Robust Online Hamiltonian Learning

Christopher E. Granade*[,1,2], Christopher Ferrie[1,3], Nathan Wiebe[1,3], and D. G. Cory[1,4,5]

1 **Institute for Quantum Computing, University of Waterloo, Waterloo, Ontario, Canada**
2 **Department of Physics, University of Waterloo, Waterloo, Ontario, Canada**
3 **Department of Applied Mathematics, University of Waterloo, Waterloo, Ontario, Canada**
4 **Department of Chemistry, University of Waterloo, Waterloo, Ontario, Canada**
5 **Perimeter Institute for Theoretical Physics, Waterloo, Ontario, Canada**

Abstract

In this work we combine two distinct machine learning methodologies, sequential Monte Carlo and Bayesian experimental design, and apply them to the problem of inferring the dynamical parameters of a quantum system. The algorithm can be implemented *online* (during experimental data collection), avoiding the need for storage and post-processing. Most importantly, our algorithm is capable of learning Hamiltonian parameters even when the parameters change from experiment-to-experiment, and also when additional noise processes are present and unknown. The algorithm also numerically estimates the Cramer-Rao lower bound, certifying its own performance. We further illustrate the practicality of our algorithm by applying it to two test problems: (1) learning an unknown frequency and the decoherence time for a single–qubit quantum system and (2) learning couplings in a many–qubit Ising model Hamiltonian with no external magnetic field.

1998 ACM Subject Classification G.3 Probability and Statistics

Keywords and phrases Quantum information, sequential Monte Carlo, Bayesian, experiment design, parameter estimation

Digital Object Identifier 10.4230/LIPIcs.TQC.2013.106

1 Introduction

The problem of characterizing quantum systems is of fundamental importance to quantum information science. Without an accurate understanding, for example, of the noise processes that a quantum computer experiences, error correction may be quite difficult; furthermore, certification of quantum dynamics is essential for determining whether the predictions made by a quantum simulator can be trusted. This latter problem is especially timely since quantum simulation experiments are approaching a complexity where classical computers are unable to simulate their evolution [1, 2, 3]. Natural solutions to this problem, such as tomographic methods [4, 5, 6, 7, 8, 9, 10], are often impractical for learning parameters for large quantum systems, as well as for learning parameters such as T_2. This prompts the question of whether there exists a practical error robust technique that can be used to characterize quantum systems with unknown decoherence processes.

* Corresponding author (cgranade@cgranade.com).

8th Conference on Theory of Quantum Computation, Communication and Cryptography.
Editors: Simone Severini and Fernando Brandao; pp. 106–125


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

We make this learning process tractable by utilizing information about a system, rather than starting from worst-case assumptions such as those made in traditional quantum process and state tomography. In practice, we often have knowledge about the dynamical model that describes a system of interest, and wish to improve that knowledge by estimating specific model parameters. Thus, practical Hamiltonian finding can often be achieved via a suitable parameterization of the Hamiltonian, $H(x_1, \ldots, x_d)$, reducing the problem to estimating the vector of parameters $\boldsymbol{x} = (x_1, \ldots, x_d)$. The task we consider is the design of experiments for the purpose of deducing these parameters in the smallest number of experiments possible. Our algorithm also provides a *region estimation* for the Hamiltonian parameters that encloses some fixed volume of parameter space in which the mean or the variance of the Hamiltonian parameters are expected to be found with high–probability. We also generalize this concept to allow the algorithm to learn *hyperparameters*, which describe the distribution of the Hamiltonian parameters in cases where the parameters randomly drift between experiments. Pseudocode for all of our algorithms is given in Appendix B.

Our algorithm achieves this by combining sequential Monte Carlo methods [11] with Bayesian experiment design [12] to choose experiments that maximize the expected reduction in the uncertainty in the unknown parameters based on the results of prior experiments. We call such derived strategies *adaptive* or *online*. This approach not only reduces the number of experiments needed to learn the unknown parameters within a fixed error tolerance, but it also makes the learning process more robust. In addition to robustness, Bayesian updating provides a natural estimate of the uncertainty in the unknown parameters in the form of the width of the prior distribution. In contrast, it can be difficult to quantify the uncertainty in the estimated Hamiltonian using traditional methods based on inversion or tomography.

It is worth noting that approaches that are similar to our own have been considered very recently in a wide variety of classical contexts [13, 14, 15, 16, 17], and also for measurement adaptive quantum state tomography [18]. Other machine learning ideas have also been generalized to the quantum domain [19, 20, 21, 22, 23, 24, 25]; however, to the best of our knowledge, no method based on ideas from machine learning has been proposed for learning unknown Hamiltonian parameters that is as broadly applicable or as robust to noise as our method.

This paper is organized as follows. In section 2, we review the formalism of Bayesian experimental design. Section 3 introduces the sequential Monte Carlo algorithm. In section 4, we discuss the application of our algorithm to region estimation and hyperparameter estimation. We then explore the implications of the numerical benchmarking results in sections 5 and 6 before concluding.

2 Experimental Design Formalism

The essence of our experimental design process is that we choose experiments not according to a pre–determined sequence, but rather our algorithm adaptively chooses experiments that are expected to be very informative (given the current state of knowledge of the unknown parameters). We model a sequence of experiments as a sequence of experimental controls $\{c_1, \ldots, c_N\}$ and a corresponding sequence of acquired data $\{d_1, \ldots, d_N\}$. Bayesian updating is then used to formalize how the acquired data impacts our current state of knowledge about the unknown Hamiltonian, where we connect to the theory of parameter estimation by using the predictions of quantum mechanics as a probabilistic model, called a likelihood function.

To clarify, suppose we have performed an experiment with control settings c_1. We are then ultimately interested in the *posterior distribution* $\Pr(\boldsymbol{x}|d_1; c_1)$, the probability distri-

bution of the model parameters $\boldsymbol{x}$ given this data. By Bayes' rule and the conditional independence of each datum, the posterior distribution is given by

$$\Pr(\boldsymbol{x}|d_1;c_1) = \frac{\Pr(d_1|\boldsymbol{x};c_1)}{\Pr(d_1|c_1)}\Pr(\boldsymbol{x}),$$

where $\Pr(\boldsymbol{x})$ is the *prior*, which encodes any *a priori* knowledge of the model parameters. $\Pr(d_1|\boldsymbol{x};c_1)$ is the likelihood, which can be computed using Born's rule. The total likelihood $\Pr(d_1|c_1)$ can simply be thought as a normalization factor. Subsequent experiments update the prior according to the following iterative rule

$$\Pr(\boldsymbol{x}|d_{j+1},\dots;c_{j+1},\dots) = \frac{\Pr(d_{j+1}|\boldsymbol{x};c_{j+1})}{\Pr(d_{j+1}|c_{j+1})}\Pr(\boldsymbol{x}|d_j,\dots;c_j,\dots).$$

The idea of adaptive experiment design can be formalized in various ways, the most natural for our purposes being called *Bayesian experimental design* [12]. For this, we conceive of possible future data d_{N+1} obtained from a, possibly different, set of experimental controls c_{N+1}. The probability of obtaining this data can be computed from the distributions at hand via marginalizing over model parameters

$$\Pr(d_{j+1}|d_j,\dots;c_{j+1}) = \int \Pr(d_{j+1}|\boldsymbol{x};c_{j+1})\Pr(\boldsymbol{x}|d_j,\dots;c_j,\dots)\,\mathrm{d}\boldsymbol{x}.$$

Note, in the remainder of this work, we will use the following abbreviated notation for expectation values:

$$\Pr(d_{j+1}|d_j,\dots;c_{j+1},\dots) = \mathbb{E}_{\boldsymbol{x}|d_{j+1},\dots;c_{j+1},\dots}[\Pr(d_{j+1}|\boldsymbol{x};c_{j+1})], \tag{1}$$

where the subscript on $\mathbb{E}$ denotes the variable for the expectation to be taken over.

The expectation value in (1) can be used to inform the algorithm about the choices of experimental parameters that are more useful than others. This usefulness is quantified, for a given choice of a *utility function* $U(d_{j+1},c_{j+1})$, by the expected *utility* of an experiment

$$U(c_{j+1}) = \mathbb{E}_{d_{j+1}|d_j,\dots;c_{j+1},\dots}[U(d_{j+1},|c_{j+1})],$$

where $U(d_{j+1},c_{j+1})$ is the utility we would derive if experiment c_{j+1} yielded result d_{j+1}. The choice of the utility function is motivated by the figure of merit that we want to optimize, and will be considered in Appendix A.

3 Sequential Monte Carlo Algorithm

A major drawback of using Bayesian inference for Hamiltonian learning stems from the fact that the parameter space is continuous. This means that the prior will have in general support over an infinite number of possible Hamiltonians, which in turn makes applying Bayes' rule and sampling from the resultant posterior intractable. We address this problem by using *sequential Monte Carlo* (SMC) methods, such as those described in the recent tutorial by Doucet and Johansen [11].

At each step of the SMC algorithm, the current distribution is approximated by a weighted sum of Dirac-delta functions, so that $\Pr(\boldsymbol{x}|D) \approx \sum_{k=1}^{n} w_k(D)\delta(\boldsymbol{x}-\boldsymbol{x}_k)$, where $w_k(D)$ is a *weight* that describes the relative plausibility of the hypothesis $\boldsymbol{x}_k$, having observed the data record D. Each term in this sum is referred to as a *particle*.

Since the Bayes update rule for many observations $\{d_1,d_2,\dots,d_N\}$ can be processed sequentially by updating the weights: $w_k(d_{j+1}\cup D) = \Pr(d_{j+1}|\boldsymbol{x}_k)w_k(d_j)/\mathcal{N}$, where $\mathcal{N}$ is found by the normalization constraint that $\sum_k w_k(D) = 1$.

The particle approximation can be made arbitrarily accurate by increasing the number of particles, and will be a good approximation at every update provided we feed in, at the initial stage, the appropriate weights $\{w_k\}$ and support points $\{\boldsymbol{x}_k\}$. Since both the weights and support points of the particles carry information about distributions over the model parameters $\boldsymbol{x}$, we can without loss of generality choose the initial weights to be uniform, $w_k = 1/n$ for all k, and the initial support points to be samples from the correct prior $\Pr(\boldsymbol{x})$. Having made the particle approximation, we perform Bayes updates using the algorithm below.

Sequential Monte Carlo techniques require careful effort to avoid introducing errors due to limited numerical precision. The first problem any SMC algorithm runs into is zero weights. This is doubly painful since we are effectively operating with fewer particles but using the same amount of computational resources. Since the support of our approximate distribution is a measure-zero set according to the correct distribution, all the weights will eventually be zero; we cannot avoid this but it can be postponed by using *resampling* techniques.

Generally, the idea behind resampling is to adaptively change the location of the particles to those which are most likely. This works because a particle approximation to a probability distribution can be equally well approximated using constant weight particles with variable density, or variable weight particles with constant density. Hence, we can "resample" the distribution by using constant weight particles to approximate to the prior distribution to alleviate problems caused by the weights of the particles becoming small enough to impact the numerical stability of the methods. The simplest of these types of algorithm chooses n particles (the original number), with replacement, according to the distribution of weights then reset the weights of all particles to $1/n$. Thus, zero weight particles are "moved" to higher weight locations. To determine when to resample, we shall compare the effective sample size $n_{\text{ess}} = 1/\sum_i w_i^2$ to a threshold `resample_threshold`, which is the effective ratio of the original number of particles n. We use `resample_threshold` $= 0.5$, as suggested by [26].

The resampling algorithm we use was first proposed in [26] and is given explicitly in Algorithm 2. The idea behind the algorithm conforms to the intuition given above but it incorporates randomness to search larger volumes of the parameter space. This randomness is inserted in the resampling algorithm by applying a random perturbation to the location of each particle that is introduced during the resampling process. Thus, the new particles are randomly spread around the previous locations of the old. More formally, we model this by randomly choosing a particle location $\boldsymbol{x}_i$, then perturbing it by a normally distributed vector $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \Sigma)$ (we will come back to how to choose the mean and covariance). The new particles are thus samples of the convolved distribution

$$p(\boldsymbol{x}') = \sum_i w_i \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}' - \boldsymbol{\mu}_i)^{\mathrm{T}} \Sigma^{-1} (\boldsymbol{x}' - \boldsymbol{\mu}_i)\right), \tag{2}$$

where k is the number of model parameters. A distribution of this form is known as a *mixture distribution*, and can be efficiently sampled by first choosing a particle, then choosing a perturbation vector.

To choose the mean $\boldsymbol{\mu}_i$ of each term in the resampling mixture distribution, we choose a vector that is a convex combination of the original particle location $\boldsymbol{x}_i$ and the expected model $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{x}]$, so that $\boldsymbol{\mu}_i = a\boldsymbol{x}_i + (1-a)\boldsymbol{\mu}$, where a is a tunable parameter of the resampling algorithm. We will use $a = 0.98$, as suggested by [26]. The covariance of each perturbation is then given by $\boldsymbol{\Sigma} = (1 - a^2)\operatorname{Cov}[\boldsymbol{x}]$. Our resampling algorithm then involves drawing n new particles from the distribution given by (2) and setting the weight of each new particle to $1/n$.

We combine these prior algorithms to obtain Algorithm 4, which is our complete algorithm for adaptively designing experiments using the SMC approximation. Note that we have left unspecified here the choice of local optimizer; in practice, this will be chosen depending on what works for a given experimental model. Due to the simulation cost of optimization, Algorithms 3 and 4 allow for the setting of an additional parameter, `approx_ratio`, that controls the quality with which the utility function is calculated.

4 Region and Hyperparameter Estimation

In addition to providing an accurate estimate of the true model parameters for the system, it is important to be able to quantify the uncertainty in the estimated model parameters. This task can be achieved by finding a region $\hat{X}$ of the space of models such that $\Pr(\boldsymbol{x}_0 \in \hat{X})$ is maximized and such that the volume $\mathrm{Vol}(\hat{X})$ is minimized.

We make the problem of region estimate amenable to analysis by SMC by reducing it to a problem of estimating an expectation value. In particular, the probability of the true model being within a region can be expressed as $\Pr(\boldsymbol{x}_0 \in \hat{X}) = \mathbb{E}[1_{\hat{X}}]$, where $1_{\hat{X}}$ is the *indicator function* for $\hat{X}$. The expectation value of this indicator function can then be computed using SMC as $\mathbb{E}[1_{\hat{X}}] \approx \sum_i w_i 1_{\hat{X}}(\boldsymbol{x}_i) = \sum_{i,\boldsymbol{x}_i \in \hat{X}} w_i$.

Thus, by construction, any region containing particles of total weight at least r will have an approximate probability mass of at least r. We formalize this intuition by introducing a *probability mass* function $m(R)$ on regions R such that $m(R) = \mathbb{E}[1_R]$. Similarly, let $\tilde{m}(R) = \sum_{i,\ i \in R} w_i$ be an approximation of $m(R)$ using the SMC algorithm.

We thus seek a region $\hat{X}$ such that $\mathrm{Vol}(\hat{X})$ is small, $m(\hat{X})$ is large and such that $\hat{X}$ is an efficiently computable property of the current SMC state. We achieve the latter two properties by choosing some appropriate geometric function of a set of particles X_r whose weight is above some threshold weight r; for example, the convex hull or the minimum-volume enclosing ellipse of X_r both satisfy $\tilde{m}(X_r) \geq r$ and may be computed using well-known classical algorithms [27, 28].

In practice, the covariance matrix of the posterior distribution will often suffice as a region estimate because the posterior distribution will often be approximately normally distributed. This assumption holds when the Fisher information is non–singular. More generally, under the assumption of a normally distributed posterior, the error ellipse of points $\boldsymbol{x}$ satisfying

$$(\boldsymbol{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \leq Z^2 \tag{3}$$

for some $Z > 0$ will contain a ratio $(\mathrm{cdf}_{\mathcal{N}}(Z) - \mathrm{cdf}_{\mathcal{N}}(-Z))^d = \mathrm{erf}\left[\frac{Z}{\sqrt{2}}\right]^d$ of the particle weight, where $\mathrm{cdf}_{\mathcal{N}}(Z)$ is the cumulative distribution function for the normal distribution, evaluated at Z. Thus, if the assumption of a normal posterior is a good approximation, then the covariance matrix of the posterior distribution as approximated by SMC can be used as a region estimator.

We can generalize further by considering the fact that quantum systems seldom have consistent Hamiltonians from experiment to experiment, due to experimental errors. Thus, we would like to form a region estimate for such Hamiltonians that encompasses experiment-to-experiment variation, but that expands that region as little as possible. Hyperparameters allow us to address this by switching from the problem of estimating Hamiltonian parameters to one that involves learning the parameters that describe the *distribution* of Hamiltonian parameters.

We denote the hyperparameters for a model Hamiltonian as $\boldsymbol{y}$ to avoid subtle conceptual differences between the hyperparameters and the distributions on $\boldsymbol{x}$ that they describe. The

probability distribution for $\boldsymbol{x}$ can then be written as $\Pr(\boldsymbol{x}|\boldsymbol{y})$. Despite interpretational differences, the hyperparameters can also be learned using Algorithm 4 in exactly the same way that $\boldsymbol{x}$ is learned. The region estimates yielded by the algorithm are region estimations for $\boldsymbol{y}$ and, as we will show shortly, can easily be converted into region estimates for $\boldsymbol{x}$.

The drawback to this approach is that computations of the likelihood function can become much more expensive because it typically will have to be computed by sampling from the parameterized distribution. In some important special cases, this drawback can be avoided by analytically performing the marginalization over $\boldsymbol{x}$,

$$\Pr(D|\boldsymbol{y}) = \int \mathrm{d}\boldsymbol{x} \Pr(D|\boldsymbol{x}) \Pr(\boldsymbol{x}|\boldsymbol{y}).$$

In Section 5, we discuss a particular case where the marginalization is analytically tractable.

The resulting means and covariance matrices for $\boldsymbol{y}$ can be readily converted to the corresponding quantities for $\boldsymbol{x}$ by using the chain rule for expectation values,

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}[\boldsymbol{x}] = \mathbb{E}_{\boldsymbol{y}}[\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y}}[\boldsymbol{x}]]. \tag{4}$$

This expectation value can be computed using the posterior distribution $\Pr(\boldsymbol{y}|D)$ and the intermediate model distribution $\Pr(\boldsymbol{x}|\boldsymbol{y})$, which will typically be easy to compute from the definition of the hyperparameters. The covariance matrix for $\boldsymbol{x}$ is slightly more complicated. It is straightforward to verify that

$$\mathrm{Cov}_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{y}}\left[\mathrm{Cov}_{\boldsymbol{x}|\boldsymbol{y}}(\boldsymbol{x})\right] + \mathrm{Cov}_{\boldsymbol{y}}\left(\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y}}[\boldsymbol{x}]\right). \tag{5}$$

For the special case that $\boldsymbol{x}$ is a single parameter, the covariance can be replaced with the variance to obtain that

$$\mathrm{Var}_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{y}}\left[\mathrm{Var}_{\boldsymbol{x}|\boldsymbol{y}}(\boldsymbol{x})\right] + \mathrm{Var}_{\boldsymbol{y}}\left(\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y}}[\boldsymbol{x}]\right). \tag{6}$$

Using the covariance ellipse region estimate given by (3) to estimate a hyperparameter region thus translates to a region estimator for the model parameters $\boldsymbol{x}$, if the distribution over hyperparameters $\boldsymbol{y}$ is approximately Gaussian near its peak. In the limit of many experiments, we find that this is a good assumption, as is discussed in Section 5.3.

5 Single-Qubit Test Case

We will now proceed to apply our techniques to learning unknown parameters in a single qubit system. Our model has a qubit that evolves under an internal Hamiltonian of the form $H(\omega) = \frac{\omega}{2}\sigma_z$. Here ω is an unknown parameter whose value we want to estimate. An experiment consists of preparing a single known input state $\psi_{\mathrm{in}} = |+\rangle$, the $+1$ eigenstate of σ_x, evolving under H for time t and performing a measurement in the σ_x basis.

We will slightly generalize this model by allowing noise sources which lead to a decay in the information extractable from any measurement. This can manifest from, for example, a T_2 dephasing process which leads to the following likelihood function:

$$\Pr(0|\omega;t) = e^{-\frac{t}{T_2}} \cos^2\left(\frac{\omega}{2}t\right) + \frac{1 - e^{-\frac{t}{T_2}}}{2}, \tag{7}$$

where ω is the unknown parameter to be estimated, t is the controllable parameter and T_2 is a known constant. This model was studied in references [29, 30, 31] where analytical solutions based on Fisher information and the Cramer-Rao bound were given.

We consider the seemingly simple generalization of this model where both ω *and* T_2 are unknown. Even for such a simple generalization as this, the methods discussed in [29, 30, 31] are not adequate for this more general problem. In particular, the Fisher matrix of any one measurement is singular and hence the standard Cramer-Rao bound does not hold – nor is it possible to utilize standard asymptotic approximations to normal distributions.

This generalization is closely related to the case in which T_2 is infinite, but where the "true" precession frequency ω is itself distributed according to a Gaussian distribution of mean μ and variance σ^2. In this case, following the discussion of Section 4, the probability of data conditioned on the *hyperparameters* $\boldsymbol{y} = (\mu,\ \sigma)$ can be found by marginalizing over the intermediate random variable ω, so that

$$\Pr(d|\mu,\sigma;t) = \int \Pr(d|\omega)\Pr(\omega|\mu,\sigma)\mathrm{d}\omega. \tag{8}$$

For the specific example of the Gaussian distribution,

$$\Pr(0|\mu,\sigma;t) = \frac{1}{\sigma\sqrt{2\pi}}\int \cos^2\left(\frac{\omega t}{2}\right) e^{-\frac{(\omega-\mu)^2}{\sigma^2}}\mathrm{d}\omega = \frac{1}{2}\left(1 + e^{-2\sigma^2 t^2}\cos(2\mu t)\right). \tag{9}$$

At this point, we have entirely removed ω from the problem, leaving a two-parameter model, where we wish to estimate the mean and variance of an unknown normal distribution.

As another example, instead of marginalizing against a Gaussian distribution, we consider the case that the intermediate model parameter ω is drawn from a Lorentz distribution. A Lorentz distribution is completely determined by its location and scale parameters ω_0 and γ, respectively, and so we use these hyperparameters to derive a new model,

$$\Pr(0|\omega_0,\gamma;t) = \int \cos^2(\omega t/2)\frac{1}{\pi\gamma\left(\frac{(\omega-\omega_0)^2}{\gamma^2}+1\right)}\mathrm{d}\omega = \frac{1}{2}\left(1 + e^{-t\gamma}\cos\left(t\omega_0\right)\right). \tag{10}$$

Note that if we identify $\gamma = T_2^{-1}$, then the Lorentz hyperparameter model is the identical to that of Equation (7). This illustrates the relationship between decoherence processes and the lack of knowledge formalized by a hyperparameter model. In a similar fashion, (9) is also model of decoherence. Due to the t^2 dependence of the Gaussian-hyperparameter model, (9) represents a decoherence process that cannot be written in Lindblad form [32] because it cannot be drawn from a quantum dynamical semigroup.

5.1 Results for Unknown T_2

Here we report on the performance of our algorithm for the comparatively challenging task of learning Hamiltonian parameters without a precise estimate of T_2. These calculations were performed using the true distributions $\omega \sim \mathcal{N}(0.5, 0.0025)$ and $1/T_2 \sim \mathcal{N}(0.001, 0.00025^2)$, and with the scale matrix $\boldsymbol{Q} = \mathrm{diag}(1, 0.0025/0.00025^2) = \mathrm{diag}(1, 100)$.

The guess heuristic that we focus on chooses times randomly from an exponential distribution with mean 1 000, corresponding to the mean value of T_2 according to the initial prior. This choice of guess function is motivated by the fact that the most informative experiments (as measured by Fisher information) tend to occur at $t \approx T_2$ [31]. A secondary benefit is that the guess function is certainly not optimal for the problem, and will allow us to illustrate that a sub–optimal guess function can be used in concert with local optimization (in our case Newton conjugate gradient optimization (NCG) is used) to find near optimal experiments given the current state of knowledge about the unknown Hamiltonian.

We examine the variation of the MSE with the number of guesses used in Figure 1. The figure shows that, in the absence of local optimization of experiment times, the MSE for both

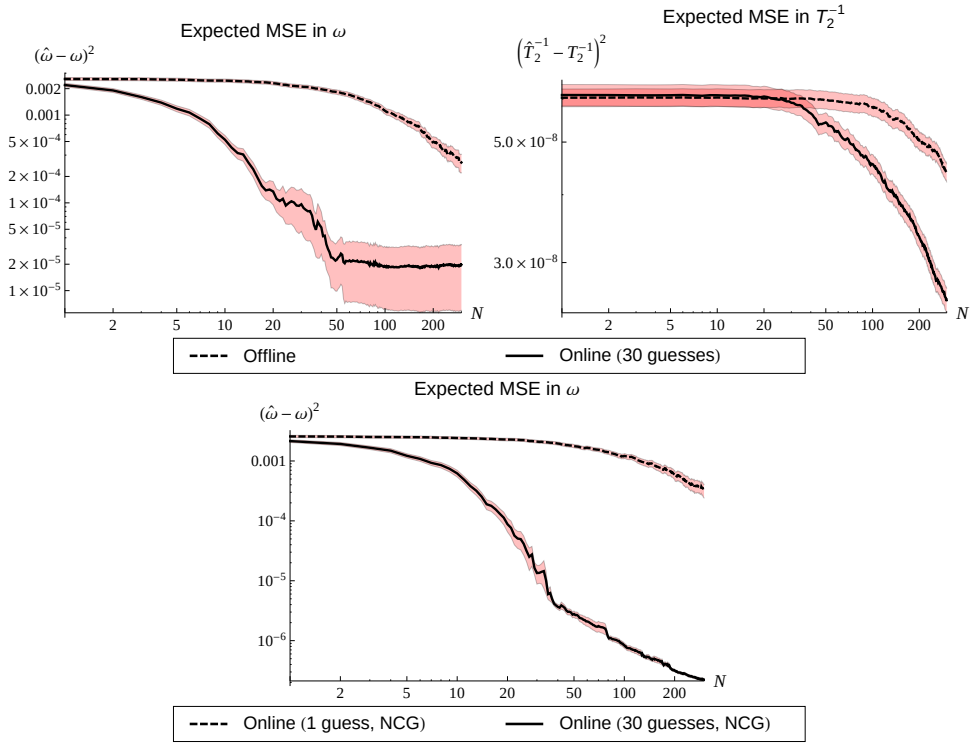

Figure 1 Benchmarking of the "unknown-T_2"model using $n = 5\,000$ particles and random initial guesses without local optimization. Data indicated dashed lines correspond to trials where a single initial guess was used for each experiment, while data indicated by solid lines were collected using 30 guesses per experiment. Errors in estimating performance are indicated by red shaded regions about each curve.

ω and $1/T_2$ is significantly improved by using an increased number of guesses. In particular, we find that if 30 guesses are used, then only 50 experiments are required on average to learn ω within a 0.9% error, even without a well characterized T_2. The improvement is much more substantial for ω than it is for $1/T_2$ because the contrast on T_2 is much less significant.

Figure 1 examines the effect of increasing the number of guesses for strategies that use NCG. The most significant qualitative difference between the data collected using NCG and that of Figure 1 is that the MSE for ω shows no evidence of saturating and instead continues to shrink as the number of experiments are increased (as seen most clearly in Figure 2). This implies that our randomized guess heuristic is unlikely to randomly guess very informative experiments after a fixed number of experiments, but the landscape is sufficiently devoid of local optima that NCG optimization finds informative experiments in the vicinity of our uninformed guesses. We also observe that NCG does not substantially improve the MSE if 1 guess is used. This suggests that the landscape is not sufficiently convex that local optimization about an individual guess is likely to find experiments that are substantially more informative. We therefore conclude that increasing number of guesses used and using NCG substantially improves the MSE for ω and has a much more subtle effect on the knowledge of T_2 if local optimization is used.

It is useful to benchmark the performance of our algorithm against the Bayesian Cramer-Rao bound (BCRB—see appendix), which gives a lower bound on the MSE. Figure 3 provides

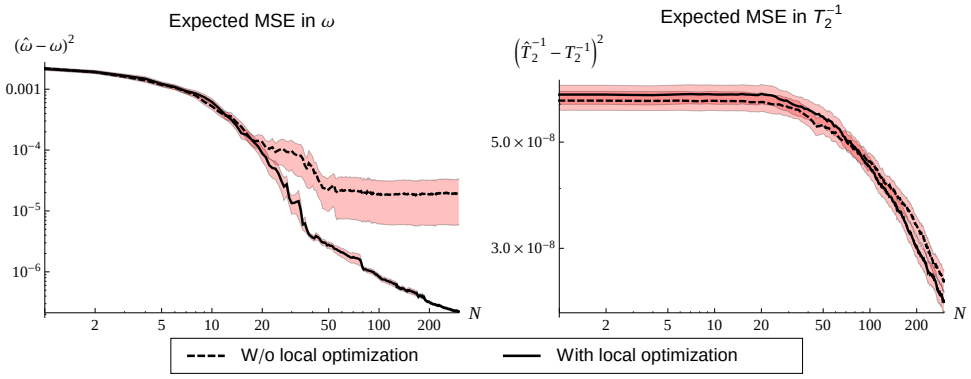

Figure 2 Benchmarking of the "unknown-T_2"model using $n = 5\ 000$ particles and 30 random initial guesses. Data indicated dashed lines correspond to trials where a each initial guess was used without local optimization, while data indicated by solid lines were collected using NCG optimization for each guess. The unoptimized data is averaged over 1,109 trials while the optimized data is averaged over 930 trials. Errors in estimating performance are indicated by red shaded regions about each curve.

a comparison of the MSE, the estimate of the MSE given by the variance of the posterior and the BCRB for ω, T_2^{-1} and $\mathrm{Tr}(\Sigma \cdot \boldsymbol{Q})$. We see that the expected posterior variance is typically within statistical error of the MSE for all three of these quantities, suggesting that the posterior variance can be used as a very good estimate of the MSE for this model. We also note that the MSE is very close to the MSE for the T_2^{-1} data and $\mathrm{Tr}(\Sigma \cdot \boldsymbol{Q})$. The MSE for ω is within a constant multiple of the BCRB. We do not, in fact, expect that the MSE in ω should approach the BCRB because the algorithm chooses experiments to optimize $\mathrm{Tr}(\Sigma \cdot \boldsymbol{Q})$ rather than the error for either ω or T_2^{-1} individually.

5.2 Region Estimation

One of the most substantial contributions of our algorithm is its ability to provide region estimates for the location of the true Hamiltonian, which allow us to quantify our uncertainty in the true model parameters. We compare the probability mass enclosed by the covariance region estimator described in Section 4. A simplifying assumption is made in our analysis: we assume that the posterior distribution is approximately Gaussian. Although difficult to justify theoretically, we have find for the examples that we consider that the posterior appears Gaussian locally around our estimate after a sufficiently large number of experiments. We expect this behavior to be generic, although region estimators such as the convex hull or the minimum-volume enclosing ellipse may be used even if the posterior is not approximately normal.

Under the Gaussian model of the posterior distribution, we expect the true model parameters to be within an ellipse described by the covariance matrix whose volume is then described by the Z–score used. For example, in the one–dimensional case approximately 95% of the probability mass is located within 2–standard deviations, which corresponds to $Z = 2$. We choose $Z = 3$ standard deviations from the mean for these examples which correspond to probability masses of $\tilde{m}(\mathrm{Cov}(\hat{\boldsymbol{x}})^{-1}/Z^2) \approx 0.9973$ and $\tilde{m}(\mathrm{Cov}(\hat{\boldsymbol{x}})^{-1}/Z^2) \approx 0.9946$ for the one– and two–parameter cases respectively.

We show in Figure 4, and [33], that the approximate probability mass $\tilde{m}$ approaches the probability mass we would expect for a normal distribution for the known-T_2 model in

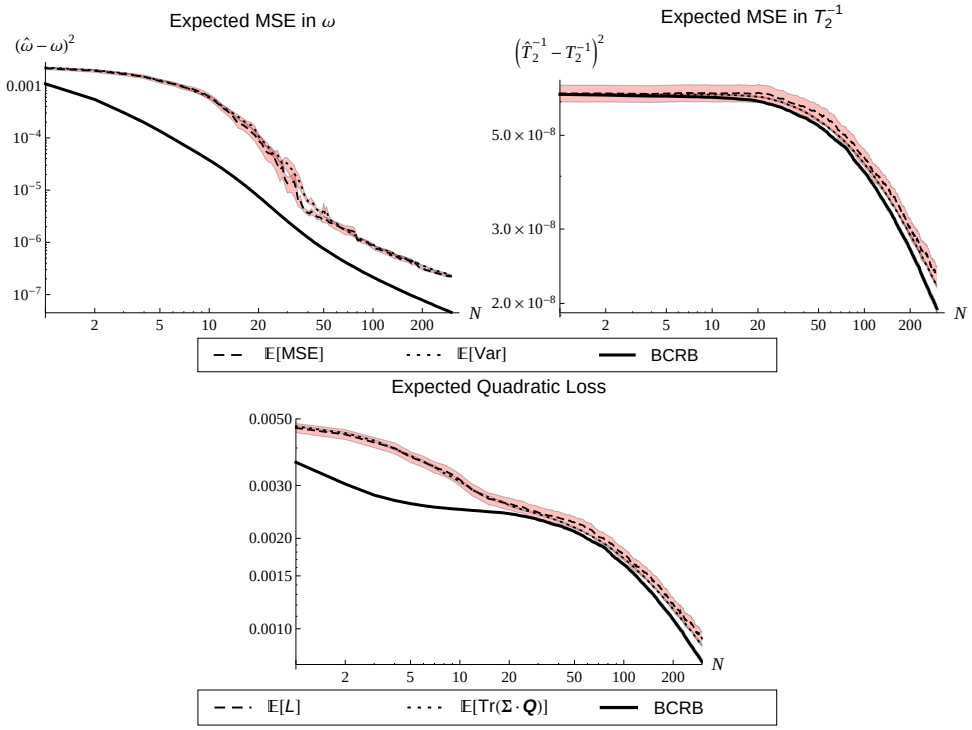

Figure 3 The actual and estimated performance, as a function of the number of measurements N, of the sequential Monte Carlo algorithm for $n = 5\,000$ particles. The model is that of equation (7) with unknown T_2 (which is estimated as $\Gamma = 1/T_2$ for numerical precision considerations). The dotted curve is the posterior variance of the particles; dashed is the actual mean squared error and solid is numerically calculated Bayesian Cramer-Rao lower bound. In the upper subfigures, the MSE and variances are those of the individual parameters ω and T_2^{-1}, respectively, while the lower subfigure shows the actual and estimated quadratic losses scaled using $\boldsymbol{Q} = \mathrm{diag}(1, \sigma_\omega^2/\sigma_{T_2^{-1}}^2)$, where σ_ω^2 and $\sigma_{T_2^{-1}}^2$ are the variances in ω and T_2^{-1} according to the initial prior π.

the limit of large N, providing evidence in favor of our use of the covariance ellipse as a region estimator on the posterior. In particular, we note that the value of $\tilde{m}$ approaches 0.9973, such that the quality of the Gaussian approximation improves as we collect data. The transient behavior for small experiment numbers occurs because insufficient experiments have been considered for the posterior to approach a Gaussian. In this specific example, the average differences in enclosed probability mass after each experiment are on the order of 0.01%, and thus may not be of practical significance.

5.3 Hyperparameter Region Estimation Performance

Having demonstrated the effectiveness of our region estimation algorithm, it remains to show that the generalization to hyperparameter regions works as described in Section 4. The objective here is to analyze the robustness of our algorithm in the presence of fluctuating "true" parameters of the Hamiltonian. We do so by using the Gaussian hyperparameter model as discussed in Section 5, then comparing the model parameter region volume and probability mass for the region estimated from Equation (5) to the volume and probability mass of the corresponding "true" model parameter region. We benchmark this model by

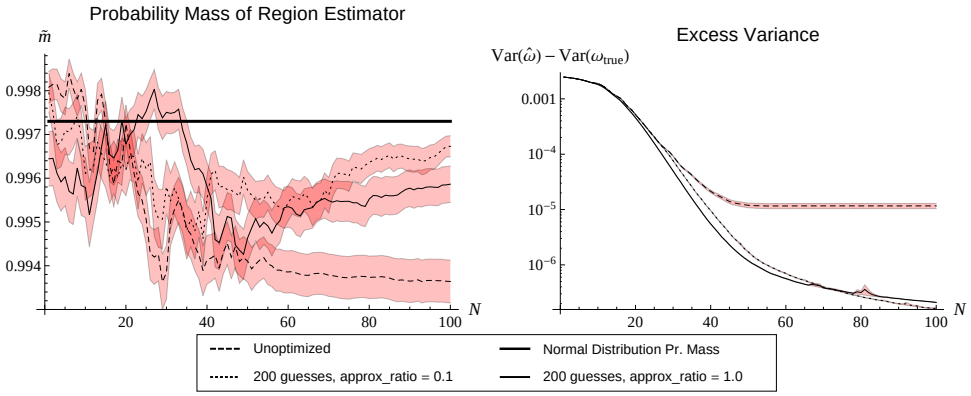

Figure 4 Benchmarking region estimators for Gaussian hyperparameter model using $n = 2\,000$ particles, $\omega \sim \mathcal{N}(\mu, \sigma^2)$ where $\mu \sim \mathcal{N}(0.5, 0.001^2)$ and $\sigma^2 \sim \mathcal{N}(0.0025, 0.0025^2)$.

choosing "true" hyperparameters μ and σ^2 for ω according to the normal distribution

$$\mu, \sigma^2 \sim \mathcal{N}\left[(\mu_\mu, \mu_{\sigma^2}), \mathrm{diag}(\sigma_\mu^2, \sigma_{\sigma^2}^2)\right]. \tag{11}$$

Recall that the unknown frequency is distributed as $\omega \sim \mathcal{N}(\mu, \sigma^2)$. In particular, this true distribution does not admit any correlation between the mean and variance hyperparameters. We then use the true distribution as our prior distribution.

In Figure 4, we find that the probability mass contained within our estimated region for the Hamiltonian agrees well with our theoretical expectations. In particular, we assume a Gaussian posterior and use a Z-score of 3 which implies that we should anticipate that 99.7% of the probability mass will lie within the region estimation of $\mathbb{E}[\hat{\omega}] \pm 3\sqrt{\mathrm{Var}(\hat{\omega})}$. We find very good agreement with this assumption, and find that at worst 99.4% of the probability mass for the hyperparameters lies within the estimated region. The data also suggests that these small differences vanish for the optimized data sets, which appear to approach the ideal enclosed probability mass of 99.7% in the limit of large N.

Hyperparameters are not typically a quantity of interest by themselves. They usually are of relevance because they parameterize a distribution of the unknown parameter. Following Equation (6), we calculate $\mathrm{Var}(\hat{\omega})$ as $\mathrm{Var}(\hat{\omega}) = \mathrm{Var}(\hat{\mu}) + \mathbb{E}[\hat{\sigma}^2]$. We find that, as the number of experiments grows, our region estimator for ω slightly overestimates the "true" variance of ω (on average). This bias vanishes as the number of experiments increases. We can therefore conclude that we can use the method of hyperparameters to robustly estimate the distribution of an unknown frequency, even in the presence of noise.

5.4 Computational Cost

Another way that we can assess the cost of inferring the Hamiltonian of a system is in terms of the classical computing time needed to learn the Hamiltonian parameters to within a fixed error tolerance (as measured by the number of likelihood calls made). Our previous discussion found that the experimental time (measured by the number of experiments) can be minimized by choosing measurements that minimize the risk, and showed that increasingly sophisticated heuristics for generating these guesses tended to reduce the experimental time. This suggests that a trade-off may be present between the experimental time and the classical processing time needed to learn the parameter. This tradeoff will become increasingly relevant as the size of the quantum system grows, since existing quantum simulation

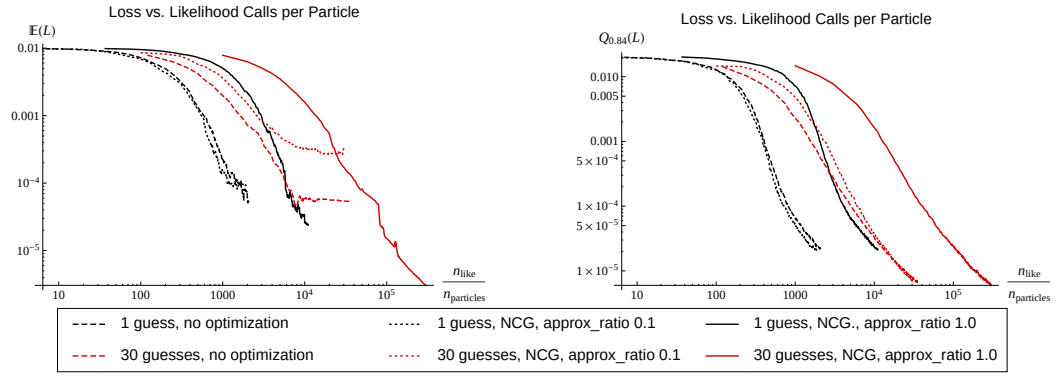

Figure 5 This figure compares the mean–square error as a function of the computational time for the known T_2 model with $T_2 = 100$, 5 000 particles, `approx_ratio` $= 1$ and guessed experimental times chosen randomly from an exponential distribution with mean T_2. The expected loss incurred by each optimization strategy is shown in the left figure and the figure on the right shows the 84^{th} percentile $Q_{0.84}$ of the loss, such that no more than 16% of trials incur loss greater than the shown percentile.

techniques do not scale efficiently with the number of particles in the system and thus the cost of performing a likelihood call may asymptotically become much more expensive than performing an experiment.

If computational time is of primary importance (rather than experimental time), then the relative merits of the experimental design heuristics changes. In total, our data sets in Figure 5 required (on average) a number of likelihood calls that fell within the range $[1.05 \times 10^7, 1.5 \times 10^9]$. A likelihood call required the evaluation of $\exp(-t/T_2)\cos^2\left(\frac{\omega}{2}t\right) + (1 - \exp(-t/T_2))/2$, which required time on the order of 10^{-7} seconds on our computers and lead to total computational times that were on the order of a second to a minute. If the rate at which experiments can be performed were much faster than 200 Hz then the utility of our algorithm as a means to speed up data collection may be lost. If the two rates are approximately comparable, then interesting trade-offs appear between the computational time needed and the total experimental time.

These trade-offs become apparent by plotting the scaling of the MSE as a function of the computational time for the randomized guess heuristic in Figure 5. The first feature that is obvious from the plot is that the strategies which yielded the lowest MSE per experiment tend to yield the highest MSE per likelihood call; although several of these strategies cause the expected loss (mean–square error) to saturate after a finite number of experiments. In particular, this causes the strategy with 30 guesses and no optimization as well as the strategy with 30 guesses, NCG optimization and `approx_ratio` $= 0.1$ to intersect the curve for the cases with NCG optimization and `approx_ratio` $= 1$. Here the approximation ratio is the ratio of the particles that are used in the updating (see Algorithm 3). On the surface, this seems to indicate that the more expensive heuristics may have an advantage if small loss is desired; but this is misleading and to get a complete picture we need to look at more than just the expected performance of the strategies.

We can get a better understanding of this saturation by looking at the plot of the 84^{th} percentile of the loss in Figure 5, which shows that all of these strategies continue to provide improved estimates of ω even into this regime of saturation for at least 84% of the trials considered. This shows that there were a few trials where very poor guesses were chosen and the algorithm became stuck at a large MSE. The data also suggests that the use of NCG and

a large value of the approximation ratio can mitigate these problems, causing the learning algorithm to become more stable at the price of requiring more computational time.

6 Multi–Qubit Test Case

We will now focus on an example that shows the viability of our algorithm in cases where the Hamiltonian acts on many qubits rather than just one. The model that we consider is the Ising Model with no external magnetic field with a complete graph of interactions on n qubits:

$$H = \sum_{i>j} x_{i,j} \sigma_i^z \sigma_j^z, \tag{12}$$

where $x_{i,j}$ are real valued coefficients. In these examples, we choose these coefficients randomly from the interval $[0, 1]$ (the absence of frustrations does not affect the difficulty of the learning problem). The goal of the learning problem is to learn each $x_{i,j}$. We represent these parameters for the Hamiltonian H_k using the vector $\boldsymbol{x}_k$.

In direct analogy to the single qubit examples, an experiment involves setting the initial state to be $|+\rangle^{\otimes n}$, evolving the state under the Hamiltonian in (12) for some evolution time t and finally applying the Hadamard transform to each qubit and measuring the result in the computational basis. The evolution time is the only control parameter in these experiments.

We also use a different guess heuristic to choose t for this problem than the exponential random guess heuristic that was used in previous experiments since, most notably, we are not considering decoherence processes. We choose the times by drawing two Hamiltonians H_j and $H_{j'}$ from the current prior $\Pr(H_k|d_n, \ldots; c_n, \ldots)$ and then choose $t = 1/|\boldsymbol{x}_j - \boldsymbol{x}_{j'}|_2$, rather than choosing the times randomly and using NCG to find a locally optimal experiment near that guess. We call this strategy the "particle guess heuristic." Although it may not seem it, this strategy is adaptive. In particular, the particle guess heuristic will tend to choose experiments that have short evolution times when the posterior distribution is broad, and longer evolution times when the distribution is narrow. Long evolution times are needed to distinguish dynamics of nearby Hamiltonians, thus the heuristic adaptively chooses experiments that will be informative based on the current uncertainty in the unknown parameters. We pick this strategy because it outperforms the exponential guessing strategy for such problems, especially in absentia of local optimization.

We avoid local optimization in these numerical experiments because the effects of incorporating local optimization have been well discussed in previous examples and the improvements brought about by using local optimization in these cases is qualitatively similar to the single qubit case. Furthermore, the cost of computing the likelihood in these cases is substantially higher so including local optimization would only restrict the range of numerical examples that we could provide.

We examine the scaling of the quadratic loss that occurs when using SMC to learn Hamiltonian parameters for (12), using $\mathbf{Q} = \mathbb{1}$, in Figure 6. It is clear that our algorithm is capable of learning parameters of many–qubit Hamiltonians. The scaling of the quadratic loss is, similar to the single qubit case, exponential in the number of experiments taken. The slower rate of learning for the $n = 5$ and $n = 6$ cases is due largely to the fact that these cases have 10 and 15 unknown parameters that must be learned, in contrast to the 6 that must be learned in the $n = 4$ case. This shows that our algorithm is capable of learning Hamiltonian parameters in not just single qubit cases, but also in multi–parameter estimation problems that are relevant in real world applications such as characterizing superconducting quantum devices or certifying an analog quantum simulator.

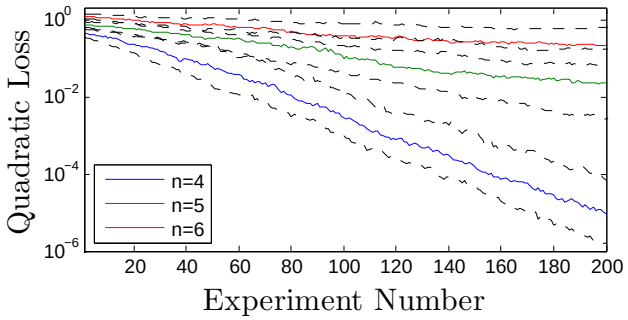

Figure 6 This figure compares the quadratic loss as a function of the computational time for the Ising model with 20 000 particles, `approx_ratio` = 1 and guessed experimental times chosen using the particle guess heuristic. The dashed lines represent the 25^{th} and 75^{th} percentile of the quadratic loss, whereas the solid lines represent the median.

7 Conclusions

Our work provides a simple algorithm that applies Bayesian inference to learn a Hamiltonian in an online fashion; that is to say, that our algorithm learns the Hamiltonian parameters as the experiment proceeds rather than collecting data and inferring the Hamiltonian through post–processing. This eliminates the need to store and process gigabytes of data that are recovered from even relatively short experiments. Our work has several advantages over existing approaches to learning Hamiltonian parameters. First, it can be used to estimate the optimal parameterization of the dynamics of an arbitrary quantum system within a space of model Hamiltonians. Second, it can be used to provide a region estimatate of the Hamiltonian parameters. The importance of this is obvious: it allows us to not only learn the unknown parameters but also quantify our uncertainty in them. Third, our analysis of the algorithm shows a clear trade off between the experimental time and the computational time needed to parameterize the Hamiltonian.

We note a natural extension of our algorithm to include classical simulators which do not deterministically compute the likelihood function but generate random samples according to it [34]. The distinction between *strong* and *weak* simulation has been a topic of recent interest in computational complexity [35, 36]. The present work and that of [34] add to the discussion of this distinction by clarifying the relationship between simulating a physical model classically and estimating the parameters in it.

An extension of our work would be to consider more advanced optimization heuristics than conjugate gradient searches (such as particle swarm optimization algorithms). Similarly, more advanced resampling techniques may lead to substantial reductions in the number of particles which in turn would reduce the computational cost of the algorithm. Finally, estimates of how the number of experiments required to achieve a specific mean–square error scales with the number of unknown parameters would be an important extension of this work since it would assess the viability of these techniques for controlling and characterizing larger quantum systems.

Acknowledgements. This work was financially supported by the Canadian government through NSERC and CERC and by the United States government through DARPA. NW would like to acknowledge funding from USARO-DTO.

References

1 B. P. Lanyon, C. Hempel, D. Nigg, M. M¨ uller, R. Gerritsma, F. Zähringer, P. Schindler, J. T. Barreiro, M. Rambach, G. Kirchmair, M. Hennrich, P. Zoller, R. Blatt, and C. F. Roos. Universal digital quantum simulation with trapped ions. *Science* **334** 57, 2011.

2 R. Gerritsma, B. P. Lanyon, G. Kirchmair, F. Zähringer, C. Hempel, J. Casanova, J. J. Garcia-Ripoll, E. Solano, R. Blatt, and C. F. Roos. Quantum simulation of the klein paradox with trapped ions. *Physical Review Letters* **106** 060503, 2011.

3 K. Kim, M.-S. Chang, S. Korenblit, R. Islam, E. E. Edwards, J. K. Freericks, G.-D. Lin, L.-M. Duan, and C. Monroe. Quantum simulation of frustrated Ising spins with trapped ions. *Nature*, **465** 590, 2010.

4 Matteo Paris and Jaroslav Rehacek, editors. *Quantum State Estimation*, volume 649 of *Lecture Notes in Physics*. Springer, 2004.

5 Ariel Bendersky, Fernando Pastawski, and Juan Pablo Paz. Selective and efficient estimation of parameters for quantum process tomography. *Physical Review Letters* **100** 190403, 2008.

6 Ariel Bendersky, Fernando Pastawski, and Juan Pablo Paz. Selective and efficient quantum process tomography. *Physical Review A* **80** 032116, 2009.

7 M. Mohseni and A. T. Rezakhani. Equation of motion for the process matrix: Hamiltonian identification and dynamical control of open quantum systems. *Physical Review A* **80** 010101, 2009.

8 M P A Branderhorst, J Nunn, I A Walmsley, and R L Kosut. Simplified quantum process tomography. *New Journal of Physics* **11** 115010, 2009.

9 Steven T. Flammia and Yi K. Liu. Direct Fidelity Estimation from Few Pauli Measurements. *Physical Review Letters* **106** 230501, 2011.

10 Marcus P. da Silva, Olivier L. Cardinal, and David Poulin. Practical Characterization of Quantum Devices without Tomography. *Physical Review Letters* **107** 210404, 2011.

11 Arnaud Doucet and Adam M. Johansen. A Tutorial on Particle Filtering and Smoothing: Fifteen Years Later. In *The Oxford Handbook of Nonlinear Filtering*. Oxford University Press, 2009.

12 Thomas J. Loredo. Bayesian Adaptive Exploration. *AIP Conference Proceedings* **707** 330, 2004.

13 Hendrik Kuck, Nando de Freitas, and Arnaud Doucet. SMC Samplers for Bayesian Optimal Nonlinear Design. In *Nonlinear Statistical Signal Processing Workshop*. IEEE, 2006.

14 Bruno Scarpa and David B. Dunson. Bayesian methods for searching for optimal rules for timing intercourse to achieve pregnancy. *Statistics in Medicine* **26** 1920, 2007.

15 D. R. Cavagnaro, M. A. Pitt, and J. I. Myung. Adaptive Design Optimization in Experiments with People. *Advances in Neural Information Processing Systems* **22** 234, 2010.

16 N. Kantas, A. Lecchini-Visintini, and J. M. Maciejowski. Simulation-based Bayesian optimal design of aircraft trajectories for air traffic management. *International Journal of Adaptive Control and Signal Processing* **24** 882, 2010.

17 Xun Huan and Youssef M. Marzouk. Simulation-based optimal Bayesian experimental design for nonlinear systems. *Journal of Computational Physics* **232** 288, 2013.

18 F. Huszár and N. M. T. Houlsby. Adaptive Bayesian quantum tomography. *Physical Review A* **85** 052120, 2012.

19 Rocco A. Servedio and Steven J. Gortler. Equivalences and Separations Between Quantum and Classical Learnability. *SIAM Journal on Computing* **33** 1067, 2004.

20 Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. Machine Learning in a Quantum World Advances in Artificial Intelligence. volume 4013 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2006.

21 Scott Aaronson. The learnability of quantum states. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, **463** 3089, 2007.

22 Alexander Hentschel and Barry C. Sanders. Machine Learning for Precise Quantum Measurement. *Physical Review Letters*, **104** 063603, 2010.

23 Kristen L. Pudenz and Daniel A. Lidar. Quantum adiabatic machine learning. *Quantum Information Processing* **12** 2027, 2013.

24 Alexander Hentschel and Barry C. Sanders. Efficient algorithm for optimizing adaptive quantum metrology processes. *Physical Review Letters*, **107** 233601, 2011.

25 Alexandr Sergeevich and Stephen D. Bartlett. Optimizing qubit Hamiltonian parameter estimation algorithms using PSO. *Proceedings of 2012 IEEE Conference on Evolutionary Computation* 1, 2012.

26 J. Liu and M. West. *Combined parameter and state estimation in simulation-based filtering.* Springer-Verlag, 2000.

27 Michael J. Todd and E. Alper Yıldırım. On Khachiyan's algorithm for the computation of minimum-volume enclosing ellipsoids. *Discrete Applied Mathematics* **155** 1731, 2007.

28 C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software* **22** 469, 1996.

29 Alexandr Sergeevich, Anushya Chandran, Joshua Combes, Stephen Bartlett, and Howard Wiseman. Characterization of a qubit Hamiltonian using adaptive measurements in a fixed basis. *Physical Review A* **84** 052315, 2011.

30 Christopher Ferrie, Christopher E. Granade, and D. G. Cory. Adaptive hamiltonian estimation using bayesian experimental design. *AIP Conference Proceedings* **1443** 165, 2012.

31 Christopher Ferrie, Christopher Granade, and D. Cory. How to best sample a periodic probability distribution, or on the accuracy of Hamiltonian finding strategies. *Quantum Information Processing* **12** 611, 2013.

32 G. Lindblad. On the generators of quantum dynamical semigroups. *Communications in Mathematical Physics* **48** 119, 1976.

33 Christopher E Granade, Christopher Ferrie, Nathan Wiebe, and D G Cory. Robust online Hamiltonian learning. *New Journal of Physics* **14** 103013, 2012.

34 Christopher Ferrie and Christopher E Granade. Likelihood-free quantum inference: tomography without the born rule, URL http://arxiv.org/abs/1304.5828, 2012.

35 Scott Aaronson and Alex Arkhipov. The Computational Complexity of Linear Optics, URL http://arxiv.org/abs/1011.3245, 2010.

36 M. Van den Nest. Simulating quantum computers with probabilistic methods. *Quantum Information & Computation* **11** 784, 2011.

37 Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

38 E. L. Lehmann and George Casella. *Theory of Point Estimation.* Springer, 2nd edition, 1998.

39 James O. Berger. *Statistical Decision Theory and Bayesian Analysis.* Springer, 2nd edition, 1985.

40 Robin Blume-Kohout and Patrick Hayden. Accurate quantum state estimation via "Keeping the experimentalist honest", URL http://arxiv.org/abs/quant-ph/0603116, 2006.

41 Richard D. Gill and Boris Y. Levit. Applications of the van Trees Inequality: A Bayesian Cramér-Rao Bound. *Bernoulli* **1** 59, 1995.

42 P. Tichavsky, C. H. Muravchik, and A. Nehorai. Posterior Cramer-Rao bounds for discrete-time nonlinear filtering. *IEEE Transactions on Signal Processing* **46** 1386, 1998.

A Utility Functions and the Cramer-Rao Lower Bound

Given a set of observed outcomes, the choice of subsequent experimental parameters that informs us most about the model parameters is given by the *utility function*. We test our method with a utility function that minimizes the expected variance in $\Pr(\boldsymbol{x}|d_{N+1}, D; c_{N+1}, C)$. We show that this choice is optimal for minimizing the the mean squared error of the protocol.

An *estimator* is a function $\hat{\boldsymbol{x}}$ that takes a set of observed data D collected from a set of experiments with controls C and produces an estimate for the unknown parameters $\boldsymbol{x}$. Here, we evaluate the quality of an estimator $\hat{\boldsymbol{x}}$ by using a generalization of the *squared error loss* called the *quadratic loss* as our figure of merit. The quadratic loss is defined for a vector of parameters $\boldsymbol{x}$, data D and experiment designs C, as

$$L_{\boldsymbol{Q}}(\boldsymbol{x}, \hat{\boldsymbol{x}}(D, C)) = (\boldsymbol{x} - \hat{\boldsymbol{x}}(D, C))^{\mathrm{T}} \boldsymbol{Q} (\boldsymbol{x} - \hat{\boldsymbol{x}}(D, C)), \tag{13}$$

where $\boldsymbol{Q}$ is a positive definite matrix on the space of unknown parameters that defines the relative scale between the various parameters of interest. The quadratic loss function is useful to us in that it is computationally inexpensive to calculate and may be analyzed by well-known statistical techniques. In particular, the Cramer-Rao bound can be used to lower-bound the mean quadratic loss incurred by an estimator, under the hypothesis of a given true model $\boldsymbol{x}$ [38].

Following a decision theoretic methodology [39], the *risk* of an estimator given a set of experiment designs C is its expected performance over all possible outcomes D with respect to the loss function:

$$R(\boldsymbol{x}, \hat{\boldsymbol{x}};\ C) = \mathbb{E}_{D|\boldsymbol{x};C}[L(\boldsymbol{x}, \hat{\boldsymbol{x}}(D; C))].$$

The Bayes risk is the average of this quantity with respect to a prior distribution on $\boldsymbol{x}$ (denoted π) and is given explicitly by

$$r(\pi; C) = \mathbb{E}_{\boldsymbol{x}}[R(\boldsymbol{x}, \hat{\boldsymbol{x}};\ C)] = \int \pi(\boldsymbol{x}) R(\boldsymbol{x}, \hat{\boldsymbol{x}};\ C) \mathrm{d}\boldsymbol{x}.$$

where $\hat{\boldsymbol{x}}$ is assumed to be a *Bayes estimator*, which means it is the one which minimizes the Bayes risk. When the loss function is taken to be squared error (in the single parameter case) or the quadratic loss (in the multi-parameter case), the Bayes risk is more familiarly known as *mean squared error* (MSE).

For quadratic loss (and many others [40]) the unique Bayes estimator is the mean of the posterior distribution $\hat{\boldsymbol{x}}(D; C) = \mathbb{E}_{\boldsymbol{x}|D;C}[\boldsymbol{x}]$. Minimizing the Bayes risk of a choice of parameters is equivalent to maximizing the negative Bayes risk for that set; therefore, it is reasonable to choose the negative Bayes risk as our utility function. It also has theoretical benefits in that it is easy to compare the performance of algorithms that take $U(c_{N+1}) = -r(\pi; c_{N+1}, C)$.

The question of how well can we estimator $\boldsymbol{x}$ becomes the question of how low can we make the Bayes risk $r(\pi; C)$. We lower bound the achievable risk via the Bayesian variant of the Cramer-Rao bound [41]. Both require finding the Fisher information:

$$\boldsymbol{I}(\boldsymbol{x}; C) = \mathbb{E}_{D|\boldsymbol{x};C}\left[\boldsymbol{\nabla}_{\boldsymbol{x}} \log\left(\Pr(D|\boldsymbol{x}; C)\right) \cdot \boldsymbol{\nabla}_{\boldsymbol{x}}^{\mathrm{T}} \log\left(\Pr(D|\boldsymbol{x}; C)\right)\right].$$

The Fisher information does not depend at all on the prior distribution, and thus is calculated in the same way regardless of how many experiments have already been performed.

The standard Cramer-Rao bound is then given by $\mathrm{Cov}(\hat{\boldsymbol{x}}) \geq \boldsymbol{I}(\boldsymbol{x}; C)^{-1}$, where $\boldsymbol{X} \geq \boldsymbol{Y}$ means that $\boldsymbol{X} - \boldsymbol{Y}$ is positive semi-definite. If we choose the matrix $\boldsymbol{Q}$ associated with the quadratic loss to be $\boldsymbol{Q} = \mathbb{1}$, then $R(\boldsymbol{x}, \hat{\boldsymbol{x}}; C) = \mathrm{Tr}(\mathrm{Cov}(\hat{\boldsymbol{x}})) \geq \mathrm{Tr}(\boldsymbol{I}(\boldsymbol{x}; C)^{-1})$. Clearly, this statement of the multivariate Cramer-Rao bound assumes that $\boldsymbol{I}$ is non-singular. Singular Fisher information matrices arise when there are experiments that provide no information about *at least one* of the experimental parameters. Unfortunately, that assumption is not met in general. We avoid this problem by considering the Bayesian information matrix $\boldsymbol{J}(\pi; C) = \mathbb{E}_{\boldsymbol{x}}[\boldsymbol{I}(\boldsymbol{x}; C)]$. Then, the *Bayesian Cramer-Rao bound* (BCRB) is given by [41]

$$r(\pi; C) \geq \boldsymbol{J}(\pi; C)^{-1}.$$

Lower bounds can be found for specific values of C using numerical integration. In practice, we calculate the BCRB using an iterative method, similar to [42].

B Pseudo–Code for Algorithms

Algorithm 1 Sequential Monte Carlo update algorithm.

Input: Particle weights $w_i(D)$, $i \in \{1, \ldots, n\}$, Particle locations $\boldsymbol{x}_i$, $i \in \{1, \ldots, n\}$, New datum d_{j+1}, obtained from an experiment with control c_{j+1}.
Output: Updated weights $w_i(D \cup d_{j+1})$.

```
function UPDATE({w_i(D)}, {x_i}, d_{j+1}, c_{j+1})
    for i ∈ 1 → n do
        w̃_i ← w_i(D) Pr(d_{j+1} | x_i, c_{j+1})
    end for
    return {w̃_j / Σ_i w̃_i}        ▷ We must normalize the updated weights before returning.
end function
```

Algorithm 2 Sequential Monte Carlo resampling algorithm.

Input: Particle weights w_i, $i \in \{1, \ldots, n\}$, Particle locations $\boldsymbol{x}_i$, $i \in \{1, \ldots, n\}$, Resampling parameter $a \in [0, 1]$.
Output: Updated weights w_i' and locations $\boldsymbol{x}_i'$.

```
function RESAMPLE({w_i}, {x_i}, a)
    μ ← MEAN({w_i}, {x_i}), Σ ← h² COV({w_i}, {x_i})
    h ← √(1 − a²)
    for i ∈ 1 → n do
        draw j with probability w_j        ▷ Choose a particle j to perturb.
        μ_i ← a x_j + (1 − a) μ            ▷ Find the mean for the new particle location.
        draw x_i' from N(μ_i, Σ)           ▷ Draw a perturbed particle location.
        w_i' ← 1/n                         ▷ Reset the weights to uniform.
    end for
    return {w_i'}, {x_i'}
end function
```

Algorithm 3 Reduced particle approximation for Sequential Monte Carlo utility functions.

Input: Particle weights w_i, $i \in \{1, \ldots, n\}$, Particle locations $\boldsymbol{x}_i$, $i \in \{1, \ldots, n\}$, Ratio `approx_ratio` of the particles to keep in the reduced approximation.
Output: Reduced sets of particle weights $\{\tilde{w}_i\}$ and locations $\{\tilde{\boldsymbol{x}}_i\}$.

```
function REAPPROX({w_i}, {x_i}, approx_ratio)
    ñ ← ⌊n · approx_ratio⌋
    draw π uniformly at random from Sym(n), the symmetric group acting on n elements
    {w̃_i} ← {w_{π(i)}}        ▷ Permute the elements to avoid patterns when sorting the weights.
    {x̃_i} ← {x_{π(i)}}
    {s_k} ← SORT({w̃_i})        ▷ Get a list of indices s_i such that w̃_{s_i} ≥ w̃_{s_j} for all i, j.
    return {w̃_i} ← {w̃_{s_i} : i ∈ 1 → ñ}, {x̃_i} ← {x̃_{s_i} : i ∈ 1 → ñ}
end function
```

Algorithm 4 Complete adaptive Bayesian experiment design algorithm, using sequential Monte Carlo approximations.

Input: A number of particles n to be used, A prior distribution π over models, A number of experiments N to perform, A resampling parameter $a \in [0,1]$, A threshold `resample_threshold` $\in [0,1]$ specifying how often to resample, An approximation ratio `approx_ratio`, An local optimization algorithm LocalOptimize, A heuristic GuessExperiment for choosing experiment controls, and a number n_{guesses} of potential experiments to consider in each iteration.

Output: An estimate $\hat{\boldsymbol{x}}$ of the true model $\boldsymbol{x}_0$.

function EstimateAdaptive(n, π, N, a, `resample_threshold`, `approx_ratio`, Optimize, n_{guesses}, GuessExperiment)

$w_i \leftarrow 1/n$ ▷ Start by initializing the SMC variables.
draw each $\boldsymbol{x}_i$ independently from π

for $i_{\text{exp}} \in 1 \to N$ **do** ▷ We now iterate through each experiment.
▷ If we are using a reduced particle set, populate that first.
if `approx_ratio` $\neq 1$ **then**
$\{\tilde{w}_i\}, \{\tilde{\boldsymbol{x}}_i\} \leftarrow$ Reapprox($\{w_i\}$, $\{\boldsymbol{x}_i\}$, `approx_ratio`)
else
$\{\tilde{w}_i\}, \{\tilde{\boldsymbol{x}}_i\} \leftarrow \{w_i\}, \{\boldsymbol{x}_i\}$
end if

▷ Heuristicly choose potential experiments, and optimize each independently.
for $i_{\text{guess}} \in 1 \to n_{\text{guesses}}$ **do**
$c_{i_{\text{guess}}} \leftarrow$ GuessExperiment(i_{exp})
$\hat{c}_{i_{\text{guess}}}, U_{i_{\text{guess}}} \leftarrow$ LocalOptimize(Utility, $c_{i_{\text{guess}}}$, $\{\tilde{w}_i\}$, $\{\tilde{\boldsymbol{x}}_i\}$)
end for

$i_{\text{best}} \leftarrow \text{argmax}_{i_{\text{guess}}} U_{i_{\text{guess}}}$ ▷ Pick the controls that maximize the optimized utility.
$\hat{c} \leftarrow \hat{c}_{i_{\text{best}}}$
$d_{i_{\text{exp}}} \leftarrow$ the result of performing $\hat{C}$ ▷ Perform the best experiment.
$\{w_i\}, \{\boldsymbol{x}_i\} \leftarrow$ Update($\{w_i\}$, $\{\boldsymbol{x}_i\}$, D, C) ▷ Find the new posterior distribution.

if $\sum_i w_i^2 < N \cdot$ `resample_threshold` **then** ▷ Resample if n_{ess} is too small.
$\{w_i\}, \{\boldsymbol{x}_i\} \leftarrow$ Resample($\{w_i\}$, $\{\boldsymbol{x}_i\}$, a)
end if
end for

▷ After all experiments have been performed, return the mean as an estimate.
return $\hat{\boldsymbol{x}} \leftarrow$Mean($\{w_i\}$, $\{\boldsymbol{x}_i\}$)

end function