Simplified Algorithms for Adaptive Experiment Design in Parameter Estimation

Robert D. McMichael[*] and Sean M. Blakley[†]
National Institute of Standards and Technology, Gaithersburg, Maryland 20899, USA

(Received 26 April 2022; revised 13 July 2022; accepted 12 September 2022; published 1 November 2022)

Measurements to estimate parameters of a model are commonplace in the physical sciences, where the traditional approach to automation is to use a sequence of preselected settings followed by least-squares fitting of a model function to the data. This measure-then-fit approach is simple and effective and entirely appropriate for many applications but when measurement resources are limited, efficiency becomes more important. To increase efficiency, Bayesian experiment design allows measurement settings to be chosen adaptively based on accumulated data and utility, the predicted improvement in results as a function of settings. However, the calculation of utility has been judged too impractical for most applications. In this paper, we introduce computational methods and simplified algorithms that accelerate utility calculations by over an order of magnitude, with only slight degradation in measurement efficiency. The methods eliminate utility calculation as a barrier to practical application of efficient adaptive measurement.

DOI: 10.1103/PhysRevApplied.18.054001

I. INTRODUCTION

Sequential adaptive experiment designs frequently produce efficient measurement because they use accumulated data to inform decisions on measurement settings. Whether by choosing optimal designs or simply by avoiding wasteful efforts, adaptive design generally requires fewer measurements to achieve a given level of precision. However, adaptive design adds two statistical tasks that add computational cost to the measurement cycle. Each cycle, or *epoch*, includes a design task to choose settings and an inference task to analyze data in addition to measurement (see Fig. 1). A key question is whether the benefits of adaptive experiment design will outweigh the added computational cost for a particular measurement. Qualitatively, adaptive design is most likely to benefit expensive, slow, and noisy measurements. As computations become faster, adaptive design becomes advantageous for a larger class of measurements.

Bayesian optimal experiment design offers methods for the setting decision task, with roots in information theory and decision theory [1], and it is well suited for nonlinear model functions. Chaloner and Verdinelli [2] have published a review of early work that has become a touchstone in the field and, in recent decades, the availability of computational power has sparked a resurgence of interest in Bayesian optimal design. In Ref. [3], Overstall *et al.* provide a brief review of design approaches, while computational methods have been reviewed by Ryan *et al.* [4] More recently, publication of software has provided direct access to Bayesian design methods, potentially reducing the effort required for implementation [5–9].

The Bayesian optimal design method centers around locating maxima of the *utility*, $U(d)$, which expresses the goals of the measurement as a function of candidate setting designs d. The goal may be to minimize uncertainty in the parameters but a more conventional information-theoretic approach strives to decrease the information entropy of the parameter distribution.

Several authors have identified computation of the utility as a particularly difficult part of Bayesian experiment design, even identifying it as prohibitive [3,4,10,11]. The utility calculation generally involves statistics calculated over the combined spaces of parameter distributions and of design options. Importantly, for a sequential design to be preferable to a static design, the cost of implementing and running the design processes must not exceed the value of saved resources. For practitioners, utility calculations must be fast.

Several algorithmic approaches have been pursued to decrease the design computation time. Forgoing utility functions in favor of heuristic algorithms can be very effective at delivering good design decisions quickly but these often require careful design for single-purpose measurements. In the case of Ramsey measurements (relevant to magnetic resonance and interferometry), adaptive heuristics perform well in noisy measurements [12,13], while nonadaptive designs, such as the quantum phase estimation algorithm in particular [14–25], offer extremely efficient protocols for low-noise measurements. Amortization uses neural nets to shift the computational burden of experiment design to a time before measurement, when the neural net is trained. Then, during measurements, the

[*]rmcmichael@nist.gov
[†]sean.blakley@nist.gov

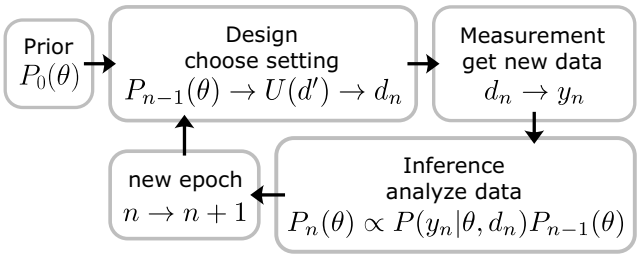

FIG. 1. The adaptive design and measurement cycle. The design task makes setting decisions informed by data collected and analyzed in preceding epochs. The benefits of adaptive design are diluted by the time required for inference and design computation.

neural net quickly incorporates readings and generates adaptive designs [26,27]. An advantage of this approach is that the resulting design decisions are nonmyopic; they take future measurements into account. Another approach involves using simplifying assumptions to reduce the overall computational complexity of utility calculations. Some authors use the Laplace approximation to transform the double integral in the utility calculation to a single integral [11,28,29] and others utilize approximations as surrogates for calculating the entire utility function [5,30,31]. The use of an approximate surrogate utility function is a powerful option if a sufficiently simple and accurate surrogate can be found.

Computation time can also be reduced by employing highly parallel computational hardware. While CPU-based computational approaches are versatile and relatively easy to program for data acquisition, they do not take advantage of parallel processing. Graphics processing unit (GPU) modules specialize in performing many simple individual instructions in parallel and GPUs have been used to perform calculations for Bayesian inference [32]. However, so far, the complex nature of the utility calculation [4,33] has not been adapted to computations in a GPU. The field programmable gate array (FPGA), on the other hand, can be configured to possess many parallel computational pipelines capable of complex operations including random sampling and advanced arithmetic. As with the GPU, FPGAs have already been used to accelerate Bayesian inference calculations, offering results in microseconds [34–37]. FPGA programming is nontrivial, however, and the limited availability of floating-point resources on FPGAs have left the task of parallelizing the challenging likelihood computations unfulfilled.

In previous work, we have used low-fidelity utility calculations [7] with a few hundred discrete setting values and relatively few parameter samples to make design decisions in a few tens of milliseconds on conventional CPUs. The quality of the resulting design choices and the efficiency gains have been demonstrated in live experiments [38,39]. In simulated Ramsay measurements with realistic timing, these low-fidelity utility calculations have also compared well against adaptive heuristics and non-adaptive
protocols [40].

Here, we explore the limits of simplification by testing four Bayesian-inspired surrogate utility algorithms on two physics-related parameter-estimation problems. The most complex algorithm computes the information-theoretic utility, equivalent to the estimated change in parameter information entropy, based on many measurement predictions. The simplest algorithm uses only a difference of two measurement predictions. Surprisingly, we find that our four algorithms achieve similar results on a per-measurement basis, while the simplest utility algorithm provides design decisions in under a millisecond, orders of magnitude faster than the information-theoretic version.

Section II provides background, including an overview of the sequential design approach, Bayesian inference methods, and the information-theoretic utility function. Section III presents the numerical utility algorithms used in testing and computational methods. The results are presented and discussed in Sec. IV, followed by conclusions and outlook in Sec. V.

II. BACKGROUND

We use a simulation-based Bayesian experiment design [41,42] approach following the examples provided by Granade *et al.* [43] and Huan and Marzouk [10]. The measurement run comprises a sequence of measurement cycles, or epochs, depicted in Fig. 1, each including a design process to select measurement settings, a measurement using those settings and yielding new data, and Bayesian inference to incorporate the new data.

The goal of the measurement run is to provide precise estimates of the parameters. The Bayesian approach is treats model parameters $\theta = \{\theta_1, \theta_2, \ldots\}$ as random variables with joint distribution $P(\theta)$ and uses information from measurement data to narrow $P(\theta)$ around the true values. After n measurements, we wish to infer the posterior parameter distribution $P(\theta|\mathbf{y}_n, \mathbf{d}_n)$ given accumulated data $\mathbf{y}_n \equiv \{y_1, y_2, \ldots y_n\}$ measured with chosen settings $\mathbf{d}_n \equiv \{d_1, d_2, \ldots d_n\}$. Bold font indicates accumulated data, while italic font indicates data from a single epoch. The inference can be performed iteratively using Bayes' rule,

$$P(\theta|\mathbf{y}_n, \mathbf{d}_n) \propto P(y_n|\theta, d_n)\, P(\theta|\mathbf{y}_{n-1}, \mathbf{d}_{n-1}) \qquad (1)$$

beginning with prior parameter distribution $P_0(\theta)$. The posterior $P(\theta|\mathbf{y}_n, \mathbf{d}_n)$ then becomes the prior in the subsequent epoch. For more compact notation, we define $P_n(\theta) \equiv P(\theta|\mathbf{y}_n, \mathbf{d}_n)$.

The likelihood, $P(y_n|\theta, d_n)$ in Eq. (1), is the probability of experimental outcome y_n given parameters θ and

setting design d_n. As an example, quantum models predict outcomes with probabilities that depend on parameters and external controls. In another example, a Poisson distribution might model photon-count outcomes from a theoretical mean count rate that depends on parameters and settings. In simulation-based Bayesian experiment design, $P(y_n|\theta, d_n)$ is the measurement model expressed as a distribution of measurement outcomes, with the distribution depending on parameters and settings.

For concreteness, we restrict attention to a large class of measurements where outcomes can be modeled as the sum of a mean value and a random noise variable,

$$y = f(\theta, d) + \eta. \tag{2}$$

Here, y is the measurement output, $f(\theta, d)$ models the mean measurement value as a function of model parameters θ and experimental settings (design) d, and η models measurement noise. The model function $f(\theta, d)$ may be nonlinear and nonmonotonic but it is assumed to be a well-behaved and accurate model of the mean measurement results.

The probability of observing outcome y_n is the same as the probability that the noise value satisfies Eq. (2),

$$P(y_n|\theta, d_n) = P_\eta[y_n - f(\theta, d_n)]. \tag{3}$$

For still more concreteness, we assume that measurement noise is normally distributed with standard deviation σ_η. The likelihood of result y_n is then given by

$$P(y_n|\theta, d_n) = \frac{1}{\sqrt{2\pi}\sigma_\eta} e^{[y_n - f(\theta, d_n)]^2/2\sigma_\eta^2}, \tag{4}$$

providing a quantitative answer to the question "How well do the parameters explain the data?"

The main topic of this paper is the task of selecting a design d_{n+1} that provides the greatest predicted improvement in the parameter distribution. Once the desired qualities of the posterior distribution are defined, a utility function $U(d')$ is the quality or amount of improvement predicted for a measurement using setting d'. For example, one option would be to use variance or, more generally, the determinant of the covariance of $P_{n+1}(\theta)$, as a measure of width. It would follow that the inverse of that determinant can be used as a utility function. Ryan [4] reviews related methods that use functionals of the posterior distribution as a utility.

Here, we follow conventional Bayesian experiment design [1,2], choosing information entropy as a measure of the parameter distribution width. The entropy $\mathcal{H}$ of a distribution $P(x)$ is given by

$$\mathcal{H}_x[P] = -\int P(x) \log(P(x))\ dx. \tag{5}$$

The relative entropy in going from $P_n(\theta)$ to $P_{n+1}(\theta)$ is the Kullback-Liebler divergence,

$$K_{\mathrm{LD}} = -\int P_{n+1}(\theta) \log\left(\frac{P_n(\theta)}{P_{n+1}(\theta)}\right) d\theta, \tag{6}$$

where $P_n(\theta)$ is the parameter distribution after n epochs and $P_{n+1}(\theta)$ is the projected distribution after incorporating the data from epoch $n+1$.

An advantage of the K_{LD} as a utility is that it can be rewritten as a functional of distributions of measurement outcomes y', eliminating the need to calculate posterior distributions. Using the Bayes rule, $P_{n+1}(\theta)$ can be expanded as

$$\begin{aligned} P_{n+1}(\theta) &\equiv P(\theta|y', d', \mathbf{y}_n, \mathbf{d}_n) \\ &= \frac{P(y'|\theta, d')}{P(y')} P(\theta|\mathbf{y}_n, \mathbf{d}_n) = \frac{P(y'|\theta, d')}{P(y')} P_n(\theta), \end{aligned} \tag{7}$$

for a predicted result y' using a candidate setting design d'. The prime notation indicates speculative values as opposed to values d and y, which are measurement records. With this expression for $P_{n+1}(\theta)$, the Kullback-Leibler divergence in Eq. (6) would then be written as

$$K_{\mathrm{LD}}(d', y') = -\int \frac{P(y'|\theta, d')}{P(y')} P_n(\theta) \log\left[\frac{P(y')}{P(y'|\theta, d')}\right] d\theta. \tag{8}$$

Averaging over the possible y' predicted values yields the utility $U(d')$:

$$\begin{aligned} U(d') = &-\int P(y'|d') \log\left[P(y'|d')\right] dy' \\ &+ \int P_n(\theta) \left\{\int P(y'|\theta, d') \log\left[P(y'|\theta, d')\right] dy'\right\} d\theta, \end{aligned} \tag{9}$$

with

$$P(y'|d') = \int P(y'|\theta, d') P_n(\theta)\ d\theta. \tag{10}$$

Equation (9) is the conventional information-theoretic utility of a candidate setting and it is the starting point for the investigations in this paper.

The utility in Eq. (9) has an intuitive interpretation. In the second term of Eq. (9), the distribution $P(y'|\theta, d')$

appearing in the bracketed integral is the forecast distribution of measured values given a set of parameters θ and design d'. But with θ and d' fixed, the arguments of the model function are all fixed and the distribution of y' values is the noise distribution, offset by the model function value [see Eq. (3)]. Therefore, the second term may be interpreted as the (negative) entropy of the measurement noise, averaged over parameters.

The first term in Eq. (9) is also an entropy of the predicted measurement-value distribution but here the distribution includes both noise effects and the effects of the parameter distribution as written explicitly in Eq. (10). In total, the utility is the entropy of possible measurement outputs produced by parameter uncertainty and noise, discounted by the entropy of noise.

The intuitive interpretation of Eq. (9) is simply that useful settings are those where the spread or dispersion of the parameter distribution produces large variations in measurement predictions relative to the measurement noise. In Sec. III, we present utility variations that all share this interpretation but differ in their metrics for dispersion.

III. METHODS

A. Utility expressions

Here, we explore utility algorithms based on, or at least inspired by, Eq. (9), with the goal of practical application in laboratory settings. To be useful, design algorithms must produce benefits of measurement efficiency that outweigh the cost of calculation. To meet that goal, we place a heavy emphasis on simplicity and speed.

In pursuit of simplicity and speed, we also permit ourselves to invent trial utility functions without rigorous derivation. In doing so, we sacrifice understanding what statistic (if any) is optimized the trial utilities. We argue that the penalty for less-than-perfect design choices is a mild decrease in measurement efficiency, without invalidating the measurement results. Acceptable measurements are, after all, frequently made using completely arbitrary settings. So while we relax requirements for rigor in the formulation of utility algorithms, we are careful to monitor precision and accuracy of the parameter estimation results in the results presented below.

In the following subsections, we draw on the qualitative interpretation of Eq. (9) as an expression of measurement-value dispersion relative to noise dispersion and we create candidate utility functions that replace entropy with simpler measures of dispersion. Probability distributions are implemented using particle filter methods described in Appendix A. Pseudocodes for the algorithms are provided in Appendix B.

1. KLD utility algorithm

This algorithm attempts to approximate Eq. (9) directly. For the first term, N_s measurement outcomes are simulated using the measurement model given in Eq. (2) with N_s parameter samples drawn from the prior $P_n(\theta)$ and N_s noise values drawn from the noise distribution $P(\eta)$. The simulated outcomes are samples from $P(y'|\theta, d')$. The differential entropy is estimated from these samples using either Ebrahimi's method [44] or Vasicek's method [45]. For the examples in this paper, the noise is normally distributed with standard deviation σ_η, independent of θ and d, so the second term in Eq. (9) can be determined from the known entropy of a normal distribution:

$$\mathcal{H}_\eta = \frac{1}{2}\log(2\pi e\sigma_\eta^2), \tag{11}$$

where $\log(e) = 1$.

2. Variance algorithm

For the variance algorithm, we choose the logarithm of variance as a measure of dispersion [46], yielding the utility

$$U^{\text{var}}(d') = \frac{1}{2}\log\left[v_\theta(d') + v_\eta\right] - \frac{1}{2}\log(v_\eta), \tag{12}$$

which simplifies to

$$U^{\text{var}}(d') = \frac{1}{2}\log\left[1 + \frac{v_\theta(d')}{v_\eta}\right]. \tag{13}$$

Here, v_η is the variance of the noise distribution and $v_\theta(d')$ is the variance of a distribution of noise-free model values, $P(y'|d', \eta)|_{\eta=0}$. Samples of this distribution are provided by the model function $f(\theta_i|d')$ using sampled parameter vectors $\{\theta_1 \ldots \theta_{N_s}\}$. The first term in Eq. (12) represents the combined effects of parameter uncertainty and noise and the second term isolates the contribution due to noise alone. The variance utility also corresponds to the Kullback-Leibler utility given in Eq. (9) in the special case where all distributions are normal.

A feature of the variance algorithm is that the variances of the parameter distribution and noise distributions are simply additive, so that effects of measurement noise and the effects of parameter dispersion are separated. In contrast, the contributions to entropy are not easily distinguished. Another advantage is that variance computes faster than entropy.

3. Pseudo-H utility algorithm

Recognizing that standard deviation is not always a good proxy for entropy, the pseudo-H utility attempts to recapture some of the properties of entropy as a dispersion measure while preserving the separation of parameter effects and noise effects provided by variance. With the

same form as the variance utility, the pseudo-H utility is defined as

$$U^{\mathrm{ps}H}(d') = \frac{1}{2}\log\left[1 + \frac{v_{\mathcal{H}}(d')}{v_\eta}\right], \tag{14}$$

where $v_{\mathcal{H}}$ is an effective variance derived from the entropy of the distribution of noise-free model values. As in the variance algorithm, model values are generated by the model function $f(\theta_i|d')$ for parameter samples $\{\theta_1 \ldots \theta_{N_s}\}$ and $\eta = 0$. The entropy of the distribution represented by the model values is calculated [44,45] and then the entropy is converted into an effective variance $v_{\mathcal{H}}$ using Eq. (11), yielding

$$v_{\mathcal{H}} = (2\pi e)^{-1}\exp[2\mathcal{H}_{y'}(d', \eta = 0)]. \tag{15}$$

The pseudo-H utility is pieced together without a strong claim for validity but it has been used in previous work [40].

4. Max-min utility algorithm

The max-min utility is designed for simplicity and fast computation using the range statistic as a measure of dispersion. A relatively small set of samples θ_j, $\{\theta_1 \ldots \theta_{N_s}\}$ are drawn from $P(\theta)$ and a corresponding set of N_s model values $\{y(d')\} \equiv \{y_1(d') \ldots y_{N_s}(d')\}$ are calculated using $y_j(d') = f(\theta_j, d')$. The range $t(d')$ is defined as the difference between the maximum and minimum values in the set,

$$t(d') = \max(\{y(d')\}) - \min(\{y(d')\}). \tag{16}$$

Again following the form of the variance utility, the max-min utility is defined as

$$U^{\mathrm{mm}}(d') = \frac{1}{2}\log\left[1 + \frac{t(d')^2}{v_\eta}\right]. \tag{17}$$

5. Random design

The random-design algorithm is included as a baseline nonadaptive protocol for comparison with the adaptive algorithms listed above. As its name suggests, the random design chooses randomly and with uniform probability from among the candidate settings, uninfluenced by measurement data. A more intuitive choice to represent nonadaptive protocols might be repeated scans through settings but the random design avoids artifacts that might be generated by periodic repetition of the settings.

B. Sample reuse

Computational methods have been proposed by several authors to compute the double integral in the utility expression in Eq. (9) with sufficient precision [10,11,47–49]. To simplify calculation of Eq. (9), Huan and Marzouk [10] have proposed using the same set of parameter samples both for calculation of Eq. (10) and for the averaging over parameters in Eq. (9). Where possible, we take this approach one step further by using one set of samples to calculate utility for all candidate settings. Note that only the KLD algorithm requires noise samples. The variance, pseudo-H and max-min algorithms only require the noise variance.

Figure 2 illustrates how sample reuse affects the utility function. Figures 2(a) and 2(b) plot samples from $P(y'|x)$ for a Lorentzian model function

$$y' = \frac{1}{[(x - x_0)/0.1]^2 + 1} + \eta, \tag{18}$$

with setting or design variable x and the peak center parameter x_0 and added noise η both normally distributed. Figure 2(a) plots simulated measurement outcomes, where N_s parameter samples and N_s noise samples are drawn for each of N_d candidate designs, requiring $2N_sN_d$ samples [$N_s = 100$; $N_d = 200$]. Figure 2(b) represents the same distribution but a single set of x_0 samples and η samples are reused for all values of design variable x, requiring only $2N_s$ samples. Figure 2(c) shows the effect of these contrasting approaches on the utility function using the KLD algorithm described below. The use of new samples for each setting value ensures that

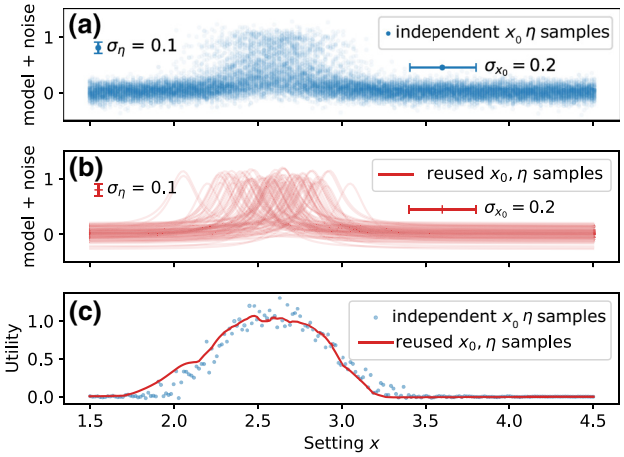

FIG. 2. An illustration of the effects of parameter and noise-sample reuse on utility-function estimation. The model function is a Lorentzian given by Eq. (18). (a),(b) Distributions of predicted measurement values $P(y'|x)$ using sets of $N_s = 100$ samples each from normal distributions of center x_0 and added noise η. Standard deviations are indicated by bars. In (a), a fresh set of samples is drawn for each value of setting x, while in (b), a single set of samples is reused for all x. (c) Corresponding estimates of utility, both yielding maxima near $x = 2.6$.

the utility values are statistically independent but introduces sampling noise that makes the maximum difficult to locate [4].

The computational simplification allowed by sample reuse reduces the demand for weighted sampling but the smoothing effect shown in Fig. 2(c) may be more important. Given a consistent set of parameter and noise samples across all d, the qualitative smoothness of the calculated utility will persist even for small sample numbers N_s. Since the required number of measurement simulations is $N_s \times N_d$, small N_s will allow faster computation. Of course, the calculated utility values may be increasingly inaccurate for small N_s and the resulting design choices may lead to poor performance. We assess the effects of low N_s empirically in testing in Sec. IV. Unless noted otherwise, we use $N_s = 1000$.

IV. RESULTS AND DISCUSSION

A. Lorentzian peak

In this subsection, the simulated experiment is a Lorentzian absorption profile, with a single "unknown" peak-center parameter $\theta \equiv \{x_0\}$ and setting $d \equiv x$:

$$f_L(x_0, x) = b + \frac{a}{[(x - x_0)/\Delta]^2 + 1}. \tag{19}$$

The background b, the amplitude a, and the half-width Δ are constants. The maximum Fisher information for the Lorentzian is found at $x = x_0 \pm \Delta/\sqrt{3}$ and the Cramer-Rao bound is given by

$$\sigma_{x_0} \geq \frac{8}{3\sqrt{3}} \frac{\Delta}{a} \frac{\sigma_\eta}{\sqrt{n}}. \tag{20}$$

The Lorentzian function describes the frequency response of a damped harmonic oscillator, which appears frequently in many branches of science and engineering. Previous work has demonstrated the effectiveness of sequential Bayesian experiment design in a laboratory experiment, estimating five parameters from measurements on triplets of Lorentzian peaks. Compared to a swept-setting approach, the Bayesian method has required 40-fold fewer measurements to reach a comparable uncertainty [38].

Figure 3 presents results of simulated measurement runs using the adaptive Bayesian design methods along with random design as a benchmark nonadaptive design. The simulations use constants $b = 50\,000$, $a = -1000$, and $\Delta = 0.1$. The measurement data are simulated using true-value $\theta^* = \{x_0 = 2.6\}$ and the measurement noise is normally distributed with $\sigma_\eta = 1000$. The prior $P_0(x_0)$ is normal with mean 3.0 and standard deviation 0.5 and the parameter distribution is implemented as a particle filter with 1×10^4 particles.

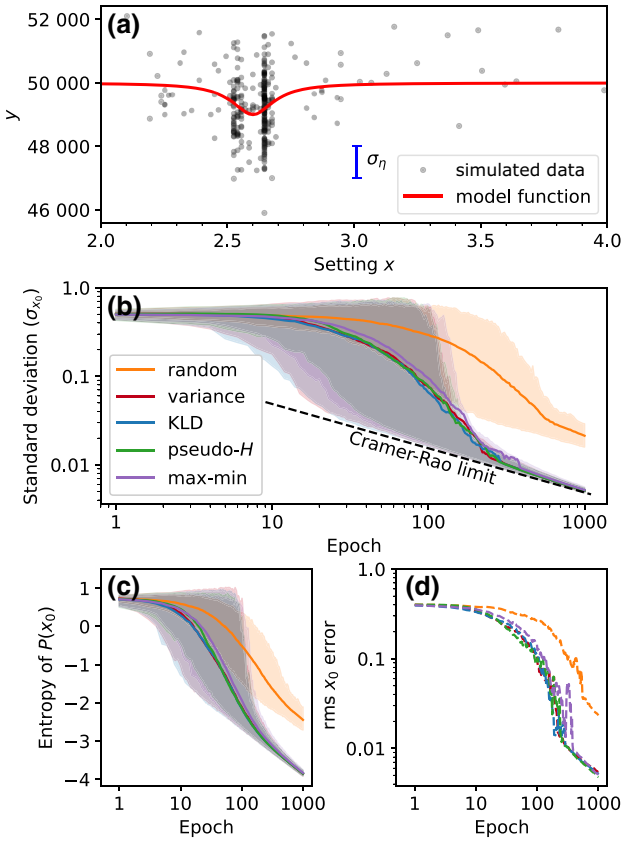

FIG. 3. Simulated measurement runs to determine the center of a Lorentzian. (a) The first 300 simulated data points of a selected run. The Lorentzian model function is plotted in red with the center parameter $x_0 = 2.6$ and the standard deviation of the noise is equal to the peak height. The adaptive design initially calls for settings over a broad range but later repeats measurements on the sides of the peak after approximately 100 epochs. (b),(c) The standard deviation (b) and entropy (c) of $P(x_0)$ calculated using various utility algorithms. The mean (solid lines) and 5% to 95% credibility intervals (shaded areas) are calculated from 400 runs. The gray areas result from coincidence of the credibility intervals. (d) The root-mean-square (rms) error. The variance, KLD, and pseudo-H algorithms use $N_s = 1000$ parameter samples; max-min uses $N_s = 2$.

Figure 3(a) shows data from a single run, plotted with the model "true" curve. After approximately 90 epochs in this run, the design algorithm focuses on settings close to the Fisher information maxima on the slopes of the peak or dip.

Figures 3(b)–(d) support some of the main conclusions of this paper. In summaries of 400 measurement runs, Figs. 3(b), 3(c), and 3(d), respectively, trace the standard deviation and entropy of the parameter distribution and the rms error of the mean.

A striking feature of Fig. 3 is that virtually identical results are obtained regardless of which utility algorithm is used. The Bayesian designs all follow similar approaches

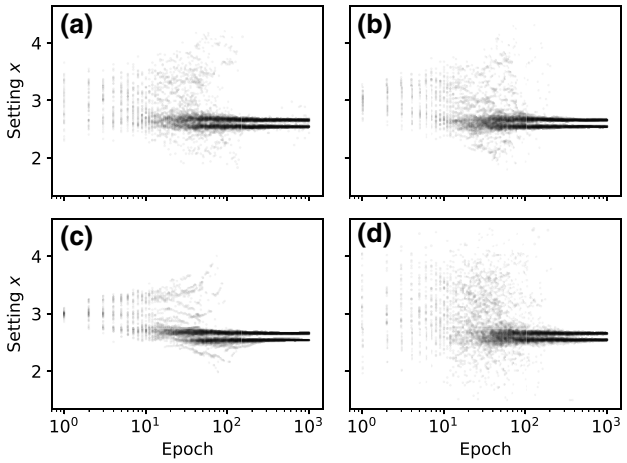

FIG. 4. For each of the utility algorithms, setting values from 50 runs are superposed, demonstrating similar overall patterns in design choices: (a) KLD; (b) variance; (c) pseudo-H; (d) max-min. Epochs > 50 are subsampled for uniform appearance on the logarithmic scale.

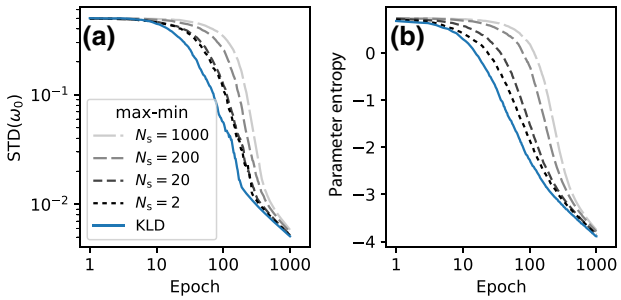

FIG. 5. Tests of the max-min algorithm for varying numbers of parameter sample N_s including (a) the standard deviation and (b) the entropy of the x_0 parameter distribution. The performance approximates the KLD algorithm for small N_s.

to the Cramer-Rao limit given in Eq. (20) after a few hundred measurements. Compared to uninformed Random setting selection, all adaptive designs achieve σ_{x_0} values that are a factor of 4 lower, effectively requiring $1/16$ as many measurements to reach the same level of uncertainty.

Similarities in the results are underscored by similarities in the algorithm-chosen settings, as shown in Fig. 4. Each run uses different random numbers to represent the prior and to model noise values, so different design choices are expected in different runs. However, patterns in the design choices have strong similarities across the various utility algorithms, with an initial focus on x values close to the center of the prior distribution at $x_0 = 3.0$, a broadening of the setting choices, and rapid convergence to the settings for Fisher information maxima.

Figure 5 highlights the behavior of the max-min utility algorithm and its dependence on the number of parameter samples N_s. The max-min algorithm uses the range of model-function outputs over the parameter samples as a crude measure of dispersion due to parameter uncertainty. Atypically, the max-min algorithm provides poorer performance with larger numbers of parameter samples. Figure 5 shows that the use of $N_s > 2$ makes less efficient design decisions, despite more thorough sampling of parameters. We speculate that the relative inefficiency of $N_s > 2$ occurs because the max-min algorithm disregards representative parameter values from the center of a distribution when $N_s > 2$ and instead takes direction from the atypical extreme values.

Figure 6 shows the computation time required for the design algorithms and selected subtasks in 1000-epoch runs [50]. Each of the design calculations requires computation of the model function using $N_d = 200$ candidate designs for each of $N_s = 1000$ parameter samples, followed by N_d calculations of a statistic over N_s samples. The exception is the max-min algorithm using only $N_s = 2$ parameter samples, which is unsurprisingly much faster.

B. Decaying sinusoid

This subsection presents simulations of measurements to determine four parameters of a decaying sinusoid model function:

$$f\left(\{h, c, \omega_0, T_2\}, \tau\right) = h + c\ \cos(\omega_0 \tau)\ e^{-(\tau/T_2)^2}, \qquad (21)$$

where $\theta \equiv \{h, c, \omega_0, T_2\}$ are "unknown" parameters corresponding, respectively, to a background level, contrast, mean angular frequency, and dephasing with true values $h^* = 0.8, c^* = 0.13, \omega_0^* = 9.4$, and $T_2^* = 10$. The lone setting is $d \equiv \tau$. For each measurement, the system is initialized, the oscillations are started at $t = 0$, and a single measurement is allowed after the selected delay time τ.

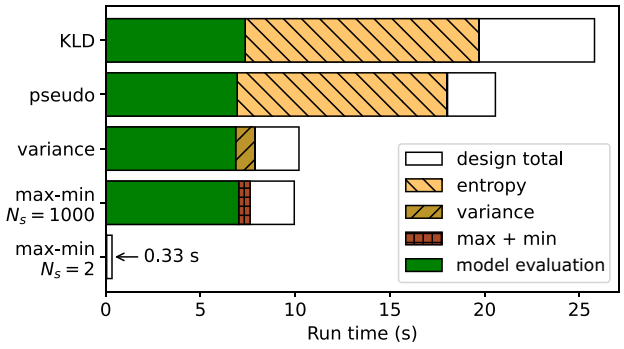

FIG. 6. A comparison of the design computation times among the utility algorithms in 1000-epoch runs. The runs simulate measurements to determine the center, x_0, of a Lorentzian peak using 200 candidate settings. The overall bar length indicates the total design computation time, with subtasks shown as segments. The model function and entropy calculations consume the majority of the total time. The max-min $N_s = 2$ algorithm executes in 0.33 ms per epoch.

For optimal measurements of ω_0, the maximum slope $df/d\omega_0$ occurs for $\tau \approx T_2/\sqrt{2}$ and $\sin(\omega_0\tau) \approx 0$. For n repeated measurements at this setting,

$$\sigma_{\omega_0} = \frac{\sqrt{2e}}{cT_2}\frac{\sigma_y}{\sqrt{n}}. \tag{22}$$

For optimal measurements of T_2, the maximum slope df/dT_2 occurs for $\tau \approx T_2$ and $\sin(\omega_0\tau) \approx \pm 1$.

Equation (22) is derived under the assumption that the uncertainty is limited by measurement noise but this approach neglects ambiguity. Even a noise-free measurement cannot eliminate uncertainty, since many values of ω_0 can produce the same value of y.

The Ramsey measurement modeled by Eq. (21) is the canonical method for measuring energy differences $\Delta E = \hbar\omega_0$ between quantum states [51] and it is closely related to the problem of measuring quantum phase. Interest in Ramsey measurement protocols is driven by the possibility that high-fidelity measurements allow ΔE to be measured (i.e., disambiguated) with precision $\Delta E \propto 1/\Delta t$ in a time Δt [14,15]. This behavior is termed Heisenberg scaling in analogy with the Heisenberg uncertainty principle, $\Delta E \Delta t \geq \hbar/2$. When readout fidelity is high but not perfect, Heisenberg scaling can be approached using adaptive [16–20] and by optimized nonadaptive [21–23] measurements.

Figure 7 presents the results of Ramsey sequence simulations. Figure 7(a) plots the "true" model function given in Eq. (21) and simulated data using true values $h^* = 0.8, c^* = 0.13, \omega_0^* = 9.4$, and $T_2^* = 10$ and normally distributed noise, with $\sigma_\eta = 0.13$. The parameter distribution is implemented by a particle filter with 5×10^4 particles. The settings have 1991 allowed values covering the range $0.1 \leq \tau \leq 20$ in steps of 0.01. Figure 7(b) compares the standard deviation of ω_0 for various utility algorithms as a function of the number of measurement epochs.

As with the results of the Lorentzian peak example, the utility algorithms achieve similar performance on a per-measurement basis. In contrast to the Lorentzian model results in Fig. 3, where there are large performance differences between the adaptive algorithms and random parameter selection, the random design applied to the Ramsey model achieves similar performance to adaptive designs after approximately 400 epochs. We attribute the relatively good performance of random setting selection here to the feature-rich structure of the Ramsey model function and the larger number of parameters. Taken together, these factors create a situation where virtually any selection of τ is somewhat informative. The broad distributions of settings chosen by adaptive designs also suggest that utility is not concentrated on a few settings in the decaying sinusoid measurement.

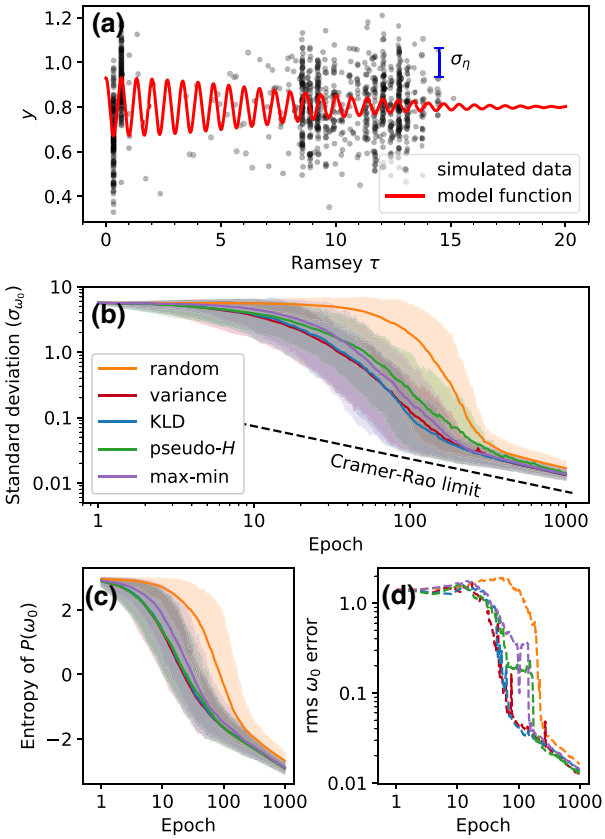

FIG. 7. Simulated measurement runs to determine four parameters of a decaying sinusoid. (a) The model function is plotted in red as a function of the setting τ. The standard deviation of the noise is equal to the oscillation amplitude at $\tau = 0$. The adaptive design initially chooses low τ settings but later favors $\tau \approx T_2$. Data from epochs later than 50 have been subsampled for clarity. (b),(c) The standard deviation (b) and entropy (c) of $P(\omega_0)$ of simulated measurements using various utility algorithms. The mean (solid lines) and 5% to 95% credibility intervals (shaded areas) are calculated from 400 runs. (d) The rms error.

The settings chosen by the algorithms are plotted in Fig. 8. Strong similarities between the designs are visible, despite differences in the algorithms. All the design algorithms begin with low-τ measurements and increase τ gradually. After 100 epochs, the algorithms concentrate settings at low τ and distribute settings in the region roughly $5 \lesssim \tau \lesssim 15$; for example, within the boxed region in Fig. 8(a).

Within the distributed settings, there are strong correlations between the τ values and the sinusoid phase. In Fig. 7(a), close examination shows that the low-τ settings are correlated with maxima and minima of the model function. For $8 \lesssim \tau \lesssim 11$, the τ values are correlated with zero crossings, suggesting optimal settings for determining ω_0. Also, τ values above 11 are correlated with maxima and minima, suggesting optimal settings for determining T_2.

Figure 9 shows the computation time required for the design algorithms and selected subtasks in 1000-epoch

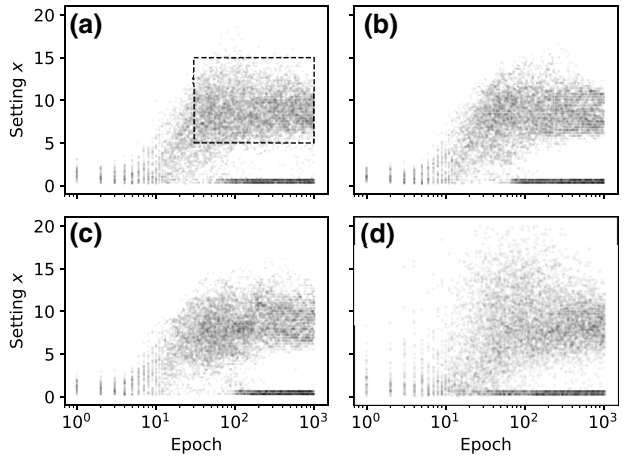

FIG. 8. The settings chosen by the indicated utility algorithms in the Ramsey simulations, plotted for 50 runs each: (a) KLD; (b) variance; (c) pseudo-H; (d) max-min. In later epochs, the algorithms choose small τ and also τ distributed across ranges; for example, the range indicated by the dashed rectangle in (a). The chosen τ values tend to fall at local extrema of the model function and local extrema of slope. These correlations are more clearly visible in Fig. 7(a).

runs [50]. Each of the design calculations requires computation of the model function using $N_d = 1991$ candidate designs for each of $N_s = 1000$ parameter samples, followed by N_d calculations of a statistic over N_s samples. The exception is the max-min algorithm, which uses only $N_s = 2$ parameter samples.

V. CONCLUSIONS AND OUTLOOK

This paper compares results of simulated measurement runs using two nonlinear experiment models and four variations of the information entropy utility using different statistical functions but all treating the utility as a measure of the dispersion of modeled measurement results relative

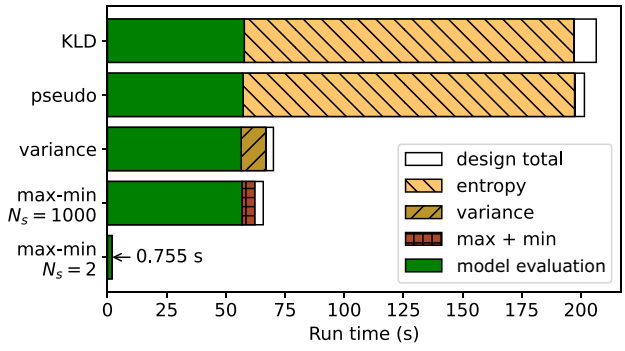

FIG. 9. A comparison of the design task computation among the utility algorithms in 1000-epoch runs using the decaying sinusoid model with four parameters and 1991 setting values. Within the total time, the model function and entropy calculations are the most costly parts of the algorithms. The max-min $N_s = 2$ algorithm executes in 0.8 ms per epoch.

to the dispersion of measurement noise. These surrogate utilities avoid likelihood calculations that would be needed for utilities defined as functionals of the posterior distribution [4]. Testing under conditions where the signal-to-noise ratio (SNR) is approximately 1, the four utility algorithms generated similar design choices and similar reductions in parameter variance, entropy, and rms error but the simplest algorithm provides a manyfold reduction in computation time.

A key component of our utility algorithms is the reuse of parameter samples for all candidate designs. The benefits of this technique are, first, that the utility estimates become smooth functions of the setting value with easily located maxima and, second, that fewer random parameter samples are required. Although these benefits come at the cost of a systematic error, the effects of the systematic error are transient, because new parameter samples are drawn for each epoch.

We point to the max-min algorithm with $N_s = 2$ as an important outcome of this study. This algorithm pushes the utility-as-dispersion idea to its simplest extreme, calculating the model function for each candidate design and only two parameter samples. On a consumer-grade CPU, this simplified utility algorithm executes in submillisecond times, which is on par with the time spent performing Bayesian inference on incoming data and which approaches latency times for USB and Ethernet communication with instruments. Considering only separate instruments controlled by CPU-based computing, these results show that Bayesian experiment design is not prohibitively expensive to calculate.

Ultimately, signal generation and detection physics will determine whether utility calculations are fast enough for a particular application. We argue that the time required to integrate signal for SNR ≈ 1 is a more relevant experimental time scale than, for example, the single-reading acquisition time. Within the sequential design scheme, there is flexibility in the definition of a measurement for a single epoch. A measurement defined such that SNR $\ll 1$ does not change the parameter distribution enough to justify recalculating the utility. At the other extreme, long measurements with SNR $\gg 1$ may miss opportunities to change to more profitable settings.

Systems that generate data with SNR ≈ 1 in microsecond or nanosecond time scales will require utility calculations to be embedded in the instrumentation to avoid communication latency. Since instrumentation designs incorporating FPGAs are becoming more common, FPGA-based experiment design appears to be a convenient and fast choice, potentially offering design calculations in microseconds.

We speculate that the max-min $N_s = 2$ algorithm will be well suited for implementation on FPGAs or other dedicated hardware with fixed-point arithmetic, because the max-min statistic requires relatively few subtraction and

absolute value operations. However, practical implementation of a dedicated system would still face challenges in implementing a parameter probability-density representation and mathematical operations to support model function and likelihood calculations.

The PYTHON scripts and data used in this paper are available online [52].

APPENDIX A: PARTICLE FILTER

The distribution of parameters $P(\theta)$ encapsulates the state of belief regarding different parameter values. For computational work, methods known variously as particle filters, swarm filters, importance sampling, or sequential Monte Carlo methods have gained prominence as representations of the probability distribution [53–56]. For a model with D parameters, the distribution is represented by N_p "particles" each with a vector θ corresponding to a point in D-dimensional parameter space and a corresponding weight w such that the distribution is approximated by

$$P(\theta) \approx \sum_{k=1}^{N_p} \delta(\theta - \theta_k) w_k \tag{A1}$$

with $\sum_i w_i = 1$. Here, $\delta(\cdot)$ is the Dirac delta function. In inference calculations, weights are multiplied by computed likelihoods to generate unnormalized weights $W_{i,n}$:

$$W_{i,n} = w_{i,n-1} P(y_n|\theta_i, d_n), \tag{A2}$$

yielding posterior weights

$$w_{i,n} = W_{i,n} / \sum_i W_{i,n}. \tag{A3}$$

Resampling is a necessary maintenance for particle filters, which essentially reassigns computer memory from low-probability regions of parameter space to high-probability regions. Resampling is initiated whenever the effective number of particles,

$$N_{\text{eff}} = 1/\sum w_i^2, \tag{A4}$$

falls lower than a threshold, typically set at $0.5N_p$. The covariance matrix $\mathbf{C}$ of the distribution is calculated, then N_p samples $\tilde{\theta}_j$ are drawn with replacement from the $\{\theta_i\}$ set with probability w_i. This selection process will tend to miss low-weight particles, while possibly choosing high-weight particles multiple times. The particles are then given small random displacements $\delta\theta$ to separate degenerate particles. The resampled set of particles θ_j are given by

$$\theta_j = \tilde{\theta}_j + \delta\theta_j, \tag{A5}$$

where $\delta\theta_j$ is a sample from a multivariate normal distribution with covariance matrix $\alpha\mathbf{C}$. We use $\alpha = 0.01$.

APPENDIX B: PSEUDOCODE

Input: Parameter distribution $P(\theta)$
Input: Noise distribution $P(\eta)$
  for $j \leftarrow 1 \ldots N_s$ **do**
    Sample θ_j from $P(\theta)$ parameter distribution
    Sample η_j from $P(\eta)$ from noise distribution
    for all candidate settings d_i **do**
      Simulate measurement $y_{i,j} \leftarrow \text{model}(\theta_j, d_i) + \eta_j$
      ▷ Reusing θ_j and η_j
    end for
  end for
  for all candidate settings d_i **do**
    Estimate entropy from samples $h_{y,i} = \mathcal{H}(y_{i,1} \ldots y_{i,N_s})$
    Estimate entropy of noise distribution $h_{\eta,i} \leftarrow \mathcal{H}[P(\eta)]$
    Calculate utility $U_i^{\text{KLD}} \leftarrow h_{y,i} - h_{\eta,i}$
  end for
  Find max utility $i_{\text{best}} \leftarrow \text{argmax}(U_i^{\text{KLD}})$
Output: corresponding setting $d_{i_{\text{best}}}$

Algorithm 1. KLD utility algorithm

Input: Parameter distribution $P(\theta)$
  for N_s parameter samples θ_j drawn from $P(\theta)$ **do**
    for all candidate designs, d_i **do**
      Evaluate model $y_{i,j} \leftarrow f(\theta_j, d_i)$
      ▷ reusing θ_j samples
    end for
  end for
  for all candidate designs d_i **do**
    Variance $v_{\theta,i} \leftarrow \text{Var}(y_{i,1} \ldots y_{i,N_s})$ over parameters
    Variance of noise $v_{\eta,i} \leftarrow \text{Var}[P(\eta)]$
    Calculate utility $U_i^{\text{var}} \leftarrow \log(1 + v_{\theta,i}/v_{\eta,i})/2$
  end for
  Find max utility $i_{\text{best}} \leftarrow \text{Argmax}(U_i^{\text{var}})$
Output: corresponding setting $d_{i_{\text{best}}}$

Algorithm 2. Variance utility algorithm

Input: Parameter distribution $P(\theta)$
  for N_s parameter samples θ_j drawn from $P(\theta)$ **do**
    for all candidate designs, d_i **do**
      Evaluate model $y_{i,j} \leftarrow f(\theta_j, d_i)$
      ▷ reusing θ_j samples
    end for
  end for
  for all candidate designs d_i **do**
    Estimate entropy from samples $h_{y,i} = \mathcal{H}(y_{i,1} \ldots y_{i,N_s})$
    Calculate effective variance $v_{\theta,i}^{\text{eff}} \leftarrow \exp(2h_{y,i})/(2\pi e)$
    Variance of noise distribution $v_{\eta,i} \leftarrow \text{Var}[P(\eta)]$
    Calculate utility $U_i^{\text{PU}} \leftarrow \log(1 + v_{\theta,i}^{\text{eff}}/v_{\eta,i})/2$
  end for
  Find max utility $i_{\text{best}} \leftarrow \text{Argmax}(U_i^{\text{PU}})$
Output: corresponding setting $d_{i_{\text{best}}}$

Algorithm 3. Pseudo-utility algorithm

Input: Parameter distribution $P(\theta)$
  for N_s parameter samples θ_j drawn from $P(\theta)$ **do**
    for all candidate designs, d_i **do**
      Evaluate model $y_{i,j} \leftarrow f(\theta_j, d_i)$
                                ▷ reusing θ_j samples
    end for
  end for
  for all candidate designs d_i **do**
    Find maximum model value $y_i^{\max} = \max(y_{i,1} \ldots y_{i,N_s})$
    Find minimum model value $y_i^{\min} = \min(y_{i,1} \ldots y_{i,N_s})$
    Calculate range $t_i \leftarrow y_i^{\max} - y_i^{\min}$
    Variance of noise $v_{\eta,i} \leftarrow \mathrm{Var}[P(\eta)]$
    Calculate utility $U_i^{\mathrm{mm}} \leftarrow \log(1 + t_i^2/v_{\eta,i})/2$
  end for
  Find max utility $i_{\mathrm{best}} \leftarrow \mathrm{Argmax}(U_i^{\mathrm{mm}})$
Output: corresponding setting $d_{i_{\mathrm{best}}}$

Algorithm 4. Max-min utility

[1] D. V. Lindley, On a measure of the information provided by an experiment, Ann. Math. Stat. **27**, 986 (1956).

[2] K. Chaloner and I. Verdinelli, Bayesian experimental design: A review, Stat. Sci. **10**, 273 (1995).

[3] A. M. Overstall, J. M. McGree, and C. C. Drovandi, An approach for finding fully Bayesian optimal designs using normal-based approximations to loss functions, Stat. Comput. **28**, 343 (2018).

[4] E. G. Ryan, C. C. Drovandi, J. M. McGree, and A. N. Pettitt, A review of modern computational algorithms for Bayesian optimal design: A review of modern algorithms for Bayesian design, Int. Stat. Rev. **84**, 128 (2016).

[5] A. M. Overstall, D. C. Woods, and M. Adamou, ACEBAYES: An R package for Bayesian optimal design of experiments via approximate coordinate exchange, J. Stat. Softw. **95**, 1 (2020).

[6] S. Olofsson, L. Hebing, S. Niedenführ, M. P. Deisenroth, and R. Misener, GPdoemd: A PYTHON package for design of experiments for model discrimination, Comput. Chem. Eng. **125**, 54 (2019).

[7] R. D. McMichael, S. M. Blakley, and S. Dushenko, OPTBAYESEXPT: Sequential Bayesian experiment design for adaptive measurements, J. Res. Natl. Inst. Stand. Technol. **126**, 126002 (2021).

[8] F. Ha, PHOENICS: A Bayesian optimizer for chemistry, ACS Cent. Sci. **4**, 1134 (2018).

[9] C. Granade, C. Ferrie, I. Hincks, S. Casagrande, T. Alexander, J. Gross, M. Kononenko, and Y. Sanders, QInfer: Statistical inference software for quantum applications, Quantum **1**, 5 (2017).

[10] X. Huan and Y. M. Marzouk, Simulation-based optimal Bayesian experimental design for nonlinear systems, J. Comput. Phys. **232**, 288 (2013).

[11] Q. Long, M. Scavino, R. Tempone, and S. Wang, Fast estimation of expected information gains for Bayesian experimental designs based on Laplace approximations, Comput. Methods Appl. Mech. Eng. **259**, 24 (2013).

[12] C. Ferrie, C. E. Granade, and D. G. Cory, How to best sample a periodic probability distribution, or on the accuracy of Hamiltonian finding strategies, Quantum Inf. Process. **12**, 611 (2013).

[13] R. Santagati, A. A. Gentile, S. Knauer, S. Schmitt, S. Paesani, C. Granade, N. Wiebe, C. Osterkamp, L. P. McGuinness, J. Wang, M. G. Thompson, J. G. Rarity, F. Jelezko, and A. Laing, Magnetic-Field Learning Using a Single Electronic Spin in Diamond with One-Photon Readout at Room Temperature, Phys. Rev. X **9**, 021019 (2019).

[14] A. Y. Kitaev, Quantum measurements and the Abelian stabilizer problem (1995), ArXiv:quant-ph/9511026.

[15] A. Y. Kitaev, in *Electr. Coll. Comput. Complex.* (1996), p. TR96–003.

[16] B. L. Higgins, D. W. Berry, S. D. Bartlett, H. M. Wiseman, and G. J. Pryde, Entanglement-free Heisenberg-limited phase estimation, Nature **450**, 393 (2007).

[17] D. W. Berry, B. L. Higgins, S. D. Bartlett, M. W. Mitchell, G. J. Pryde, and H. M. Wiseman, How to perform the most accurate possible phase measurements, Phys. Rev. A **80**, 052114 (2009).

[18] R. S. Said, D. W. Berry, and J. Twamley, Nanoscale magnetometry using a single-spin system in diamond, Phys. Rev. B **83**, 125410 (2011).

[19] P. Cappellaro, Spin-bath narrowing with adaptive parameter estimation, Phys. Rev. A **85**, 030301(R) (2012).

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

[21] B. L. Higgins, D. W. Berry, S. D. Bartlett, M. W. Mitchell, H. M. Wiseman, and G. J. Pryde, Demonstrating Heisenberg-limited unambiguous phase estimation without adaptive measurements, New J. Phys. **11**, 073023 (2009).

[22] N. M. Nusran, M. U. Momeen, and M. V. G. Dutt, High-dynamic-range magnetometry with a single electronic spin in diamond, Nat. Nanotechnol. **7**, 109 (2012).

[23] G. Waldherr, J. Beck, P. Neumann, R. S. Said, M. Nitsche, M. L. Markham, D. J. Twitchen, J. Twamley, F. Jelezko, and J. Wrachtrup, High-dynamic-range magnetometry with a single nuclear spin in diamond, Nat. Nanotechnol. **7**, 105 (2012).

[24] A. J. F. Hayes and D. W. Berry, Swarm optimization for adaptive phase measurements with low visibility, Phys. Rev. A **89**, 013838 (2014).

[25] H. T. Dinani, D. W. Berry, R. Gonzalez, J. R. Maze, and C. Bonato, Bayesian estimation for quantum sensing in the absence of single-shot detection, Phys. Rev. B **99**, 125413 (2019).

[26] A. Foster, D. R. Ivanova, I. Malik, and T. Rainforth, in *Proceedings of the 38th International Conference on Machine Learning* (PMLR, 2021), p. 3384.

[27] L. J. Fiderer, J. Schuff, and D. Braun, Neural-Network Heuristics for Adaptive Bayesian Quantum Estimation, PRX Quantum **2**, 020303 (2021).

[28] Q. Long, M. Motamed, and R. Tempone, Fast Bayesian optimal experimental design for seismic source inversion, Comput. Methods Appl. Mech. Eng. **291**, 123 (2015).

[29] E. G. Ryan, C. C. Drovandi, and A. N. Pettitt, Fully Bayesian experimental design for pharmacokinetic studies, Entropy **17**, 1063 (2015).

[30] A. Tarakanov and A. H. Elsheikh, Optimal Bayesian experimental design for subsurface flow problems, Comput. Methods Appl. Mech. Eng. **370**, 113208 (2020).
[31] A. M. Overstall and D. C. Woods, Bayesian design of experiments using approximate coordinate exchange, Technometrics **59**, 458 (2017).
[32] F. Lopez, L. Zhang, A. Mok, and J. Beaman, Particle filtering on GPU architectures for manufacturing applications, Comput. Indus. **71**, 116 (2015).
[33] T. Li, M. Bolic, and P. M. Djuric, Resampling methods for particle filtering: Classification, implementation, and strategies, IEEE Signal Process. Mag. **32**, 70 (2015).
[34] B. G. Sileshi, C. Ferrer, and J. Oliver, in *Emerging Trends in Computational Biology, Bioinformatics and Systems Biology – Algorithms and Software Tools*, edited by H. R. Arabnia and Q.-N. Tran (Elsevier Inc., 2015), Ch. 2, p. 19.
[35] B. G. Sileshi, J. Oliver, and C. Ferrer, in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)* (2016), p. 591.
[36] P. Engineer, R. Velmurugan, and S. Patkar, Scalable implementation of particle filter-based visual object tracking on network-on-chip (NoC), J. Real-Time Image Process. **17**, 1117 (2020).
[37] D. Charalampidis, V. P. Jilkov, and J. Wu, in *Signal and Data Processing of Small Targets 2015*, Vol. 9596 (SPIE, 2015), p. 48.
[38] S. Dushenko, K. Ambal, and R. D. McMichael, Sequential Bayesian Experiment Design for Optically Detected Magnetic Resonance of Nitrogen-Vacancy Centers, Phys. Rev. Appl. **14**, 054036 (2020).
[39] M. Caouette-Mansour, A. Solyom, B. Ruffolo, R. D. McMichael, J. Sankey, and L. Childress, Robust Spin Relaxometry with Fast Adaptive Bayesian Estimation, Phys. Rev. Appl. **17**, 064031 (2022).
[40] R. D. McMichael, S. Dushenko, and S. M. Blakley, Sequential Bayesian experiment design for adaptive Ramsey sequence measurements, J. Appl. Phys. **130**, 144401 (2021).
[41] P. Müller, B. Sansó, and M. De Iorio, Optimal Bayesian Dedsign by inhomogeneous Markov Chain simulation, J. Am. Stat. Assoc. **99**, 788 (2004).
[42] P. Müller, in *Handbook of Statistics*, Vol. 25 (Elsevier, 2005), p. 509.
[43] C. E. Granade, C. Ferrie, N. Wiebe, and D. G. Cory, Robust online Hamiltonian learning, New J. Phys. **14**, 103013 (2012).
[44] N. Ebrahimi, K. Pflughoeft, and E. S. Soofi, Two measures of sample entropy, Stat. Probab. Lett. **20**, 225 (1994).
[45] O. Vasicek, A test for normality based on sample entropy, J. R. Stat. Soc. Ser. B (Methodological) **38**, 54 (1976).
[46] A. Solonen, H. Haario, and M. Laine, Simulation-based optimal design using a response variance criterion, J. Comput. Graph. Stat. **21**, 234 (2012).
[47] C. C. Drovandi and M.-N. Tran, Improving the efficiency of fully Bayesian optimal design of experiments using randomised quasi–Monte Carlo, Bayesian Anal. **13**, 00 (2018).
[48] J. Beck, B. M. Dia, L. F. R. Espath, and R. Tempone, Multilevel double loop Monte Carlo and stochastic collocation methods with importance sampling for Bayesian optimal experimental design, Int. J. Numer. Methods Eng. **121**, 3402 (2020).
[49] K. J. Ryan, Estimating expected information gains for experimental designs with application to the random fatigue-limit model, J. Comput. Graph. Stat. **12**, 585 (2003).
[50] The timing calculations are performed on a quad-core x64-based CPU operating at 1.9 GHz with 256 kB L1 cache, 1 MB L2 cache, and 6 MB L3 cache and 8 GB memory available. Selected PYTHON functions are precompiled using NUMBA.
[51] C. L. Degen, F. Reinhard, and P. Cappellaro, Quantum sensing, Rev. Mod. Phys. **89**, 035002 (2017).
[52] R. D. McMichael, Scripts, data and plotting for simplified algorithms for adaptive experiment design in parameter estimation (2022), https://doi.org/10.18434/mds2-2585.
[53] N. Gordon, D. Salmond, and A. Smith, Novel approach to nonlinear/non-Gaussian Bayesian state estimation, IEE Proc. F Radar Signal Process. **140**, 107 (1993).
[54] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking, IEEE Trans. Signal Process. **50**, 174 (2002).
[55] J. Carpenter, P. Clifford, and P. Fearnhead, Improved particle filter for nonlinear problems, IEE Proc.—Radar, Sonar Navig. **146**, 2 (1999).
[56] J. Elfring, E. Torta, and R. van de Molengraft, Particle filters: A hands-on tutorial, Sensors **21**, 438 (2021).