Robust online Hamiltonian learning

New Journal of Physics
The open-access journal for physics

Robust online Hamiltonian learning

Christopher E Granade[1,2,7], **Christopher Ferrie**[1,3], **Nathan Wiebe**[1,6] **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
[6] Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada
E-mail: cgranade@cgranade.com

New Journal of Physics **14** (2012) 103013 (31pp)
Received 9 July 2012
Published 3 October 2012
Online at http://www.njp.org/
doi:10.1088/1367-2630/14/10/103013

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. We design the algorithm with practicality in mind by including parameters that control trade-offs between the requirements on computational and experimental resources. 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.

[7] Author to whom any correspondence should be addressed.

Contents

1. Introduction

Building a large scale quantum information processor is a significant challenge. First, we require an accurate characterization of the dynamics experienced by the device to allow for the application of error correcting codes and other tools for implementing useful quantum algorithms. Characterization is carried out through the statistical estimation of parameters describing the quantum states and processes involved (also called *tomography*) [1]. This characterization problem is especially timely, since quantum simulation experiments are approaching a complexity where classical computers are unable to simulate their evolution [2–4]. In cases where so called analogue quantum simulation is applied, the validity of the simulation directly depends on the accuracy with which the Hamiltonian of the quantum simulation conforms to the dynamical model that the experimenter believes describes the simulator. At present, such simulators are certified by comparing their outputs to those expected from a classical-computer simulation [3, 4]. This means that new methods for Hamiltonian characterization will be vital for certifying the next generation of quantum simulators.

In quantum tomography, the usual scenario discussed has been *full* quantum process estimation—there really is a 'black box' that the experimenter knows nothing about. While conservative, this is highly unlikely in practice because physics typically gives some insight into the form of the Hamiltonian which gives rise to the process. This additional knowledge reduces the number of parameters of the process and processes for learning these parameters are sometimes called *partial* process estimation or partial tomography [5–10]. If our goal is to build a quantum information processing device, we must consider also an additional complication: a characterization of a process at a ‘snap-shot’ in time is not nearly as useful as a characterization of the dynamics a quantum system undergoes. The evolution of closed systems is given by a Hamiltonian operator, and hence this process is usually called Hamiltonian estimation in such cases.

One way to adapt the above schemes to Hamiltonian estimation is by stroboscopically estimating snap-shots of the process at fixed times and then use various algorithms to invert these to find the Hamiltonian via post-processing[8]. The key additional freedom we consider has received little attention to date, and is that the controls after some number of measurements can depend on the outcomes of those previous measurements. Hence, we call our derived strategies *adaptive* or *online*. Under its very broad definition, our method can be called *machine learning*; however, a more descriptive name is *sequential Monte Carlo (SMC) Bayesian experimental design*. The marriage of SMC methods [12] and Bayesian experimental design methods [13] has been considered very recently in a wide variety of classical contexts [14–18] and also for measurement adaptive quantum state tomography [19][9]. Other machine learning ideas have also been generalized to the quantum domain [21–27]. In this paper we present such an algorithm for learning dynamical parameters of quantum mechanical systems.

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. We often have in practice 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 vary between experiments.

This paper is organized as follows. In section 2 we review the formalism of Bayesian experimental design. Next, we discuss the statistical metrics we employ and the Cramer–Rao bound in section 3. Section 4 introduces the SMC algorithm. The test cases we use for the numerical experiments are described in section 7. In sections 5 and 6, we discuss the application of our algorithm to region estimation and hyperparameter estimation, respectively. We explore the implications of the numerical benchmarking results in section 8.

2. Experimental design formalism

The key element which interfaces quantum theory and machine learning is the equivalence of the *Born rule* from quantum theory and the *likelihood function* from statistics [28]. Each quantum mechanical problem specification produces a probability distribution $\Pr(d_k|\boldsymbol{x}; c_k)$, where d_k is the data obtained and c_k are the experimental designs (or *controls*) chosen for measurement k, and where $\boldsymbol{x}$ is a vector parameterizing the system of interest.

[8] See [11] and references therein.

[9] It is interesting to note, however, that online experimental design may be unnecessary (asymptotically requiring only one adaptive step) for the particular state tomography problem considered there [20].

Suppose we have performed experiments with control settings $C := \{c_1, c_2, \ldots, c_N\}$ and obtained data $D := \{d_1, d_2, \ldots, d_N\}$. The model specifies the likelihood function

$$\Pr(D|\boldsymbol{x}; C) = \prod_{k=1}^{N} \Pr(d_k|\boldsymbol{x}; c_k).$$

However, we are ultimately interested in $\Pr(\boldsymbol{x}|D; C)$, the probability distribution of the model parameters $\boldsymbol{x}$ given the experimental data. We achieve this using Bayes' rule

$$\Pr(\boldsymbol{x}|D; C) = \frac{\Pr(D|\boldsymbol{x}; C)\Pr(\boldsymbol{x})}{\Pr(D|C)},$$

where $\Pr(\boldsymbol{x})$ is the *prior*, which encodes any *a priori* knowledge of the model parameters. The final term $\Pr(D|C)$ can simply be thought as a normalization factor. We refer to this update procedure as batch processing because the experimental controls C do not depend on the observed data and hence the processing for the experiment design can be done offline (meaning after all data is collected).

An alternative to batch processing of given data is to adaptively choose the controls for the next experiment given the data from the past experiment. This idea can be formalized in various ways—the most natural for our purposes being called *Bayesian experimental design* [13]. For this we conceive of possible future data d_{N+1} obtained from a set of, possibly different, experimental controls c_{N+1}. The probability of obtaining this data can be computed from the distributions at hand via marginalizing over model parameters

$$\Pr(d_{N+1}|D; c_{N+1}, C) = \int \Pr(d_{N+1}|\boldsymbol{x}; c_{N+1})\Pr(\boldsymbol{x}|D; C)\mathrm{d}\boldsymbol{x}.$$

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

$$\Pr(d_{N+1}|D; c_{N+1}, C) = \mathbb{E}_{\boldsymbol{x}|D;C}[\Pr(d_{N+1}|\boldsymbol{x}; c_{N+1})], \tag{1}$$

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* $U(D; C)$, by the expected *utility* of an experiment

$$U(c_{N+1}) = \mathbb{E}_{d_{N+1}|D;c_{N+1},C}[U(d_{N+1}; c_{N+1})],$$

where $U(d_{N+1}; c_{N+1})$ is the utility we would derive if experiment c_{N+1} yielded result d_{N+1}. The choice of the utility function is motivated by the figure of merit that we want to optimize. We will consider two canonical choices: *information gain* and the negative *variance* and discuss them in detail in the subsequent section.

3. 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*. A generally well motivated measure of utility for scientific inference is *information gain* [31, 32]. In information theory, information is measured by the entropy

$$U(d_{N+1}; c_{N+1}) = \mathbb{E}_{\boldsymbol{x}|d_{N+1},D;c_{N+1},C}[\log\Pr(\boldsymbol{x}|d_{N+1}, D; c_{N+1}, C)].$$

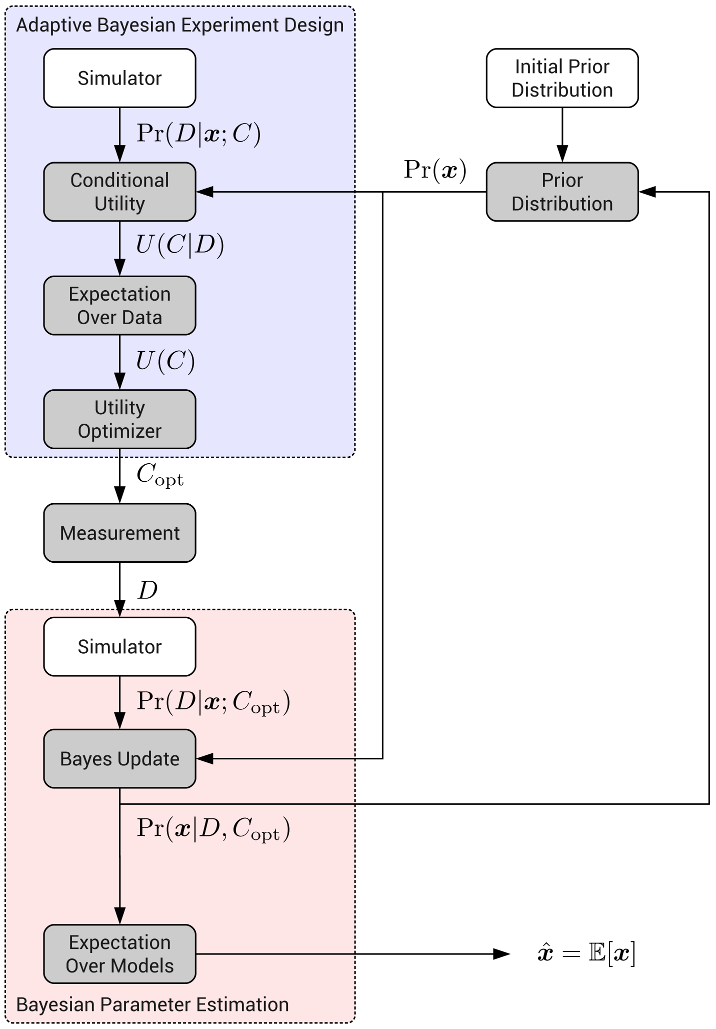

Figure 1. Overview of algorithm the combined experimental design and parameter estimation algorithm, showing the posterior distribution as a feedback into the next iteration. Blocks with a white background indicate model-dependent steps.

Maximizing the expected value of this utility function is equivalent to minimizing the expected entropy in the posterior distribution, $\Pr(\boldsymbol{x}|d_{N+1,}D;c_{N+1,},C)$. We also test or 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* $\hat{\boldsymbol{x}}$ 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

$$L_{Q}(\boldsymbol{x},\hat{\boldsymbol{x}}(D,C))=(\boldsymbol{x}-\hat{\boldsymbol{x}}(D,C))^{\mathrm{T}}\boldsymbol{Q}(\boldsymbol{x}-\hat{\boldsymbol{x}}(D,C)), \tag{2}$$

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 $\boldsymbol{x}$ [33].

Following a decision theoretic methodology [34], 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

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

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 MSE.

For quadratic loss (and many others [35]) 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 [36]. Both require finding the Fisher information. In the case of multiple parameters, the Fisher information is a matrix defined by

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

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 $\mathrm{Cov}(\hat{\boldsymbol{x}}) \geqslant \boldsymbol{I}(\boldsymbol{x}; C)^{-1}$, where $\boldsymbol{X} \geqslant \boldsymbol{Y}$ 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 $R(\boldsymbol{x}, \hat{\boldsymbol{x}}; C) = \mathrm{Tr}(\mathrm{Cov}(\hat{\boldsymbol{x}})) \geqslant \mathrm{Tr}(\boldsymbol{I}(\boldsymbol{x}; C)^{-1})$. 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 [36]

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

Lower bounds can be found for specific values of C using numerical integration. Here we apply an iterative algorithm[10] to sequentially monitor the lower bound. We will subscript the various quantities of interest by N, which means 'at the Nth measurement'. Then, the BCRB is given by

$$r(\pi; c_{N+1}) \geqslant \boldsymbol{J}_{N+1}(c_{N+1})^{-1}, \tag{3}$$

where the iteration is given by

$$\boldsymbol{J}_{N+1}(c_{N+1}) = \boldsymbol{J}(\pi; c_{N+1}) + \boldsymbol{J}_N(c_N)$$

and the initial condition is $\boldsymbol{J}_0 = \mathbb{E}_{\boldsymbol{x}}[\nabla_{\boldsymbol{x}} \log(\pi(\boldsymbol{x})) \cdot \nabla_{\boldsymbol{x}}^{\mathrm{T}} \log(\pi(\boldsymbol{x}))]$.

4. Sequential Monte Carlo algorithm

Monte Carlo methods remedy problems that deterministic numerical integrators encounter in finding the volume of multi-dimensional spaces. Specifically, the errors incurred by Monte Carlo integrators do not depend on the dimension of the space. The variant of Monte Carlo integration that we use dates back to 1993 [38] but has been rediscovered many times in a wide variety of scientific applications under the following names[11]: SMC, particle filters, sequential importance sampling, Bayes filters and so on. We will use the term SMC and will now sketch the idea behind the algorithm.

Recall the Bayes update rule for one datum d_1,

$$\Pr(\boldsymbol{x}|d_1) \propto \Pr(d_1|\boldsymbol{x}) \Pr(\boldsymbol{x}).$$

The distribution can be processed sequentially as more data arrive:

$$\begin{aligned} \Pr(\boldsymbol{x}|d_2, d_1) &\propto \Pr(d_2|\boldsymbol{x}) \Pr(\boldsymbol{x}|d_1), \\ \Pr(\boldsymbol{x}|d_3, d_2, d_1) &\propto \Pr(d_3|\boldsymbol{x}) \Pr(\boldsymbol{x}|d_2, d_1), \\ &\vdots \end{aligned}$$

These updates are difficult to perform because the evaluation of the posterior distributions require evaluations of the costly multi-dimensional integrals over the parameter space. We address this issue by using *SMC* methods, which approximate a distribution over parameters with a distribution that has support only over a finite number of points (often referred to as *particles*). The support of the distribution over these points is called the weight of the *particle*, and by convention the sum of all weights must be 1. More concretely, we approximate an arbitrary distribution by

$$\Pr(\boldsymbol{x}|D) \approx \sum_{k=1}^{n} w_k(D)\delta(\boldsymbol{x} - \boldsymbol{x}_k),$$

where the weights at each step are iteratively calculated from the previous step via

$$w_k(d_{j+1} \cup D) = \sum_{k=1}^{n} \Pr(d_{j+1}|\boldsymbol{x}_k) w_k(d_j).$$

[10] This type of algorithm has been given some attention recently for 'state space models' and classical signals with additive noise [37].

[11] See, for example, [12] for a recent tutorial on these methods.

Using this form of a SMC algorithm also has the advantage of ensuring that the prior and posterior distributions for the update always have support over a finite number of particles, which simplifies the analysis of our algorithm.

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 can compute expectation values and variances according to algorithms 1 and 2, and can perform Bayes updates according to algorithm 3.

Algorithm 1. Estimators for mean and covariance using SMC.

```
Input: Particle weights $w_i$, $i \in \{1, \ldots, n\}$.
Input: Particle locations $\boldsymbol{x}_i$, $i \in \{1, \ldots, n\}$.
Output: Approximations $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ of $\mathbb{E}[\boldsymbol{x}]$ and $\mathrm{Cov}(\boldsymbol{x})$.
  function MEAN($\{w_i\}$, $\{\boldsymbol{x}_i\}$)
      return $\boldsymbol{\mu} \leftarrow \sum_i w_i \boldsymbol{x}_i$
  end function
  function COV($\{w_i\}$, $\{\boldsymbol{x}_i\}$) ▷ Estimates the covariance matrix $\mathrm{Cov}(\boldsymbol{x}) = \mathbb{E}[\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}] - \mathbb{E}[\boldsymbol{x}]\mathbb{E}[\boldsymbol{x}]^{\mathrm{T}}$.
      $\boldsymbol{\mu} \leftarrow \sum_i w_i \boldsymbol{x}_i$
      $\boldsymbol{\Sigma} \leftarrow \sum_i w_i \boldsymbol{x}_i \boldsymbol{x}_i^{\mathrm{T}} - \mu\mu^{\mathrm{T}}$
      return $\boldsymbol{\Sigma}$
  end function
```

Algorithm 2. Estimator for arbitrary expectation values using SMC.

```
Input: Particle weights $w_i$, $i \in \{1, \ldots, n\}$.
Input: Particle locations $\boldsymbol{x}_i$, $i \in \{1, \ldots, n\}$.
Input: Function $f(\boldsymbol{x})$ of model parameters to average over.
Output: Approximation $\mathbb{E}[f(\boldsymbol{x})]$.
  function MEANFN($\{w_i\}$, $\{\boldsymbol{x}_i\}$, $f$)
      return $\sum_i w_i f(\boldsymbol{x}_i)$
  end function
```

SMC 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. 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

Algorithm 3. SMC update algorithm.

```
Input: Particle weights $w_i$, $i \in \{1, \ldots, n\}$.
Input: Particle locations $\boldsymbol{x}_i$, $i \in \{1, \ldots, n\}$.
Input: New datum $D$, obtained from an experiment with control $C$.
Output: Updated weights $w'_i$.
  function UPDATE($\{w_i\}$, $\{\boldsymbol{x}_i\}$, $D$, $C$)
      for $i \in 1 \to n$ do
          $\tilde{w}_i \leftarrow w_i \Pr(D|\boldsymbol{x}_i, C)$        ▷ The updated weights $\tilde{w}_i$ are unnormalized.
      end for
      $w'_j \leftarrow \tilde{w}_j / \sum_i \tilde{w}_i$        ▷ We must normalize the updated weights before returning.
      return $\{w'_j\}$
  end function
```

Algorithm 4. SMC resampling algorithm.

```
Input: Particle weights $w_i$, $i \in \{1, \ldots, n\}$.
Input: Particle locations $\boldsymbol{x}_i$, $i \in \{1, \ldots, n\}$.
Input: Resampling parameter $a \in [0, 1]$.
Output: Updated weights $w'_i$ and locations $\boldsymbol{x}'_i$.
  function RESAMPLE($\{w_i\}$, $\{\boldsymbol{x}_i\}$, $a$)
      $\boldsymbol{\mu} \leftarrow$ MEAN($\{w_i\}$, $\{\boldsymbol{x}_i\}$)
      $h \leftarrow \sqrt{1 - a^2}$
      $\boldsymbol{\Sigma} \leftarrow h^2$ Cov($\{w_i\}$, $\{\boldsymbol{x}_i\}$)
      for $i \in 1 \to n$ do
          draw $j$ with probability $w_j$        ▷ Choose a particle $j$ to perturb.
          $\boldsymbol{\mu}_i \leftarrow a\boldsymbol{x}_j + (1 - a)\boldsymbol{\mu}$        ▷ Find the mean for the new particle location.
          draw $\boldsymbol{x}'_i$ from $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma})$        ▷ Draw a perturbed particle location.
          $w_i \leftarrow 1/n$        ▷ Reset the weights to uniform.
      end for
      return $\{w'_i\}$, $\{\boldsymbol{x}'_i\}$
  end function
```

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 [39].

The resampling algorithm we use was first proposed in [39] and is given explicitly in algorithm 4. 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

$$p(\boldsymbol{x}') = \sum_i w_i \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} \exp\left(-\frac{1}{2}(\boldsymbol{x}' - \boldsymbol{\mu}_i)^{\mathrm{T}} \Sigma^{-1} (\boldsymbol{x}' - \boldsymbol{\mu}_i)\right), \tag{4}$$

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 Liu and West [39]. The covariance of each perturbation is then given by

$$\boldsymbol{\Sigma} = (1-a^2)\mathrm{Cov}[\boldsymbol{x}].$$

Our resampling algorithm then involves drawing n new particles from the distribution given by (4) and setting the weight of each new particle to $1/n$.

There are a few details to address regarding the efficiency of the SMC algorithm. The first thing to note is that, since the choice of resampling algorithm is usually tailored to the problem at hand, it is hard to say something in general about the algorithmic complexity of it. A more pressing issue for us, however, is that quantum simulation is required in order to evaluate the likelihood function. This step will not generally be efficient because no known classical algorithm exists that can simulate generic quantum dynamics in time polynomial in the number of interacting subsystems.

Thus, since calls to the likelihood function are expensive, we wish to minimize the number of times it is called. To achieve this, we use an approximation that involves using only the highest weighted particles to compute the expectation values appearing in the utility function. Note that this will not reduce the accuracy of the estimation of parameters *given* experiments—rather, it reduces the accuracy with which we choose optimal experiments. This is the ideal place to make the approximation since the optimization routine makes the most calls to the likelihood function and no experiment is 'bad' in the sense that the expected risk (and hence the actual risk, on average) can increase. This idea is formalized in the algorithm by setting a parameter `approx_ratio` which is the percentage of particles to use for the approximation. We give pseudocode for this algorithm in algorithm 6.

We combine these prior algorithms to obtain algorithm 7, 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. In section 8, we compare the Newton conjugate-gradient (NCG) and nonlinear conjugate-gradient methods to a 'null' optimizer that only evaluates the utility function at the initial guesses. Also in section 8, we consider which choices of n, n_{guesses} and `approx_ratio` result in a useful estimation algorithm.

5. Region 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 $\mathrm{Vol}(\hat{X})$ is minimized. This is useful, for instance, if we consider the region estimate $\hat{X}$ as input to an optimal control theory (OCT) algorithm that describes the range of dynamics

Algorithm 5. Approximate utility functions using SMC.

```
Input: Particle weights $w_i$, $i \in \{1, \ldots, n\}$.
Input: Particle locations $\boldsymbol{x}_i$, $i \in \{1, \ldots, n\}$.
Input: Control description $C$.
Input: Positive semi-definite scaling matrix $\boldsymbol{Q}$.
Output: Utility $U(C)$.
  function UtilNV($\{w_i\}$, $\{\boldsymbol{x}_i\}$, $C$, $\boldsymbol{Q}$)    ▷ Calculates the negative variance utility function.
      for $D \in 1 \to n_{\text{outcomes}}$ do
          $\{w'_i\} \leftarrow$ Update($\{w_i\}$, $\{\boldsymbol{x}_i\}$, $D$, $C$)    ▷ Find the hypothetical weights, had we obtained the datum $D$.
          $\boldsymbol{\mu} \leftarrow$ Mean($\{w'_i\}$, $\{\boldsymbol{x}_i\}$)    ▷ Calculate the mean using the updated weights.
          $u_D \leftarrow -\sum_i w_i (\boldsymbol{x}_i - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{Q} (\boldsymbol{x}_i - \boldsymbol{\mu})$    ▷ Reduce the variance to a scalar by using the scaling matrix $\boldsymbol{Q}$.
          $u_D \leftarrow u_D \cdot \sum_i w_i \Pr(D|\boldsymbol{x}_i, C)$    ▷ Weight the partial utility $u_D$ by the marginalized likelihood $\Pr(D|C)$.
      end for
      return $\sum_D u_D$
  end function

  function UtilIG($\{w_i\}$, $\{\boldsymbol{x}_i\}$, $C$)    ▷ Calculates the information gain utility function.
      for $D \in 1 \to n_{\text{outcomes}}$ do
          for $i \in 1 \to n$ do
              $p_{D|\boldsymbol{x}_i} \leftarrow \Pr(D|\boldsymbol{x}_i, C)$
          end for
          $p_D \leftarrow \sum_i w_i p_{D|\boldsymbol{x}_i}$
      end for
      return $I \leftarrow H_D(p_D) - \sum_i w_i H_D(p_{D|\boldsymbol{x}_i})$    ▷ $H_D$ is the entropy over data $D$.
  end function
```

Algorithm 6. Reduced particle approximation for SMC utility functions.

```
Input: Particle weights $w_i$, $i \in \{1, \ldots, n\}$.
Input: Particle locations $\boldsymbol{x}_i$, $i \in \{1, \ldots, n\}$.
Input: 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($\{w_i\}$, $\{\boldsymbol{x}_i\}$, approx_ratio)
      $\tilde{n} \leftarrow \lfloor n \cdot \texttt{approx\_ratio} \rfloor$
      draw $\pi$ uniformly at random from $\mathrm{Sym}(n)$    ▷ $\mathrm{Sym}(n)$ is the symmetric group acting on $n$ elements.
      $\{\tilde{w}_i\} \leftarrow \{w_{\pi(i)}\}$    ▷ We permute the elements to avoid introducing patterns when sorting the particle weights.
      $\{\tilde{\boldsymbol{x}}_i\} \leftarrow \{\boldsymbol{x}_{\pi(i)}\}$
      $\{s_k\} \leftarrow$ Sort($\{\tilde{w}_i\}$)    ▷ Get a list of indices $s_i$ such that $\tilde{w}_{s_i} \geqslant \tilde{w}_{s_j}$ for all $i, j$.
      return $\{\tilde{w}_i\} \leftarrow \{\tilde{w}_{s_i} : i \in 1 \to \tilde{n}\}$, $\{\tilde{\boldsymbol{x}}_i\} \leftarrow \{\tilde{\boldsymbol{x}}_{s_i} : i \in 1 \to \tilde{n}\}$
  end function
```

Algorithm 7. Complete adaptive Bayesian experiment design algorithm, using SMC approximations.

```
Input: A number of particles $n$ to be used.
Input: A prior distribution $\pi$ over models.
Input: A number of experiments $N$ to perform.
Input: A resampling parameter $a \in [0, 1]$.
Input: A threshold resample_threshold $\in [0, 1]$ specifying how often to resample.
Input: An approximation ratio approx_ratio.
Input: An local optimization algorithm LocalOptimize.
Input: A particular choice of utility function Util.
Input: A heuristic GuessExperiment for choosing experiment controls, and a number $n_{\text{guesses}}$
    of potential experiments to consider in each iteration.
Output: An estimate $\hat{\boldsymbol{x}}$ of the true model $\boldsymbol{x}_0$.

  function EstimateAdaptive($n$, $\pi$, $N$, $a$, resample_threshold, approx_ratio, Optimize,
  Util, $n_{\text{guesses}}$, GuessExperiment)
      $w_i \leftarrow 1/n$                                  ▷ Start by initializing the SMC variables.
      draw each $\boldsymbol{x}_i$ independently from $\pi$

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

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

          $i_{\text{best}} \leftarrow \operatorname{argmax}_{i_{\text{guess}}} U_{i_{\text{guess}}}$  ▷ We pick the experiment whose post-optimization utility is
  highest.
          $\hat{C} \leftarrow \hat{C}_{i_{\text{best}}}$
          $D_{i_{\text{exp}}} \leftarrow$ the result of performing $\hat{C}$          ▷ The best experiment is then performed.
          $\{w_i\}, \{\boldsymbol{x}_i\} \leftarrow$ Update($\{w_i\}, \{\boldsymbol{x}_i\}, D, C$)          ▷ This update carries the posterior
  distribution forward.

          if $\sum_i w_i^2 < N \cdot$ resample_threshold then   ▷ Resample if the effective sample size
  $n_{\text{ess}} <$ resample_threshold.
              $\{w_i\}, \{\boldsymbol{x}_i\} \leftarrow$ Resample($\{w_i\}, \{\boldsymbol{x}_i\}, a$)
          end if
      end for

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

  end function
```

experienced by a quantum system. Since the OCT algorithm must find a control design that is robust for every point in that range, the cost of that optimization increases with the volume of $\hat{X}$.

Because we are interested in the probability $\Pr(\boldsymbol{x}_0 \in \hat{X})$ of the true model $\boldsymbol{x}_0$ lying within our region estimate $\hat{X}$ for a given run of the SMC algorithm, we say that $\hat{X}$ is a *credible* region [40, 41]. This is in contrast to *confidence* regions which are functions from data records D to regions $\hat{X}_{\mathrm{conf}}(D)$ such that the probability of obtaining a data record for which $\boldsymbol{x}_0 \in \hat{X}_{\mathrm{conf}}(D)$ is at least some threshold [42, 43]. That is, credible regions give probabilities about a single data record, while confidence regions give probabilities about all possible data. Broadly speaking, credible regions are Bayesian analogues to the frequentist concept of confidence regions. Unless otherwise noted, we concern ourselves here with credible regions as obtained from posterior distributions. The two kinds of region estimates are closely related, as has recently been explored in the context of quantum tomography [44, 45].

We make the problem of finding a credible region estimate amenable to analysis by SMC by turning the problem into that 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 $\hat{X}$, defined by

$$1_{\hat{X}}(\boldsymbol{x}) = \begin{cases} 1, & \boldsymbol{x} \in \hat{X} \\ 0, & \boldsymbol{x} \notin \hat{X} \end{cases}.$$

The expectation value $\mathbb{E}[1_{\hat{X}}]$ can then be computed using algorithm 2, giving that

$$\begin{aligned} \mathbb{E}[1_{\hat{X}}] &\approx \sum_i w_i 1_{\hat{X}}(\boldsymbol{x}_i) \\ &= \sum_{i, \boldsymbol{x}_i \in \hat{X}} w_i. \end{aligned}$$

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 $\tilde{m}(X_r) \geqslant r$ and may be computed using well-known classical algorithms [46, 47].

We improve on these results by supposing that, after collecting a reasonable amount of data, the posterior distribution is approximately normally distributed according to $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for some $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. This assumption holds when the Fisher information is non–singular, and we find in the case of our benchmarks that it approximately holds if $\Pr(\boldsymbol{x}|D)$ is sharply peaked. Under this assumption, it follows from the definition of the multivariate normal distribution that the inverse covariance matrix $\boldsymbol{\Sigma}^{-1}$ describes an ellipse such that approximately 0.682^d of the probability mass is contained inside the ellipse, where d is the number of unknown model parameters, $d = \dim \boldsymbol{x}$. In particular, the covariance matrix transforms a vector $\boldsymbol{z}$ of length d with each component drawn from the standard normal distribution $\mathcal{N}(0, 1)$ into a random variate of a multivariate normal distribution with the given covariance, and so inverting that transformation gives the z-score for each component.

More generally, under the assumption of a normally distributed posterior, the error ellipse of points $\boldsymbol{x}$ satisfying

$$(\boldsymbol{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \leqslant Z^2 \tag{5}$$

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 estimated covariance matrix according to algorithm 1 can be used as a region estimator.

The volume of the covariance ellipse region estimator can then be found by again treating $\boldsymbol{\Sigma}$ as a transformation of a d-dimensional coordinate system, so that

$$\mathrm{Vol}(\boldsymbol{\Sigma}^{-1}) = \frac{\pi^{d/2}}{\Gamma(\frac{d}{2} + 1)} \det(\boldsymbol{\Sigma}^{1/2}).$$

We test that the normal posterior assumption is approximately met by finding the approximate probability mass $\tilde{m}(\mathrm{Cov}(\hat{\boldsymbol{x}})^{-1}/Z^2)$ for a given Z, and comparing to the expected cdf. Moreover, we compare the optimal size of the covariance ellipse to the actual size by appealing to the BCRB, since $\mathbb{E}_\pi[\mathrm{Cov}(\hat{\boldsymbol{x}})] \geqslant \boldsymbol{J}(\pi; C)^{-1}$. This comparison will be explored further in section 8, and will be used as a performance metric for our algorithm.

6. Hyperparameter estimation

Now that we have discussed the general framework for using Bayesian inference to learn Hamiltonian parameters, we will proceed to discuss an important generalization of the prior work. The generalization that we consider addresses the fact that quantum systems seldom have consistent Hamiltonians from experiment to experiment, due to experimental errors. Hyperparameters allow us to generalize the Hamiltonian learning problem from one involving learning the Hamiltonian parameters to one that involves learning the parameters that describe the distribution of Hamiltonian parameters. In this way, the method of hyperparameter estimation is an alternative to approaches such as quantum smoothing [29, 30], which integrate over the *history* of a time-varying random parameter, rather than considering the distribution from which each realization is being drawn.

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 7 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. Specifically, in order to compute the likelihood function $\Pr(D|\boldsymbol{y})$ we need to compute the probabilities of data d emerging from an ensemble of randomly sampled experimental parameters taken from the distribution described by $\boldsymbol{y}$. A large number of samples, N_s, may be required in some cases since the sample error scales as $1/\sqrt{N_s}$. On the other hand, the approach is straight forward to implement because it does not require either increases to the number of particles or changes to the underlying algorithm.

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 7.3, 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,

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}}[\boldsymbol{x}] = \mathbb{E}_{\boldsymbol{y}}[\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y}}[\boldsymbol{x}]]. \tag{6}$$

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

$$\mathrm{Cov}_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{y}}[\mathrm{Cov}_{\boldsymbol{x}|\boldsymbol{y}}(\boldsymbol{x})] + \mathrm{Cov}_{\boldsymbol{y}}(\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y}}[\boldsymbol{x}]). \tag{7}$$

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

$$\mathrm{Var}_{\boldsymbol{x},\boldsymbol{y}}(\boldsymbol{x}) = \mathbb{E}_{\boldsymbol{y}}[\mathrm{Var}_{\boldsymbol{x}|\boldsymbol{y}}(\boldsymbol{x})] + \mathrm{Var}_{\boldsymbol{y}}(\mathbb{E}_{\boldsymbol{x}|\boldsymbol{y}}[\boldsymbol{x}]). \tag{8}$$

Using the region estimate given by (5) to estimate a hyperparameter region 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 8.

An important consequence of this derivation is that the same SMC algorithm can be used to estimate regions of model parameters, even when those model parameters vary with each experiment. In particular, as the covariance $\mathrm{Cov}_{\boldsymbol{y}}(\hat{\boldsymbol{x}})$ in our estimate $\hat{\boldsymbol{x}}$ of the mean model parameter vector decreases, our estimate of the model region approaches the 'true' model region given by $\mathrm{Cov}_{\boldsymbol{x}|\boldsymbol{y}}(\boldsymbol{x}|\boldsymbol{y})$.

7. Test cases

We assess the performance of algorithm 7 through a number of test cases that are designed to examine the performance of the algorithm in a number of different relevant settings. Here we describe the test cases, which we build up in complexity starting from the simple model studied in detail in [48–50]. The first case that we consider is learning a single parameter for an experiment in the presence of a known decoherence time. The second case generalizes this by allowing T_2 to be unknown. This problem is particularly important because existing experiments require substantial processing to learn both the unknown parameter and the decoherence time. Finally, we consider a two-hyperparameter model with a single-parameter Hamiltonian and perform region estimation of both the hyperparameters and the parameters that are distributed according to them.

7.1. Single parameter Hamiltonian

We first consider the example with one unknown and one control parameter. Suppose that a qubit evolves under an internal Hamiltonian

$$H(\omega) = \frac{\omega}{2}\sigma_z.$$

Here ω is an unknown parameter whose value we want to estimate. An experiment consists of preparing a single known input state $\psi_{\rm in} = |+\rangle$, the +1 eigenstate of σ_x, evolving under the Hamiltonian H for a controllable time t and performing a measurement in the σ_x basis. The kth measurement has two outcomes which we record as $d_k \in \{0, 1\}$. The design specification for each experiment is given by the time t_k that the state is allowed to evolved for under the unknown Hamiltonian in this case. Protocols that are provably optimal have been obtained by finding analytic expressions, and lower bounds, on the accuracy of generic protocols [50].

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:

$$\begin{aligned}\Pr(0|\omega; t) &= \mathrm{e}^{-\frac{t}{T_2}}\cos^2\left(\frac{\omega}{2}t\right) + \frac{1-\mathrm{e}^{-\frac{t}{T_2}}}{2},\\ \Pr(1|\omega; t) &= 1 - \Pr(0|\omega; t),\end{aligned} \tag{9}$$

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 [48–50]. There the model was able to be treated analytically. For the case with no noise ($T_2 = \infty$), given a normal prior with variance σ^2, the risk scales exponentially as $r \sim \sigma^2(1-\mathrm{e}^{-1})^N$, whereas for finite T_2, the scaling is exponentially suppressed when the measurement times reach $t = T_2$.

7.2. Two parameter model with single control

Here, we are going to consider the same model from the previous section,

$$\begin{aligned}\Pr(0|\omega, T_2; t) &= \mathrm{e}^{-\frac{t}{T_2}}\cos^2\left(\frac{\omega}{2}t\right) + \frac{1-\mathrm{e}^{-\frac{t}{T_2}}}{2},\\ \Pr(1|\omega, T_2; t) &= 1 - \Pr(0|\omega, T_2; t),\end{aligned} \tag{10}$$

but where now both ω and T_2 are unknown. The time t remains the only experimentally controllable parameter. For numerical convenience, we choose to parameterize this model as $\boldsymbol{x} = (\omega, T_2^{-1})$, so that each unknown parameter has the same dimensions.

Even for such a simple generalization as this, the methods discussed above 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. Although we cannot find an analytic expression for the BCRB in the same way we did for the single parameter problem, we use our SMC algorithm to efficiently numerically estimate it.

In general, there is no reason to expect that ω and T_2^{-1} will be expressed in such a way as to admit the same scale, and so in estimating using this model, we must set the semidefinite matrix $\boldsymbol{Q}$ for the loss function (2). This is discussed further in section 8.2.

7.3. One parameter model with hyperparameters

We also derive a two-parameter model similar to (10) by again considering a single qubit undergoing Larmor precession as in (9), 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 6, 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

$$\Pr(d|\mu, \sigma; t) = \int \Pr(d|\omega) \Pr(\omega|\mu, \sigma) \mathrm{d}\omega. \tag{11}$$

For the specific example of the Gaussian distribution,

$$\Pr(0|\mu, \sigma; t) = \frac{1}{\sigma\sqrt{2\pi}} \int \cos^2\left(\frac{\omega t}{2}\right) \mathrm{e}^{-\frac{(\omega-\mu)^2}{\sigma^2}} \mathrm{d}\omega \tag{12}$$

$$= \frac{1}{2}\left(1 + \mathrm{e}^{-2\sigma^2 t^2} \cos(2\mu t)\right). \tag{13}$$

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,

$$\Pr(0|\omega_0, \gamma; t) = \int \cos^2(\omega t/2) \frac{1}{\pi\gamma\left(\frac{(\omega-\omega_0)^2}{\gamma^2} + 1\right)} \mathrm{d}\omega \tag{14}$$

$$= \frac{1}{2}(1 + \mathrm{e}^{-t\gamma} \cos(t\omega_0)). \tag{15}$$

Note that if we identify $\gamma = T_2^{-1}$, then the Lorentz hyperparameter model is the identical to that of equation (10). This illustrates the relationship between decoherence processes and the lack of knowledge formalized by a hyperparameter model. In a similar fashion, (13) is also model of decoherence. Due to the t^2 dependence of the Gaussian-hyperparameter model, (13) represents a decoherence process that cannot be written in Lindblad form [51] because it cannot be drawn from a quantum dynamical semigroup.

The approach of introducing hyperparameters allows us to estimate unknown distributions using the same SMC algorithm 7 that we use for estimating other model parameters, by making assumptions about the form of an unknown distribution. Using techniques developed in section 6, we can also extend region estimation to hyperparameter models to obtain regions on the intermediate model (in this case, ω) using regions on $\boldsymbol{y}$. We discuss the results of these application in detail in section 8.2.

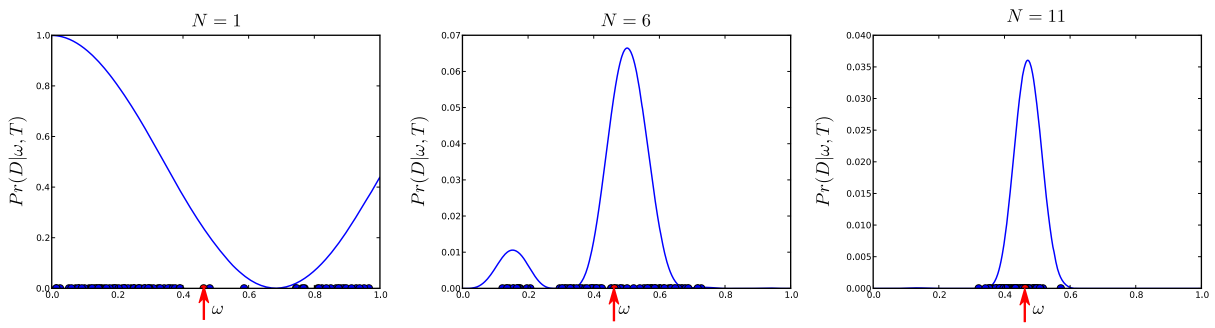

Figure 2. Left to right: the likelihood function for $N = 1, 6$ and 11 simulated measurements at random times in in $(0, 5\pi)$. The model is that given in equation (9) with $T_2 = 100\pi$. The red dots (and red arrows) are the randomly chosen true parameter ω. The blue dots are the $n = 100$ SMC 'particles'.

8. Results and discussion

We will now turn our attention toward assessing the performance of algorithm 7 in practical examples[12]. Our results show that our adaptive Bayesian algorithm is able to learn model parameters using a very small number of experiments, including adverse situations where hyperparameters are needed to describe the fluctuations of model parameters between experiments or where the decoherence time of the system is unknown. This performance is especially noteworthy in the case of an unknown decoherence time T_2. Though methods of learning unknown decoherence processes have been developed and are well-understood [52–54], these methods require multiple iterations of quantum process tomography implemented using either ensemble measurement or a large amount of data obtained using strong measurement. In the case discussed here, we can adaptively use prior information to obtain accurate estimates of T_2 using significantly less measurements than methods currently employed in systems with strong measurement [32].

Figures 2 and 12 build intuition for how algorithm 7 actually learns parameters by describing a trial run for each of the models in sections 7.1 and 7.2. These illustrate the movement of the particles, through resampling, to regions of high likelihood. In particular, we note that figure 2 demonstrates that only 11 experiments are needed in order to achieve a tight approximately Gaussian posterior distribution over the model parameter ω for the known T_2 model described in section 7.1. Figure 12 shows that performance of the algorithm for the unknown T_2 model described in section 7.2. We see in that case that only 200 experiments are needed to find a distribution that is centered around the true values of ω and T_2.

8.1. Results for known T_2 model

Our first set of numerical experiments examines the performance of our algorithm for the case where the decoherence time is known with perfect precision. Specifically, we take $T_2 = 100\pi$, $\omega \sim \mathcal{N}(0.5, 0.01)$ and choose the experimental times to be spaced uniformly such that the kth experiment occurs at time $t_k = 2k\pi/3$ (chosen arbitrarily) and examine the mean-square error in ω averaged over a minimum of 1625 trials with randomly chosen ω. This data is presented

[12] For the numerical experiments, algorithm 7 was implemented using version 0.9.0 of the SciPy package [57] with Python versions 2.7.1, 2.7.2 and 2.7.3.

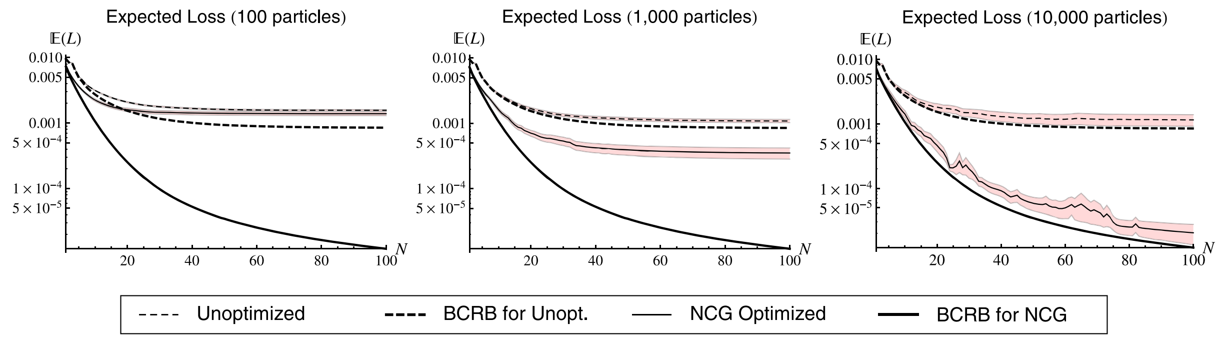

Figure 3. Left to right: the performance, as a function of the number of measurement N, of the SMC algorithm for $n = 100$, 1000 and 10 000 particles. The model is that of equation (9) with $T_2 = 100\pi$ (see also figure 2). The dashed lines indicate data taken without local optimization, while the solid lines indicate trials in which initial guesses were optimized using the NCG method. For each data set, the corresponding thick line indicates the BCRB. Errors in estimating the performance are indicated by red shaded regions around each curve.

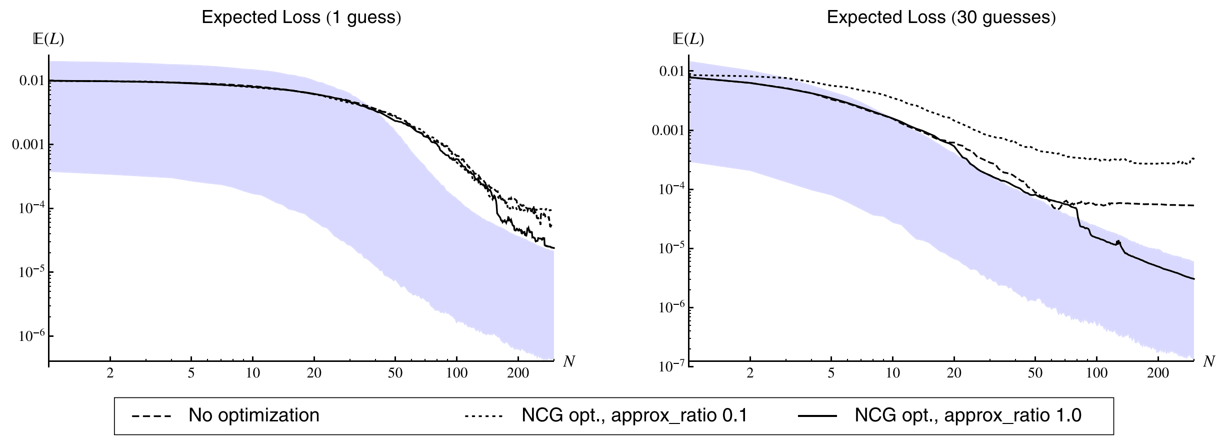

Figure 4. This figure compares the mean-square error as a function of the number of experiments used for the known T_2 model with $T_2 = 100$, 5000 particles and an approximation ratio of 1 with guessed experimental times chosen randomly from an exponential distribution with mean T_2. On the left, data is shown for 1 guess, while on the right, we show data for 30 guesses. The shaded region in each plot indicates a 68% confidence interval for data collected using NCG optimization with `approx_ratio` $= 1.0$.

in figure 3. The figure shows that the mean-square error for ω decreases as we vary the number of particles from 100 to 10 000, and in particular gives a relative MSE that is less than 1% for $N \geqslant 100$. We also see that the data remain close to the BCRB (which is a lower bound on the MSE) for either $n = 1000$ or 10 000 particles if no optimization is used, whereas $n = 10\,000$ is needed to approach the BCRB in the case where NCG is used. This not only shows that several thousand particles should be sufficient for our purposes, but also that the MSE yielded by our algorithm scales near–optimally if a sufficiently large number of particles are used in the SMC approximation.

Figure 4 provides a more in depth analysis of the error scaling for the known T_2 model. The previous data set contained only one experimental guess per experiment, whereas in general we could consider making many guesses for the optimal experiment and choosing the one with the highest utility $U(t) = -\mathbb{E}_D[L]$. We assess the performance of our algorithm as a function of the number of guesses by introducing a new guess heuristic. Instead of choosing uniformly spaced guesses, we choose each guessed time randomly from an exponential distribution with mean T_2. This guess heuristic also has the advantage of using very little intuition about the structure of the Hamiltonian that we are attempting to parameterize, which means that we expect the performance of this heuristic to be a better estimate of the worst-case performance of our algorithm.

Unsurprisingly, we find that the MSE is reduced if we choose the best of 30 possible experiments rather than a single randomly chosen experiment. We also see that local optimization tends to improve the quality of the approximation if we pick the approximation ratio to be 1. Figure 4 also shows that smaller values of the approximation ratio can cause the mean-square error to saturate at relatively large values. In fact, taking $\texttt{approx\_ratio} = 0.1$ was sufficient to cause the data taken for 30 guesses and NCG optimization to have a *larger* mean-square error in ω than the case with no optimization and 1 random guess. For this reason, we take the $\texttt{approx\_ratio} = 1$ for most of the examples in this section. In section 8.4, we shall see an example where $\texttt{approx\_ratio} < 1$ provides more benefit.

An individual trial may have a mean-square error that differs significantly from the expected loss. The shaded regions in figure 4 give 68% confidence intervals for the actual loss for the case with NCG and $\texttt{approx\_ratio} = 1$ (the confidence intervals for the other data sets were similar), and find the surprising result that the mean-square error is frequently outside the confidence interval in the cases where NCG optimization is not used. This suggests that the distribution of the utility of experiments can wildly vary and that the distribution is skewed because a significant fraction of the guesses provide virtually no information. NCG minimizes the chances of choosing an uninformative experiment and hence it forces the guesses toward the more informative experiments. We should also note that there is room for improvement here, since the MSE is closer to the upper limit of the confidence interval. Sophisticated optimization procedures may therefore be of use when searching for informative experiments given an uninformed guess heuristic (such as our random guess heuristic).

These results show that poor guess heuristics can be mitigated using our algorithm by optimizing over more guesses and using local optimization of the experiments. We also note that the median-square error performance of our algorithm tends to be much better than the mean-square error because a small fraction of the trials randomly choose very bad guesses. We see that NCG optimization can be used to cause the mean-square error to approach the median-square error. We therefore find that estimates of the error (such as the posterior variance or region estimates of ω) are needed in order to guarantee that a particular trial is not pathological.

8.2. 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 5. A simplifying assumption is made in our

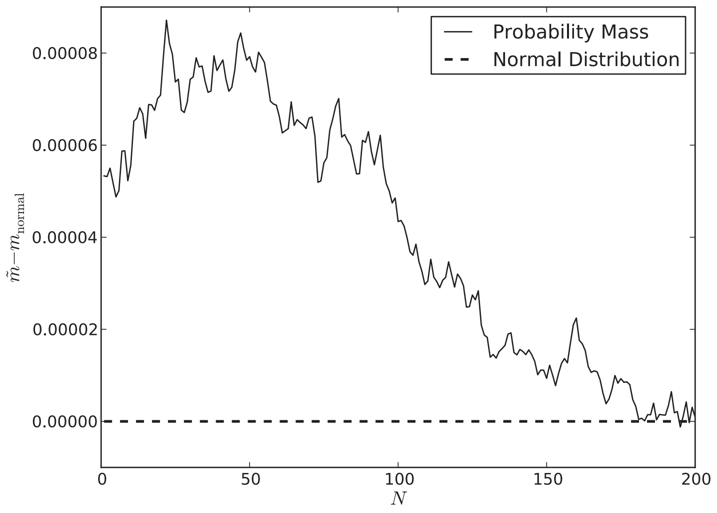

Figure 5. SMC approximated covariance probability mass $\tilde{m}(\mathrm{Cov}(\hat{\boldsymbol{x}})^{-1}/Z^2)$ for the known-T_2 single-parameter model as compared to the probability mass $m_{\mathrm{normal}} \approx 0.9973$ expected for the normal distribution and as a function of the number N of experiments performed, averaged over 20 119 trials using a guess heuristic that chooses the kth guess to occur at time $(9/8)^k$, using 30 guesses, 1000 particles and NCG optimization. The dashed line shows the probability mass for the corresponding normal distribution.

analysis: we assume that the posterior distribution is approximately Gaussian. Although difficult to justify theoretically, we have yet to find an example for the models considered here where the posterior does not appear Gaussian after a sufficiently large number of experiments. 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.

Figure 5 illustrates that the approximate probability mass $\tilde{m}$ approaches the probability mass we would expect for a normal distribution for the known-T_2 model (introduced in section 7.1) in 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.

8.3. Results for unknown T_2 model

We now turn our attention to the comparably challenging task of learning Hamiltonian parameters without a precise estimate of T_2. These calculations were performed using the true

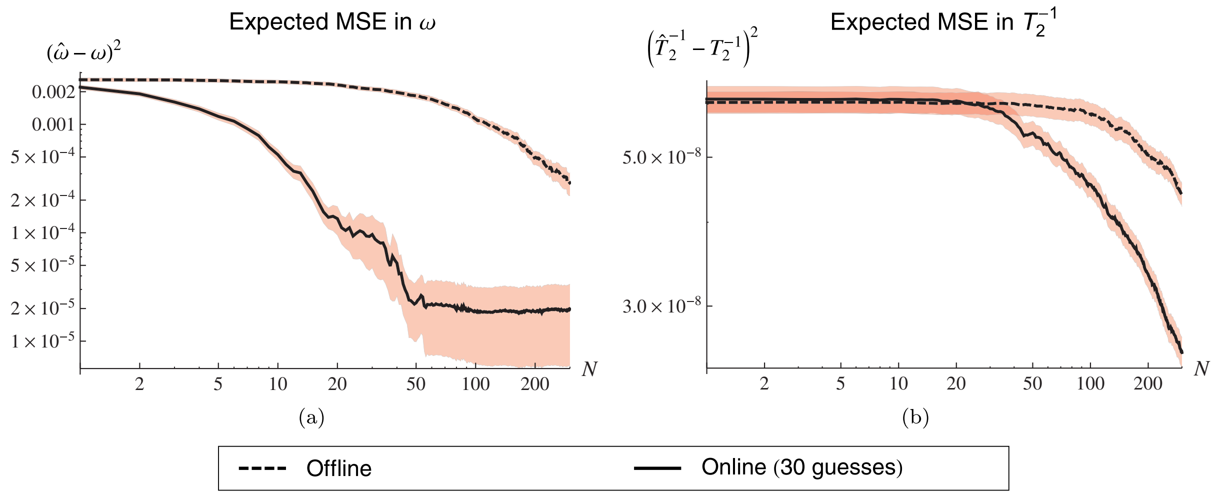

Figure 6. Benchmarking of the 'unknown-T_2' model (section 7.2) using $n = 5000$ particles and random initial guesses without local optimization. Data indicated dashed lines correspond to trials where a single initial guess was used for each experiment, while data indicated by solid lines were collected using 30 guesses per experiment. The single-guess data is averaged over 1380 trials while the 30-guess data is averaged over 1109 trials. Errors in estimating performance are indicated by red shaded regions about each curve.

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 1000, corresponding to the mean value of T_2 according to the initial prior.

We examine the variation of the MSE with the number of guesses used in figure 6. The figure shows that, in the absence of local optimization of experiment times, the MSE for both 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 7 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 6 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 8). 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 one 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, again, 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.

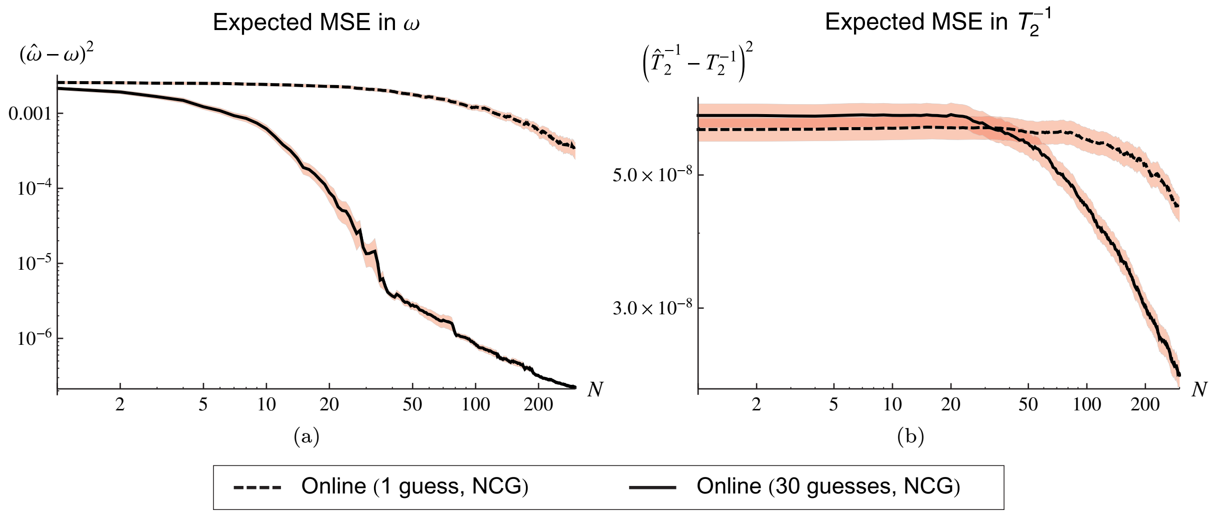

Figure 7. Benchmarking of the 'unknown-T_2' model (section 7.2) using $n=5000$ particles and random initial guesses with local optimization by the NCG method. Data indicated dashed lines correspond to trials where a single initial guess was used for each experiment, while data indicated by solid lines were collected using 30 guesses per experiment. The single-guess data is averaged over 1023 trials while the 30-guess data is averaged over 930 trials. Errors in estimating performance are indicated by red shaded regions about each curve.

Similarly to the case of known T_2, it is useful to benchmark the performance of our algorithm against the BCRB, which gives a lower bound on the MSE. Figure 9 provides 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.

8.4. 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 6. 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 of equation (12) as discussed in section 7.3, then comparing the model parameter region volume and probability mass for the region estimated from equation (7) to the volume and probability mass of the corresponding 'true' model parameter region. We benchmark this model by choosing 'true' hyperparameters μ and σ^2 for ω according to the normal distribution

$$\mu,\sigma^2\sim\mathcal{N}\left[(\mu_\mu,\mu_{\sigma^2}),\mathrm{diag}(\sigma_\mu^2,\sigma_{\sigma^2}^2)\right]. \tag{16}$$

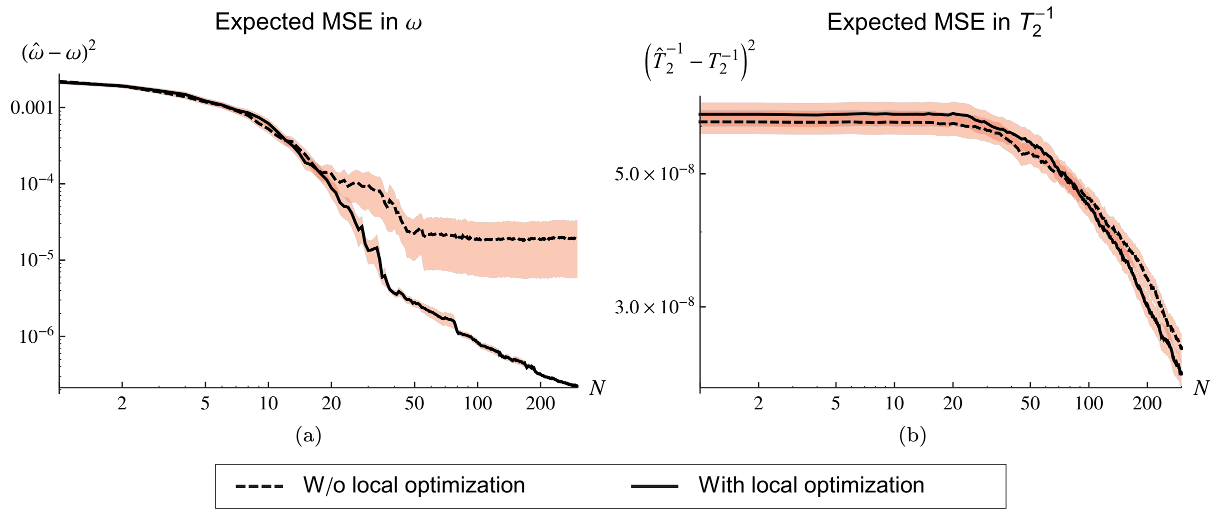

Figure 8. Benchmarking of the 'unknown-T_2' model (section 7.2) using $n = 5000$ particles and 30 random initial guesses. Data indicated dashed lines correspond to trials where a each initial guess was used without local optimization, while data indicated by solid lines were collected using NCG optimization for each guess. The unoptimized data is averaged over 1109 trials while the optimized data is averaged over 930 trials. Errors in estimating performance are indicated by red shaded regions about each curve.

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.

Figure 10(a) provides estimates of the probability mass contained within our estimated region for the Hamiltonian, which uses a Z-score of 3. Since we assume a Gaussian posterior, we anticipate that 99.7% of the probability mass should 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. It is also interesting that the data with `approx_ratio` $= 0.1$ yielded the best region estimation for the probability mass (unlike the previous examples). This is likely because the low approximation ratio de-emphasizes the tails of the posterior and non-Gaussian behavior typically is manifested in the tails. This shows that there are, perhaps surprisingly, examples where taking `approx_ratio` < 1 is useful.

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 (8), we calculate $\mathrm{Var}(\hat{\omega})$ as

$$\mathrm{Var}(\hat{\omega}) = \mathrm{Var}(\hat{\mu}) + \mathbb{E}[\hat{\sigma}^2].$$

Figure 10(b) compares $\mathrm{Var}(\omega)$ to the variance parameter σ^2. As the number of experiments grows, our region estimator for ω slightly overestimates the 'true' variance of ω (on average). We see from figure 10(b) that the estimate of the variance of the unknown frequency that is

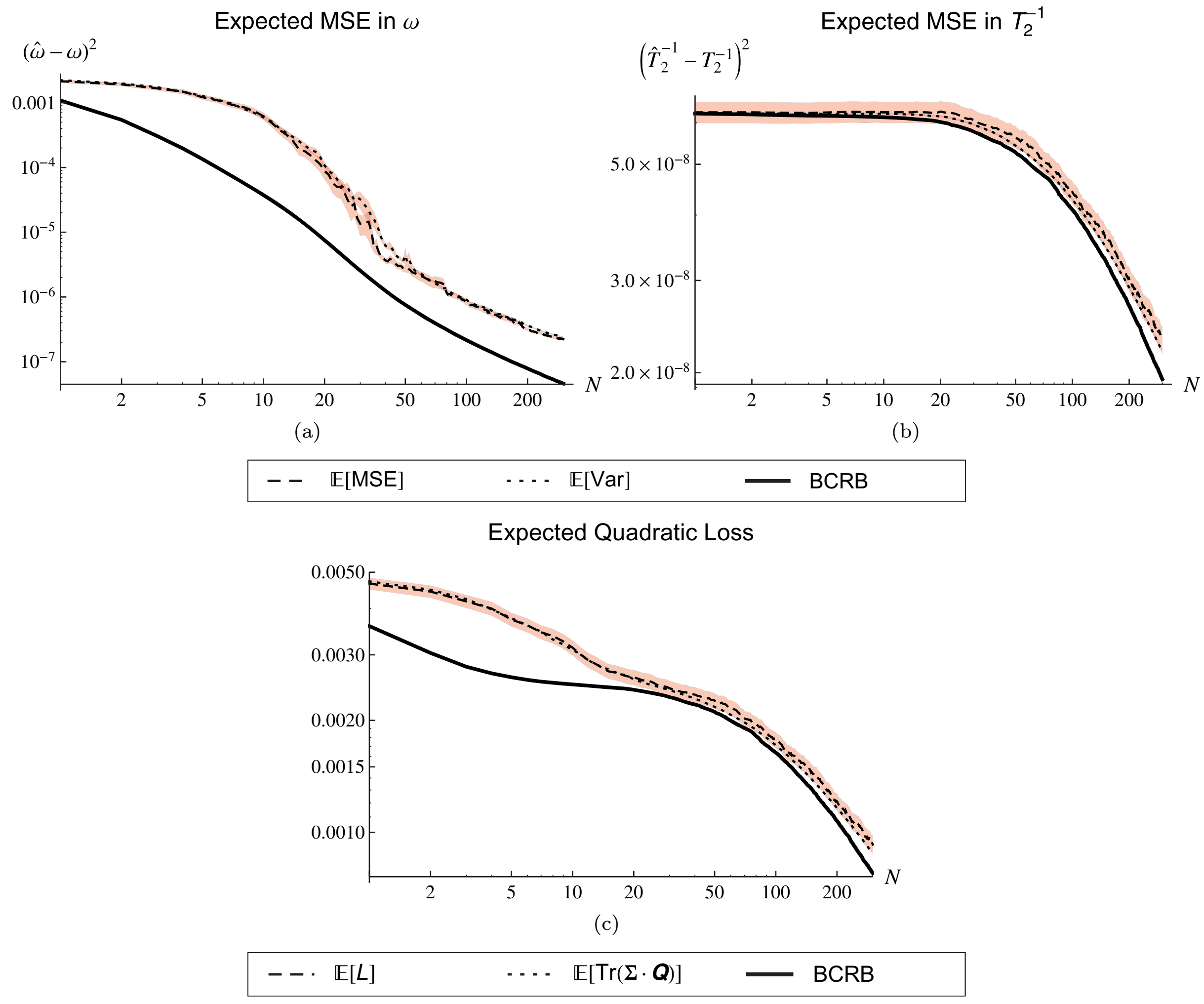

Figure 9. The actual and estimated performance, as a function of the number of measurements N, of the SMC algorithm for $n = 5000$ particles. The model is that of equation (10) with unknown T_2 (which is estimated as $\Gamma = 1/T_2$ for numerical precision considerations). The dotted curve is the posterior variance of the particles; dashed is the actual mean squared error and solid is numerically calculated Bayesian Cramer–Rao lower bound. In subfigures (a) and (b), the MSE and variances are those of the individual parameters ω and T_2^{-1}, respectively, while subfigure (c) 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.

inferred from our region estimate of the hyperparameters systematically over-estimates the variance of ω_{true} 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.

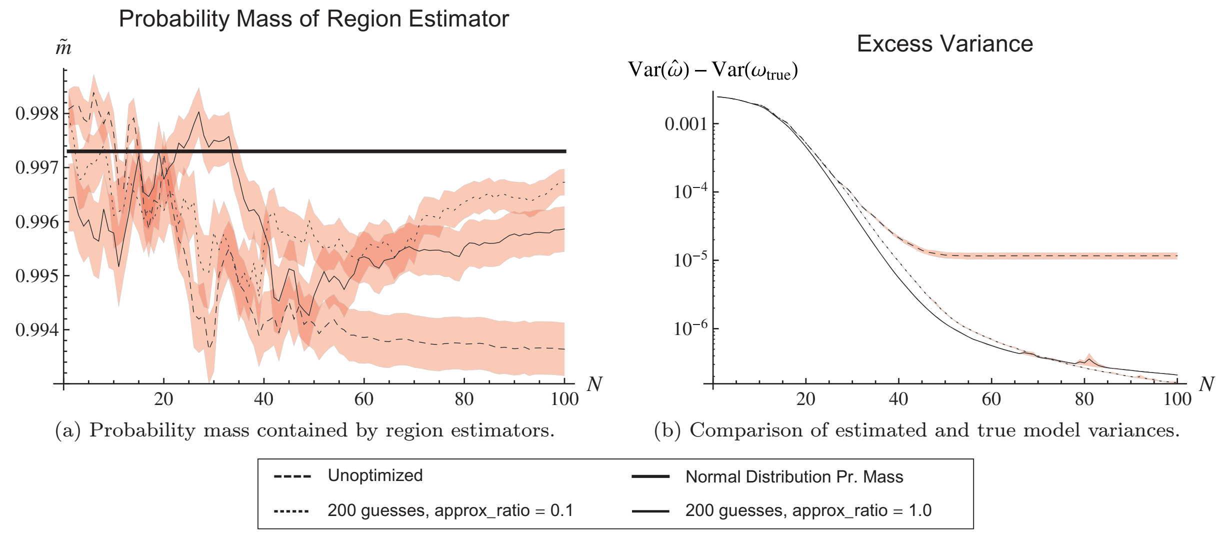

Figure 10. Benchmarking region estimators for Gaussian hyperparameter model (section 7.3) using $n = 2000$ particles, $\omega \sim \mathcal{N}(\mu, \sigma^2)$ where $\mu \sim \mathcal{N}(0.5, 0.001^2)$ and $\sigma^2 \sim \mathcal{N}(0.0025, 0.0025^2)$. (a) Probability mass contained by region estimators. (b) Comparison of estimated and true model variances.

8.5. 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 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 figures 4 and 11 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 11. The first feature that

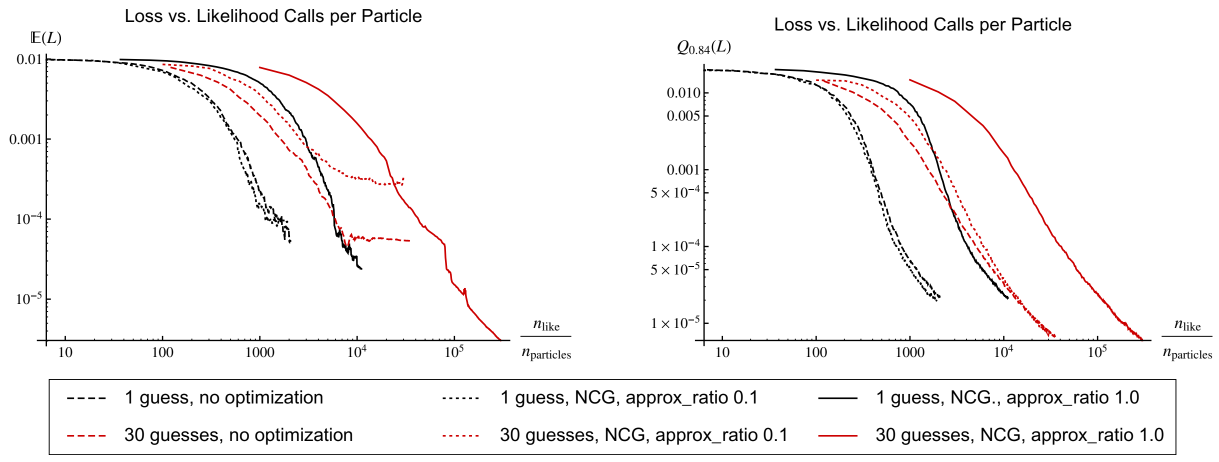

Figure 11. This figure compares the mean-square error as a function of the computational time for the known T_2 model with $T_2 = 100$, 5000 particles, `approx_ratio` $= 1$ and guessed experimental times chosen randomly from an exponential distribution with mean T_2. The expected loss incurred by each optimization strategy is shown is shown in the left figure and the figure on the right shows the 84th percentile $Q_{0.84}$ of the loss, such that no more than 16% of trials incur loss greater than the shown percentile.

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` $= 1$. 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 84th percentile of the loss in figure 4, 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.

There are many strategies that can be employed to reduce the computational time required by our algorithm. Firstly, since the calculation of each guess is independent of the other guesses, this task can be trivially parallelized on a cluster that uses very little communication between the nodes. Additionally, because the simulations do not need to be high precision for the method to be successful, a single-precision implementation of the simulation step could also be used to improve the performance of the simulation in circumstances where the quantum simulation used to evaluate the likelihood function is computationally expensive. Such simulations have

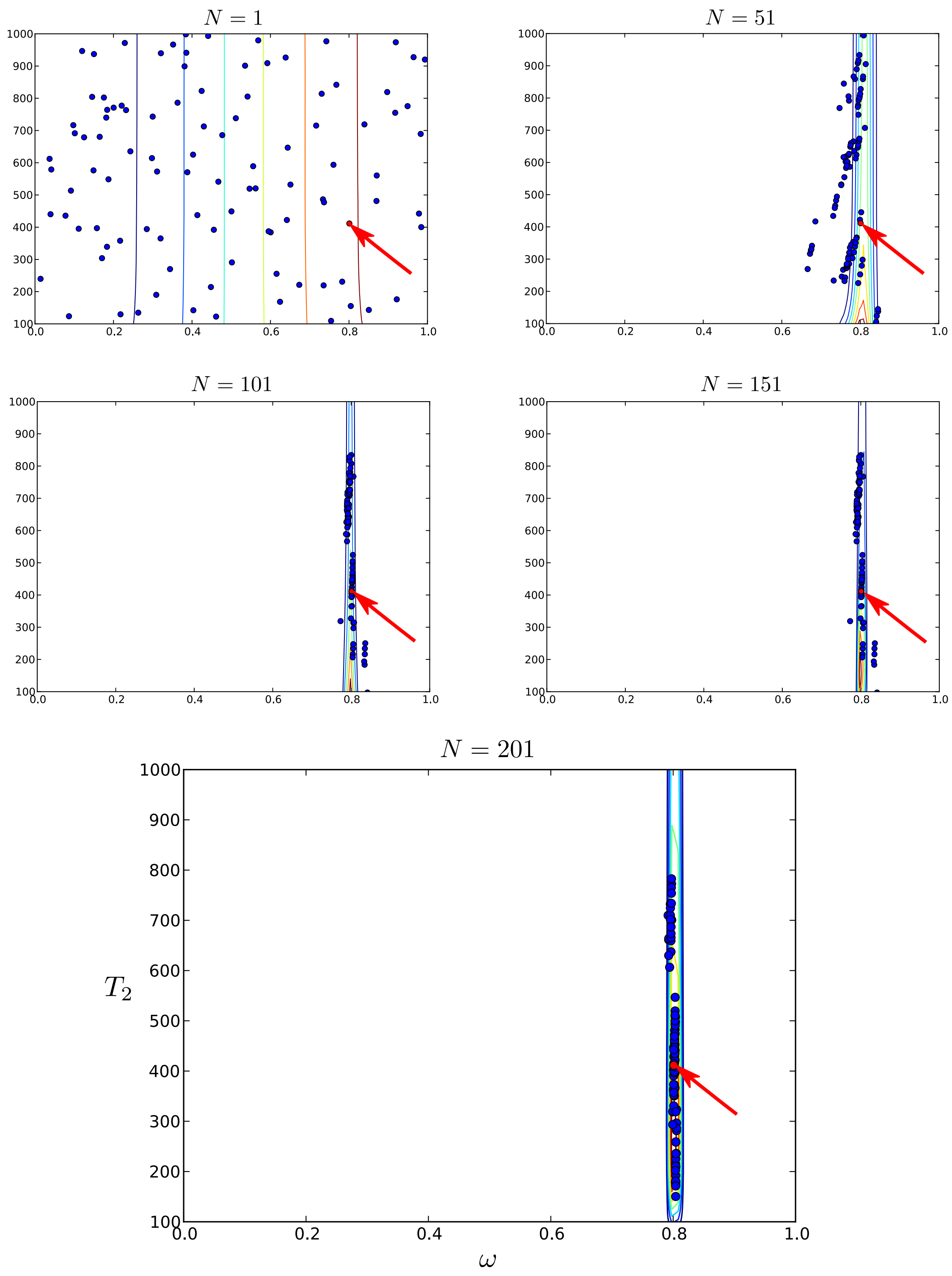

Figure 12. The likelihood function for $N = 1, 51, 101, 151$ and 201 simulated measurements at random times in in $(0, 20\pi)$. The model is that given in equation (10). The red dot (and red arrow) is the randomly chosen true parameter $\boldsymbol{x} = (\omega, T_2)$. The blue dots are the $n = 100$ SMC 'particles'.

been demonstrated using graphical processing units [55] and field-programmable gate arrays (FPGAs) [56]; the latter is of particular interest due to the use of FPGA devices in the control of quantum information processing systems.

9. 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. Firstly, it can be used to estimate the optimal parameterization of the dynamics of an arbitrary quantum system within a space of model Hamiltonians. Secondly, it can be used to provide a region estimate 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. Thirdly, our analysis of the algorithm shows a clear trade off between the experimental time and the computational time needed to parameterize the Hamiltonian.

We illustrated these advantages by benchmarking our algorithm's performance for a number of computationally tractable Hamiltonian models that involve a qubit precessing with an unknown frequency in the presence of decoherence (finite T_2 time). Our results showed that the scaling of the mean-square error of an unknown frequency with the number of experiments irrespective of whether T_2 is known approaches the BCRB, which is known to be optimal, given a set of experimental designs. In contrast, other methods for learning an unknown frequency require a well characterized T_2 time as a prerequisite. Our work, on the other hand, shows that we can learn the unknown frequency and the unknown T_2 simultaneously, which has obvious advantages if data collection is slow.

Perhaps most importantly, our algorithm also provides region estimates of the unknown parameters of the Hamiltonian. This is to say that at the end of the experiments we not only have an estimate of not only the values of the unknown parameters but also our uncertainty in those values. We showed that the method is capable of providing region estimates that contain the actual unknown parameters with probability approximately 99.7%. Our algorithm also was used to provide construct similar confidence intervals for cases where the unknown Hamiltonian parameters were not constant between experiments but instead fluctuated according to an unknown distribution. These tests showed that our algorithm is robust and also is valuable even if the cost of performing experiments is minimal.

Finally, we compared the costs in terms of experimental and computational time of our algorithm. We found that the heuristics that reduced the experimental time most significantly often required more computational time to reach a desired level of accuracy. Conversely, we found that many of the computationally inexpensive heuristics failed to reduce the mean-square errors after a finite number of experiments. This suggests that the relative merits of different heuristics change as the relative costs of computational time and experimental time and the precision with which the unknown parameters should be estimated vary.

An obvious 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.

Acknowledgments

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] Paris M and Rehacek J (ed) 2004 *Quantum State Estimation* (*Lecture Notes in Physics* vol 649) (Berlin: Springer)
[2] Lanyon B P *et al* 2011 *Science* **334** 57
[3] Gerritsma R, Lanyon B P, Kirchmair G, Zähringer F, Hempel C, Casanova J, Garcia-Ripoll J J, Solano E, Blatt R and Roos C F 2011 *Phys. Rev. Lett.* **106** 060503
[4] Kim K, Chang M-S, Korenblit S, Islam R, Edwards E E, Freericks J K, Lin G-D, Duan L-M and Monroe C 2010 *Nature* **465** 590
[5] Bendersky A, Pastawski F and Paz J P 2008 *Phys. Rev. Lett.* **100** 190403
[6] Bendersky A, Pastawski F and Paz J P 2009 *Phys. Rev.* A **80** 032116
[7] Mohseni M and Rezakhani A T 2009 *Phys. Rev.* A **80** 010101
[8] Branderhorst M P A, Nunn J, Walmsley I A and Kosut R L 2009 *New J. Phys.* **11** 115010
[9] Flammia S T and Liu Y K 2011 *Phys. Rev. Lett.* **106** 230501
[10] da Silva M P, Cardinal O L and Poulin D 2011 *Phys. Rev. Lett.* **107** 210404
[11] Oi D and Schirmer S 2012 arXiv:1202.5779
[12] Doucet A and Johansen A M 2009 *A Tutorial on Particle Filtering and Smoothing: Fifteen Years Later* (Oxford: Oxford University Press)
[13] Loredo T J 2004 *AIP Conf. Proc.* **707** 330
[14] Kuck H, de Freitas N and Doucet A 2006 *Nonlinear Statistical Signal Processing Workshop* (Piscataway, NJ: IEEE) pp 99–102
[15] Scarpa B and Dunson D B 2007 *Stat. Med.* **26** 1920
[16] Cavagnaro D R, Pitt M A and Myung J I 2009 *Advances in Neural Information Processing Systems* vol 22 pp 234–42
[17] Kantas N, Lecchini-Visintini A and Maciejowski J M 2010 *Int. J. Adapt. Control Signal Process.* **24** 882
[18] Huan X and Marzouk Y M 2011 arXiv:1108.4146
[19] Huszár F and Houlsby N M T 2012 *Phys. Rev.* A **85** 052120
[20] Bagan E, Ballester M A, Gill R D, Mϋnoz Tapia R and Isart O R 2006 *Phys. Rev. Lett.* **97** 130501
[21] Servedio R A and Gortler S J 2004 *SIAM J. Comput.* **33** 1067
[22] Aïmeur E, Brassard G and Gambs S 2006 *Lecture Notes in Computer Science* vol 4013 (Berlin: Springer) chapter 37, pp 431–42
[23] Aaronson S 2007 *Proc. R. Soc.* A **463** 3089–114
[24] Hentschel A and Sanders B C 2010 *Phys. Rev. Lett.* **104** 063603
[25] Pudenz K L and Lidar D A 2011 arXiv:1109.0325
[26] Hentschel A and Sanders B C 2011 *Phys. Rev. Lett.* **107** 233601
[27] Sergeevich A and Bartlett S D 2012 arXiv:1206.3830

[28] Caves C M 1986 *Phys. Rev.* D **33** 1643
[29] Tsang M 2009 *Phys. Rev. Lett.* **102** 250403
[30] Wheatley T A, Berry D W, Yonezawa H, Nakane D, Arao H, Pope D T, Ralph T C, Wiseman H M, Furusawa A and Huntington E H 2010 *Phys. Rev. Lett.* **104** 093601
[31] Lindley D V 1956 *Ann. Math. Stat.* **27** 986
[32] Said R S, Berry D W and Twamley J 2011 *Phys. Rev.* B **83** 125410
[33] Lehmann E L and Casella G 1998 *Theory of Point Estimation* 2nd edn (Berlin: Springer)
[34] Berger J O 1985 *Statistical Decision Theory and Bayesian Analysis* 2nd edn (Berlin: Springer)
[35] Blume-Kohout R and Hayden P 2006 arXiv:quant-ph/0603116
[36] Gill R D and Levit B Y 1995 *Bernoulli* **1** 59
[37] Tichavsky P, Muravchik C H and Nehorai A 1998 *IEEE Trans. Signal Process.* **46** 1386
[38] Gordon N J, Salmond D J and Smith A F M 1993 Radar and signal processing *IEE Proc.* F **140** 107–13
[39] Liu J and West M 2000 *Combined Parameter and State Estimation in Simulation-Based Filtering* (Berlin: Springer)
[40] Edwards W, Lindman H and Savage L J 1963 *Psychol. Rev.* **70** 193
[41] Bernardo J 2005 *TEST* **14** 317
[42] Wackerly D, Mendenhall W and Scheaffer R L 2001 *Mathematical Statistics with Applications Mathematical Statistics* 6th edn (Pacific Grove, CA: Duxbury)
[43] Beale E M L 1960 *J. R. Stat. Soc. B* **22** 41 (http://www.jstor.org/stable/2983877)
[44] Blume-Kohout R 2012 Robust error bars for quantum tomography arXiv:1202.5270
[45] Christandl M and Renner R 2011 arXiv:1108.5329
[46] Todd M J and Yldrm E A 2007 *Discrete Appl. Math.* **155** 1731
[47] Barber C B, Dobkin D P and Huhdanpaa H 1996 *ACM Trans. Math. Softw.* **22** 469–83
[48] Sergeevich A, Chandran A, Combes J, Bartlett S and Wiseman H 2011 *Phys. Rev.* A **84** 052315
[49] Ferrie C, Granade C E and Cory D G 2012 *AIP Conf. Proc.* **1443** 165
[50] Ferrie C, Granade C and Cory D 2012 *Quantum Inform. Process.* (doi:10.1007/s11128-012-0407-6)
[51] Lindblad G 1976 *Commun. Math. Phys.* (1965–1997) **48** 119
[52] Boulant N, Havel T F, Pravia M A and Cory D G 2003 *Phys. Rev.* A **67** 042322
[53] Weinstein Y S, Havel T F, Emerson J, Boulant N, Saraceno M, Lloyd S and Cory D G 2004 *J. Chem. Phys.* **121** 6117
[54] Boulant N, Emerson J, Havel T, Cory D and Furuta S 2004 *J. Chem. Phys.* **121** 2955
[55] Gutierrez E, Romero S, Trenas M and Zapata E 2008 *Computational Science* (*Lecture Notes in Computer Science* vol 5101) ed M Bubak, G van Albada, J Dongarra and P Sloot (Berlin: Springer) pp 700–9
[56] Khalid A, Zilic Z and Radecka K 2004 *ICCD 2004: Proc. IEEE Int. Conf. on Computer Design: VLSI in Computers and Processors* pp 310–15
[57] Jones E *et al* 2001 *SciPy: Open Source Scientific Tools for Python* (www.scipy.org/)