. In order to have non-myopic adaptive strategy backpropagating the gradient through the green arrows is redundant. The gradient of the log-likelihood terms in the loss of Eq. (93) is also simplified when the stop gradient is acting in Eq. (172). In the model $p(y_t|x_t,\boldsymbol{\theta})$ the only term that depends on $\boldsymbol{\lambda}$ is the control, and the gradient propagates through a single application of the agent. Had we not inserted the stop gradient, the gradient of each summands in the log-likelihood would have been propagated through all the past applications of the agent, a scenario that happens anyway if the probe state is not reinitialized between measurements. Before ending the discussion on the gradient backpropagation we want to consider yet another possibility of truncating the gradient. We could indeed image to stop the flow of the derivative through the evolution of the PF ensemble, which means cutting the red line in Eq. (164). This eliminates the recurrence equation and trivializes the gradient, which now accounts only for the very last control. That is, with such modification, the agent will learn to optimize only the last measurement. If the loss is cumulative however, like in Eq. (106), all the controls will be optimized, but in a myopic way. In this scenario we introduce the modified logarithmic loss, i.e.

$$\widetilde{\mathcal{L}}_{\log}(\boldsymbol{\lambda}) := \frac{1}{TB}\sum_{t=1}^{M-1}\log\left[\frac{\sum_{k=1}^{B}\ell(\widehat{\boldsymbol{\theta}}_{k,t},\boldsymbol{\theta}_k)}{\mathrm{sg}\left[\sum_{k=1}^{B}\ell(\widehat{\boldsymbol{\theta}}_{t-1,k},\boldsymbol{\theta}_k)\right]}\right]. \tag{173}$$

Each term in this summation is the empirical information gain for a Gaussian posterior. In the forward pass the loss is the total empirical information gain, because the stop gradient operators don't play any role, and the series can be resummed. In the computation of the gradient, however, the information gain for each measurement is optimized greedily, as done in [78] and in the package optbayesexpt [39]. If the stop gradient in the denominator is applied only every n measurement, then we are optimizing the information gain planning n steps ahead in the future. It is advisable to put a regularization at the denominator of the loss in Eq. (173) to avoid dividing by zero.