Fast Bayesian Inference with Batch Bayesian Quadrature via Kernel Recombination

Anonymous Author(s)
Affiliation
Address
email

Abstract

Calculation of Bayesian posteriors and model evidences typically requires numerical integration. Bayesian quadrature (BQ), a surrogate-model-based approach to numerical integration, is capable of superb sample efficiency, but its lack of parallelisation has hindered its practical applications. In this work, we propose a parallelised (batch) BQ method, employing techniques from kernel quadrature, that possesses a provably-exponential convergence rate. Additionally, just as with Nested Sampling, our method permits simultaneous inference of both posteriors and model evidence. Samples from our BQ surrogate model are re-selected to give a sparse set of samples, via a kernel recombination algorithm, requiring negligible additional time to increase the batch size. Empirically, we find that our approach significantly outperforms the sampling efficiency of both state-of-the-art BQ techniques and Nested Sampling in various real-world datasets, including lithium-ion battery analytics.

1 Introduction

Many applications in science, engineering, and economics involve complex simulations to explain the structure and dynamics of the process. Such models are derived from knowledge of the mechanisms and principles underlying the data-generating process, and are critical for scientific hypothesis-building and testing. However, dozens of plausible simulators describing the same phenomena often exist, owing to differing assumptions or levels of approximation. Similar situations can be found in selection of simulator-based control models, selection of machine learning models on large-scale datasets, and in many data assimilation applications [23].

In such settings, with multiple competing models, choosing the best model for the dataset is crucial. Bayesian model evidence gives a clear criteria for such model selection. However, computing model evidence requires integration over the likelihood, which is challenging, particularly when the likelihood is non-closed-form and/or expensive. The ascertained model is often applied to produce posteriors for prediction and parameter estimation afterwards. There are many algorithms specialised for the calculation of model evidences or posteriors, although only a limited number of Bayesian inference solvers estimate both model evidence *and* posteriors in one go. As such, costly computations are often repeated (at least) twice. Addressing this concern, nested sampling (NS) [59, 38] was developed to estimate both model evidence and posteriors simultaneously, and has been broadly applied, especially amongst astrophysicists for cosmological model selection [50]. However, NS is based on a Monte Carlo (MC) sampler, and its slow convergence rate is a practical hindrance.

To aid NS, and other approaches, parallel computing is widely applied to improve the speed of wall-clock computation. Modern computer clusters and graphical processing units enable scientists to query the likelihood in large batches. However, parallelisation can, at best, linearly accelerate NS, doing little to counter NS's inherently slow convergence rate as a MC sampler.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

This paper investigates batch Bayesian quadrature (BQ) [52] for fast Bayesian inference. BQ solves the integral as an inference problem, modelling the likelihood function with a probabilistic model (typically a Gaussian process (GP)). Choosing an appropriate kernel and prior makes the integrals analytical. [30] proposed Warped sequential active Bayesian integration (WSABI), which adopts active learning to select samples upon uncertainty over the integrand. WSABI showed that BQ with expensive GP calculations could achieve faster convergence in wall time than cheap MC samplers. [65] introduced batch WSABI, achieving even faster calculation via parallel computing and became the fastest BQ model to date. We improve upon these existing works for a large-scale batch case.

2 Background

Vanilla Bayesian quadrature The objective of BQ is to compute the following integrals and obtain surrogate function of posterior:

$$p(x) = \frac{\ell(x)\pi(x)}{Z} = \frac{\ell(x)\pi(x)}{\int \ell(x)\pi(x)\,\mathrm{d}x}, \tag{1}$$

where both $\ell(x)$ (e.g. a likelihood) and $\pi(x)$ (e.g. a prior) are non-negative, and $x \in \mathbb{R}^d$ is a sample, and is sampled from prior $x \sim \pi(x)$. BQ solves the above integral as an inference problem, modelling a likelihood function by a GP. The likelihood function $\ell(x)$ is modelled:

$$\ell \,|\, \mathbf{y} \sim \mathcal{GP}(\ell; m_{\mathbf{y}}, C_{\mathbf{y}}), \tag{2a}$$

$$m_{\mathbf{y}}(x) = K(x, \mathbf{X})K(\mathbf{X}, \mathbf{X})^{-1}\mathbf{y}, \tag{2b}$$

$$C_{\mathbf{y}}(x, x') = K(x, x') - K(x, \mathbf{X})K(\mathbf{X}, \mathbf{X})^{-1}K(\mathbf{X}, x'), \tag{2c}$$

where $\mathbf{X} \in \mathbb{R}^{n\times d}$ is the matrix of observed samples, $\mathbf{y} \in \mathbb{R}^n$ is the observed likelihood values, K is the kernel. Throughout the paper, we refer to a symmetric positive semi-definite kernel just as a kernel. Due to linearity, the mean and variance of the integrals are simply

$$\mathbb{E}[Z \,|\, \mathbf{y}] = \int m_{\mathbf{y}}(x)\pi(x)\,\mathrm{d}x, \tag{3a}$$

$$\mathrm{var}[Z \,|\, \mathbf{y}] = \iint C_{\mathbf{y}}(x, x')\pi(x)\pi(x')\,\mathrm{d}x\,\mathrm{d}x'. \tag{3b}$$

In particular, the integral becomes analytic when $\pi(x)$ is Gaussian and K is squared exponential kernel, $K(\mathbf{X}, x) = v\sqrt{|2\pi\mathbf{W}|}\mathcal{N}(\mathbf{X}; x, \mathbf{W})$, where v is kernel variance and $\mathbf{W}$ is the diagonal covariance matrix whose diagonal elements are the lengthscales of each dimension. Since both the mean and variance of the integrals can be calculated analytically, posterior and model evidence can be obtained simultaneously (see Supplementary). Note that non-Gaussian prior is also possible to be selected via importance sampling. Still, we use this combination throughout this paper for simplicity.

Warped Bayesian quadrature (WSABI) WSABI with linearisation approximation (WSABI-L) sets the square-root GP for non-negativity with linearisation approximation of the transform $\tilde{\ell} \mapsto \ell = \alpha + \frac{1}{2}\tilde{\ell}^2$, where $\alpha = 0.8 \times \min(\mathbf{y})$. The square-root GP is defined as $\tilde{\ell} \sim \mathcal{GP}(\tilde{\ell}; \tilde{m}_{\mathbf{y}}, \tilde{C}_{\mathbf{y}})$, and we have the following linear approximation:

$$\ell \,|\, \mathbf{y} \sim \mathcal{GP}(\ell; m^L_{\mathbf{y}}, C^L_{\mathbf{y}}), \tag{4a}$$

$$m^L_{\mathbf{y}}(x) = \alpha + \frac{1}{2}\tilde{m}_{\mathbf{y}}(x)^2, \tag{4b}$$

$$C^L_{\mathbf{y}}(x, x') = \tilde{m}_{\mathbf{y}}(x)\tilde{C}_{\mathbf{y}}(x, x')\tilde{m}_{\mathbf{y}}(x'). \tag{4c}$$

Gaussianity implies the model evidence Z and posterior $p(x)$ remain analytical (see Supplementary).

Kernel quadrature in general kernel quadrature (KQ) is the group of numerical integration rules for calculating the integral of function classes that form the reproducing kernel Hilbert space (RKHS). With a KQ rule $Q_{\mathbf{w},\mathbf{x}}$ given by weights $\mathbf{w} = (w_i)_{i=1}^n$ and points $\mathbf{x} = (x_i)_{i=1}^n$, we approximate the integral by the weighted sum

$$Q_{\mathbf{w},\mathbf{x}}(h) := \sum_{i=1}^{n} w_i h(x_i) \approx \int h(x)\pi(x)\,\mathrm{d}x, \tag{5}$$

where h is a function of RKHS $\mathcal{H}$ associated with the kernel K. We define its worst-case error by $\mathrm{wce}(Q_{\mathbf{w},\mathbf{x}}) := \sup_{\|h\|_{\mathcal{H}}\leq 1} |Q_{\mathbf{w},\mathbf{x}}(h) - \int h(x)\pi(x)\,\mathrm{d}x|$. Surprisingly, it is shown in [40] that we have

$$\mathrm{var}[Z \mid \mathbf{y}] = \inf_{\mathbf{w}} \mathrm{wce}(Q_{\mathbf{w},\mathbf{x}})^2, \tag{6}$$

with $\mathbf{X} = \mathbf{x}$ in the previous section. Thus the point configuration in KQ with a small worst-case error gives a good way to select points to reduce the integral variance in Bayesian quadrature.

Random Convex Hull Quadrature (RCHQ) Recall from (5) and (6) that we wish to approximate the integral of a function h in the current RKHS. First, we prepare $n-1$ test functions $\varphi_1, \ldots, \varphi_{n-1}$ based on M sample points using the Nyström approximation of the kernel: $\varphi_i(x) := u_i^\top K(\mathbf{x}_{\text{nys}}, x)$, where $u_i \in \mathbb{R}^M$ is the i-th eigenvector of $K(\mathbf{x}_{\text{nys}}, \mathbf{x}_{\text{nys}})$. If we let λ_i be the i-th eigenvalue of the same matrix, the following gives a practical approximation [45]:

$$K_0(x, y) := \sum_{i=1}^{n-1} \lambda_i^{-1} \varphi_i(x)\varphi_i(y). \tag{7}$$

Next, we consider extracting a weighted set of n points $(\mathbf{w}_{\text{quad}}, \mathbf{x}_{\text{quad}})$ from a set of N points $\mathbf{x}_{\text{rec}}$ with positive weights $\mathbf{w}_{\text{rec}}$. We do it by the so-called kernel recombination algorithm [46, 62], so that the measure induced by $(\mathbf{w}_{\text{quad}}, \mathbf{x}_{\text{quad}})$ exactly integrates the above test functions $\varphi_1, \ldots, \varphi_{n-1}$ with respect to the measure given by $(\mathbf{w}_{\text{rec}}, \mathbf{x}_{\text{rec}})$ [37].

In the actual implementation of multidimensional case, we execute the kernel recombination not by the algorithm [62] with the best known computational complexity $\mathcal{O}(C_\varphi N + n^3 \log(N/n))$ (where C_φ is the cost of evaluating $(\varphi_i)_{i=1}^{n-1}$ at a point), but the one of [46] using an LP solver (Gurobi [31] for this time) with empirically faster computational time. We also adopt the randomized SVD [33] for the Nyström approximation, so we have a computational time empirically faster than $\mathcal{O}(NM + M^2 \log n + Mn^2 \log(N/n))$ [37] in practice.

3 Related works

Bayesian inference for intractable likelihood Inference with intractable likelihoods is a long-standing problem, and a plethora of methods have been proposed. Most infer posterior and evidence separately, and hence are not our fair competitors, as solving both is more challenging. For posterior inference, *Markov Chain Monte Carlo* [49, 36], particularly *Hamilton Monte Carlo* [39], is the gold standard. In a *Likelihood-free inference* context, kernel density estimation (KDE) with Bayesian optimisation [32] and neural networks [29] surrogates are proposed for simulation-based inference [21]. In a *Bayesian coreset* context, scalable Bayesian inference [47], sparse variational inference [15, 16], active learning [55] have been proposed for large-scale dataset inference. However, all of the above only calculate posteriors, not model evidences. For evidence inference, *Annealed Importance Sampling* [51], and *Bridge Sampling* [6], Sequential Monte Carlo (SMC) [22] are popular, but *only* estimate evidence, not the posterior.

Bayesian quadrature The early works on BQ, which directly replaced the likelihood function with a GP [52, 53, 56], did not explicitly handle non-negative integrand constraints. [54] introduced logarithmic warped GP to handle the non-negativity of the integrand, and introduced active learning for BQ, a method that selects samples based on where the variance of the integral will be minimised. [30] introduced square-root GP to make the integrand function closed-form and to speed up the computation. Furthermore, they accelerated active learning by changing the objective from the variance of the integral var[Z | **y**] to simply the variance of the integrand $\mathrm{var}[\ell(x)\pi(x)]$. [65] introduced first

batch BQ. [18] generalised warped GP for BQ, and proposed probit-transformation. BQ has been extended to machine learning tasks (model selection [17], manifold learning [25], kernel learning [34]) with new acquisition function (AF)s designed for each purpose. This paper employs WSABI-L and its AF for simplicity, but our approach might be extended to incorporate the above advances.

Kernel quadrature There are a number of KQ algorithms from herding/optimisation [19, 4, 40] to random sampling [3, 5]. In the context of BQ, Frank-Wolfe Bayesian quadrature (FWBQ) [9] using kernel herding has been proposed. This method proposes to do BQ with the points given by herding, but the guaranteed exponential convergence $\mathcal{O}(\exp(-cn))$ is limited to finite-dimensional RKHS, which is not the case in our setting. For general kernels, the convergence rate drops to $\mathcal{O}(1/\sqrt{n})$ [4]. Recently, a random sampling method with a good convergence rate was proposed for infinite-dimensional RKHS [37]. Based on Carathéodory's theorem for the convex hull, it efficiently calculates a reduced measure for a larger empirical measure. In this paper, we call it *random convex hull quadrature* (RCHQ) and use it in combination with the BQ methods.

4 Proposed Method: BASQ

4.1 General idea

We now introduce our algorithm, named *Bayesian alternately subsampled quadrature* (BASQ).

Kernel recombination for batch selection Batch WSABI [65] selects batch samples based on the AF, taking samples having the maximum AF values greedily via gradient-based optimisation with multiple starting points. The computational cost of this sampling scheme increases exponentially with the batch size and/or the problem dimension. Moreover, this method is often only locally optimal [66]. We adopt a scalable, gradient-free optimisation via KQ algorithm based on kernel recombination [37]. Surprisingly, [40] pointed out the equivalence between BQ and KQ with optimised weights. Kernel recombination can select n samples from the N candidate samples to efficiently reduce the worst-case error ($n \ll N$). The problem in batch BQ is selecting n samples from the probability measure $f(x)$ that minimise integral variance. When subsampling $\mathbf{x}_{\text{quad}} \sim f(x)$, these are the same. [37] proposed an efficient algorithm based on kernel recombination, RCHQ, which automatically returns a sparse set of n samples based on Carathéodory's theorem. The computational cost of batch size n for our algorithm, BASQ, is lower than $\mathcal{O}(NM + M^2 \log n + Mn^2 \log(N/n))$ [37].

Alternately subsampling The performance of RCHQ relies upon the quality of a predefined kernel. Thus, we add BQ elements to KQ in return; making RCHQ an online algorithm. Throughout the sequential batch update, we pass the BQ-updated kernels to RCHQ.[1] This enables RCHQ to exploit the function shape information to select the best batch samples for the integral of the GP-modelled likelihood function $\ell(x)$. This corresponds to the batch maximisation of the model evidence for $\ell(x)$. Then, BQ optimises the hyperparameters based on samples from true likelihood $\ell_{\text{true}}(x)$, which corresponds to an optimised kernel preset for RCHQ in the next round. These alternate updates characterise our algorithm, BASQ.

Importance sampling for uncertainty We added one more BQ element to RCHQ; *uncertainty sampling*. RCHQ relies upon the quality of subsamples from the prior distribution. However, sharp, multimodal, or high-dimensional likelihoods make it challenging to find prior subsamples that overlap over the likelihood distribution. This typically happens in Bayesian inference when the likelihood is expressed as the product of Gaussians, which gives rise to a very sharp likelihood peak. The likelihood of big data tends to become multimodal [57]. Therefore, we adopt *importance sampling*, gravitating subsamples toward the meaningful region, and correct the bias via its weights. We propose a mixture of prior and an AF as a guiding *proposal distribution*. The prior encourages global (rather than just local) optimisation, and the AF encourages sampling from uncertain areas for faster convergence. However, sampling from AF is expensive. We derive an efficient sparse Gaussian mixture sampler. Moreover, introducing square-root warping [30] enables the sampling distribution to factorise, yielding faster sampling.

[1] For updating kernel, the kernel to be updated is $C_{\mathbf{y}}$, not the kernel K. The kernel K just corresponds to the *prior* belief in the distribution of ℓ, so once we have observed the samples $\mathbf{X}$ (and $\mathbf{y}$), the variance to be minimised becomes $C_{\mathbf{y}}$.

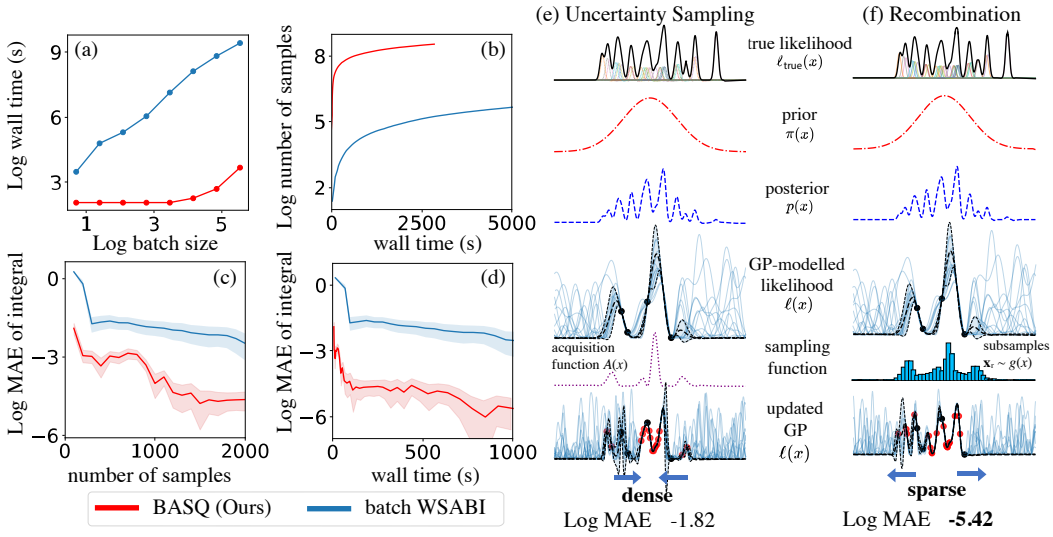

Figure 1: Performance comparison of our algorithm against batch WSABI [65]. All evaluation was performed with the likelihood of a mixture of N-dimensional Gaussians. (a), (b), (c), (d) 10-dimensional Gaussians, (e), (f) univariate Gaussians.

Summary of contribution We summarised the key differences between batch WSABI and BASQ in Figure 1.[2] (a) shows that BASQ is more scalable in batch size. (b) clarifies that BASQ can sample 10 to 100 times as many samples in the same time budget as WSABI, supported by the efficient sampler and RCHQ. (c) states the convergence rate of BASQ is faster than WSABI, regardless of computation time. (d) demonstrates the combined acceleration in wall time. While the batch WSABI reached 10^{-1} after 1,000 seconds passed, BASQ was within seconds. (e) and (f), visualised how RCHQ selects sparse samples than batch WSABI. This clearly explains that gradient-free kernel recombination is better in finding the global optimal than multi-start optimisation. These results demonstrate that we were able to combine the merits of BQ and KQ. We further tested with various synthetic datasets and real-world tasks in the fields of lithium-ion batteries and material science. Moreover, we mathematically analyse the convergence rate with proof in Section 6.

4.2 Algorithm

Table 1 illustrates the pseudo-code for our algorithm. Rows 4 - 10 correspond to RCHQ, and rows 11 - 15 correspond to BQ. We can use the variance of the integral $\mathrm{var}[Z|\mathbf{y}]$ as a convergence criterion. For hyperparameter optimisation, we adopt the type-II maximum likelihood estimation (MLE) to optimise hyperparameters via L-BFGS [14] for speed.

Importance sampling for uncertainty We define the distribution of interest as $f(x) = \pi(x)$, prior, and the proposal distribution $g(x)$ is a mixture of prior $\pi(x)$ and acquisition function $A(x)$:

$$g(x) = (1-r)f(x) + r\tilde{A}(x), \quad 0 \leq r \leq 1 \tag{8}$$

$$\tilde{A}(x) = \frac{C^L_{\mathbf{y}}(x,x)\pi(x)}{\int C^L_{\mathbf{y}}(x,x)\pi(x)\mathrm{d}x}, \tag{9}$$

$$\mathrm{w}_{\mathrm{IS}}(x) = f(x)/g(x), \tag{10}$$

where r is the partition ratio of the proposal distribution, w_{IS} is the weight of the importance sampling. To ensure global optimisation, we mix the distributions f and g with the partition ratio r. Larger r encourages uncertainty sampling to explore other peaks, whereas $r = 0$ is the vanilla MC sampler.

[2]We randomly varied the number of components between 10 and 15, setting their variance uniformly at random between 1 and 4, and setting their means uniformly at random within the box bounded by [-3,3] in all dimensions. The weights of Gaussians were randomly generated from a uniform distribution, but set to be one after integration. mean absolute error (MAE) was adopted for the evaluation of integral estimation.

Table 1: BASQ algorithm

Algorithm 1: Bayesian Alternately Subsampled Quadrature (BASQ)

Notation: x_{init}: initial guess, k: a convergence criterion,
n: batch size, M: Nyström sample size, N: recombination sample size,
ℓ: GP-modelled likelihood, ℓ_{true}: true likelihood,
$\pi(x)$: prior, $p(x)$: posterior, $A(x)$: AF, K: kernel,
S_{nys}, S_{rec}: samplers for Nyström and recombination, respectively

Input: prior $\pi(x)$, true likelihood function ℓ_{true}
Output: posterior $p(x)$, the mean and variance of model evidence $\mathbb{E}[Z \mid \mathbf{y}], \text{var}[Z \mid \mathbf{y}]$

$C_{\mathbf{y}}^L, A(x) \leftarrow$ InitialiseGPs(x_{init}) # Initialise GPs with initial guess
$S_{\text{nys}}, S_{\text{rec}} \leftarrow$ SetSampler$(A(x), \pi(x))$ # Set samplers
while $\text{var}[Z \mid \mathbf{y}] > k$:
  $\mathbf{x}_{\text{nys}} \sim S_{\text{nys}}(M)$ # Samples M points for test function φ
  $(\mathbf{w}_{\text{rec}}, \mathbf{x}_{\text{rec}}) \sim S_{\text{rec}}(N)$ # Samples N points for recombination
  $\varphi_1, ..., \varphi_{n-1} \leftarrow$ Nyström$(\mathbf{x}_{\text{nys}}, C_{\mathbf{y}}^L)$ # Define $n-1$ test functions
  solve a kernel recombination problem
    Find $\mathbf{w}_{\text{quad}} \geq \mathbf{0}$ and $\mathbf{x}_{\text{quad}} \subset \mathbf{x}_{\text{rec}}$
    s.t. $\mathbf{w}_{\text{quad}}^\top \varphi_i(\mathbf{x}_{\text{quad}}) = \mathbf{w}_{\text{rec}}^\top \varphi_i(\mathbf{x}_{\text{rec}})$, $\mathbf{w}_{\text{quad}}^\top \mathbf{1} = \mathbf{w}_{\text{rec}}^\top \mathbf{1}$
    return $\mathbf{x}_{\text{quad}}$ # The sparse set of n samples
  $\mathbf{y} = \text{Parallel}(\ell_{\text{true}}(\mathbf{x}_{\text{quad}}))$ # Parallel computing of likelihood
  $K \leftarrow$ Update$(\mathbf{x}_{\text{quad}}, \mathbf{y})$ # Train GPs
  $C_{\mathbf{y}}^L, A(x) \leftarrow$ OptHypersThenUpdate(K) # Type II MLE optimisation
  $S_{\text{nys}}, S_{\text{rec}} \leftarrow$ ResetSampler$(A(x), \pi(x))$ # Reset samplers with the updated $A(x)$
  $\mathbb{E}[Z \mid \mathbf{y}], \text{var}[Z \mid \mathbf{y}] \leftarrow$ BayesQuad$(m_{\mathbf{y}}^L, C_{\mathbf{y}}^L, \pi(x))$ # Calculate via Eqs. (3) and (4)
return $p(x), \mathbb{E}[Z \mid \mathbf{y}], \text{var}[Z \mid \mathbf{y}]$

Efficient sampler Sampling from $A(x)$, a mixture of Gaussians, is expensive, as some mixture weights are negative, preventing the usual practice of weighted sampling from each Gaussian. As the unwarped kernel $C_{\mathbf{y}}^L(x, x)$ is also computationally expensive, we adopt a *factorisation trick*:

$$Z = \int \ell(x)\pi(x)\,\mathrm{d}x = \alpha + \frac{1}{2}\int |\tilde{\ell}(x)|^2 \pi(x)\,\mathrm{d}x \approx \alpha + \frac{1}{2}\int |\tilde{\ell}(x)| f(x)\,\mathrm{d}x, \tag{11}$$

where we have changed the distribution of interest to $f(x) = |\tilde{m}_{\mathbf{y}}(x)|\pi(x)$. This is doubly beneficial. Firstly, the distribution of interest $f(x)$ will be updated over iterations. The previous $f(x) = \pi(x)$ means the subsample distribution eventually obeys prior, which is disadvantageous if the prior does not overlap the likelihood sufficiently. On the contrary, the new $f(x)$ narrows its region via $|\tilde{m}_{\mathbf{y}}(x)|$. Secondly, the likelihood function changed to $|\tilde{\ell}(x)|$, thus the kernels shared with RCHQ changed into cheap warped kernel $\tilde{C}_{\mathbf{y}}(x, x)$. This reduces the computational cost of RCHQ, and the sampling cost of $A(x)$. Now $A(x) = \tilde{C}_{\mathbf{y}}(x)\pi(x)$, which is also a Gaussian mixture, but the number of components is significantly lower than the original AF (9). As $\tilde{C}_{\mathbf{y}}(x)$ is positive, the positive weights of the Gaussian mixture should cover the negative components. Interestingly, in many cases, the positive weights vary exponentially, which means that limited number of components dominate the functional shape. Thus, we can ignore the trivial components for sampling.[3] Then we adopt SMC [43] to sample $A(x)$. We have a highly-calibrated proposal distribution of sparse Gaussian mixture, leading to efficient resampling from real $A(x)$ (see Supplementary).

Variants of proposal distribution Although we can mathematically derive the efficacy of the above proposal distribution $g(x)$, sometimes we have prior information not included in the above procedure. To incorporate human inductive bias, we propose two additional "biased" proposal

[3]Negative elements in the matrices only exist in $K(\mathbf{X}, \mathbf{X})^{-1}$, which can be drawn from the memory of the GP regression model without additional calculation. The number of positive components is half of the matrix on average, resulting in $\mathcal{O}(n^2/2)$. Then, taking the threshold via the inverse of the recombination sample size N, the number of components becomes $n_{\text{comp}} \ll n^2$, resulting in sampling complexity $\mathcal{O}(n^2/2 + n_{\text{comp}}N)$.

distributions. The first case is where we know both the maximum likelihood points and the likelihood's unimodality. This is typical in Bayesian inference because we can obtain (sub-)maximum points via a maximum a posteriori probability (MAP) estimate. In this case, we know exploring around the perfect initial guess is optimal rather than unnecessarily exploring an uncertain region. Thus, we introduce the initial guess believer (IGB) proposal distribution, $g_{\text{IGB}}(x)$. This is written as $g_{\text{IGB}}(x) = (1-r)\pi(x) + r\sum_{i=1} w_{i,\text{IGB}}\mathcal{N}(x; X_i, \mathbf{W})$, where $w_{i,\text{IGB}} = \{0 \text{ if } y_i \leq 0, \text{ else } 1\}$, $X_i \in \mathbf{X}$. This means exploring only the vicinity of the observed data $\mathbf{X}$. The second case is where we know the likelihood is multimodal. In this case, determining all peak positions is most beneficial. Thus more explorative distribution is preferred. As such, we introduce the uncertainty believer (UB) proposal distribution, $g_{\text{UB}}(x)$. This is written as $g_{\text{UB}}(x) = A(x)$, meaning pure uncertainty sampling. To contrast the above two, we term the proposal distribution in Eq. (8) as integral variance reduction (IVR) $g_{\text{IVR}}(x)$.

5 Experiments

Given our new model BASQ, with three variants of the proposal distribution, IVR, IGB, and UB, we now test for speed against MC samplers and batch WSABI. We compared with MultiNest NS [59, 24] and Dynamic NS [38], and Reactive NS with MLFriends [10, 11], coded with dynesty [60] and UltraNest [13] library. According to the review [12], MLFriends is the state-of-the-art NS sampler to date. The code is implemented based on BayesQuad [64], GPy [28], NumPy [35] SciPy [63], Gurobi [31], and code around kernel recombination [20, 37] with additional modification. All experiments on synthetic datasets were averaged over 10 repeats, computed in parallel with multicore CPUs, without GPU for fair comparison.[4] The posterior distribution of NS was estimated via KDE with weighted samples [27]. For maximum speed performance, batch size was optimised for each method in each dataset, in fairness to the competitors. Batch WSABI needs to optimise batch size to balance the likelihood query cost and sampling cost, because sampling cost increases rapidly with batch size, as shown in Figure 1(a). Therefore, it has an optimal batch size for faster convergence. By wall time cost, we exclude the cost of integrand evaluation; that is, the wall time cost is the overhead cost of batch evaluation. Details can be found in the Supplementary.

5.1 Synthetic problems

We evaluate all methods on three synthetic problems. The goal is to estimate the integral and posterior of the likelihood modelled with the highly multimodal functions. Prior was set to a two-dimensional multivariate normal distribution, with a 0 mean vector, and covariance whose diagonal elements are 2. The optimised batch sizes for each methods are BASQ: 100, batch WSABI: 16. The synthetic likelihood functions are cheap (0.5 ms on average). This is advantageous setting for NS: Within 10 seconds, the batch WSABI, BASQ, and NS collected 32, 600, 23225 samples, respectively. As for the metrics, posterior estimation was tested with Kullback-Leibler (KL) upon random 10,000 samples from true posterior. Evidence was evaluated with MAE, and ground truth was derived analytically.

Likelihood functions *Branin-Hoo* [41] is 8 modal function in two-dimensional space. *Ackley* [61] is a highly multimodal function with point symmetric periodical peaks in two-dimensional space. *Oscillatory function* [26] is a highly multimodal function with reflection symmetric periodical peaks of highly-correlated ellipsoids in two-dimensional space.

5.2 Real-world dataset

We consider three real-world applications with different computation times. The first two are simulator-based likelihoods, and the third is a hierarchical GP for high-resolution time-series analysis. We cannot obtain ground-truth posterior and model evidence. Thus, we adopted the empirical metric. For the posterior, we can calculate the true conditional posterior distribution along the line passing through ground truth parameter points. Then, evaluate the posterior with root mean squared error (RMSE) against 50 test samples for each dimension. For integral, we compare the model evidence itself. With regard to the sampling speed, querying these likelihoods is expensive. Therefore, there is no significant difference between batch WSABI and BASQ in the sample size that can earn per

[4] Performed on MacBook Pro 2019, 2.4 GHz 8-Core Intel Core i9, 64 GB 2667 MHz DDR4

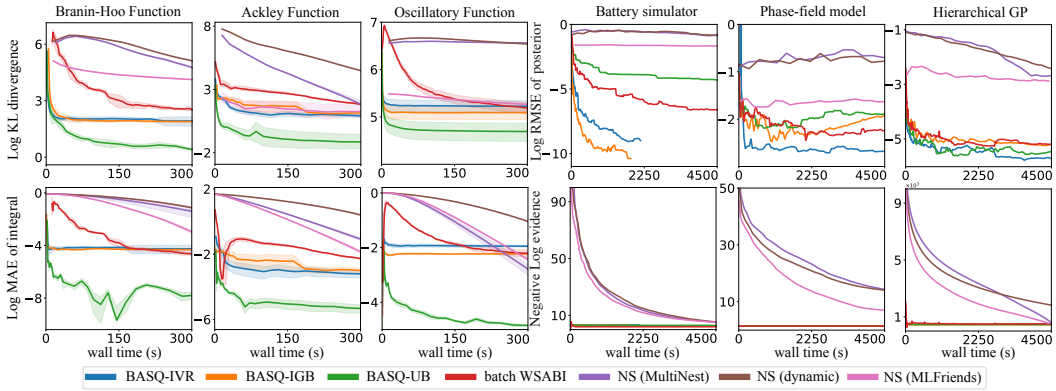

Figure 2: Time in seconds vs. KL divergence for posterior, and MAE for evidence in the synthetic datasets. Time in seconds vs. RMSE for posterior and evidence itself in the real-world dataset.

wall time. However, NS is based on rejection sampling. Thus, more than 50% of queried samples are dismissed on average, meaning the sample size of NS is roughly half of BASQ. Each simulator reduced its spatial and time resolution for simplicity. Still, the time becomes much longer when it increases for accuracy. Similarly, the number of data for the hierarchical GP model was reduced to approximately 40,000. Yet, the original data size is roughly 1,000 times. The batch sizes are BASQ: 32, batch WSABI: 8. For the details, see Supplementary.

Parameter estimation of the lithium-ion battery simulator The simulator is the SPMe [48], [5] estimating 3 simulation parameters at a given time-series voltage-current signal (the diffusivity of lithium-ion on the anode and cathode, and the experimental noise variance). Prior is modified to log multivariate normal distribution from [2]. Each query takes 1.2 seconds on average.

Parameter estimation of the phase-field model The simulator is the phase-field model [48], [6] estimating 4 simulation parameters at given time-series two-dimensional morphological image (temperature, interaction parameter, Bohr magneton coefficient, and gradient energy coefficient). Prior is a log multivariate normal distribution. Each query takes 7.4 seconds on average.

Hyperparameter marginalisation of hierarchical GP model The hierarchical GP model was designed for analysing the large-scale battery time-series dataset from solar off-grid system field data [1].[7] For fast estimation of parameters in each GP, the recursive technique [58] is adopted. The task is to marginalise 5 GP hyperparameters at given hyperprior, which is modified to log multivariate normal distribution from [1]. Each query takes 1.1 seconds on average.

5.3 Results

We find BASQ consistently delivers strong empirical performance, as shown in Figure 2. On all benchmark problems, BASQ-IVR, IGB, or UB outperform all baseline methods. Interestingly, the superiority is regardless of the computational cost of the likelihood. This suggests that BASQ could be a generally fast Bayesian solver as far as we investigated. In the multimodal setting of the synthetic dataset, explorative UB proposal distribution exceeds, whereas IVR and IGB work in a simulator-based likelihood setting. This is mainly because the parameter posteriors are not highly multimodal in the battery simulator case and the prior is well-tuned from [2]. When comparing each proposal distribution, only BASQ-IVR consistently outperforms WSABI. Our results support the general use of BASQ-IVR, or BASQ-UB if the likelihood is known to be highly multimodal.

[5] SPMe code used was translated into Python from MATLAB [7, 8]. This open-source code is published under the BSD 3-clause Licence. See more information on [7]

[6] PFM code used was from [44]. All rights of the code are reserved by the author of the book [44]. Thus, we do not redistribute the original code. Please purchase the book or contact the author if necessary.

[7] The code and data were shared from the author of [1]. All rights of the code and data are reserved by the authors [1]. Thus, we do not redistribute the original code. Please contact the authors if necessary.

6 Convergence analysis for BASQ

Note that the kernel K on $\mathbb{R}^d$ in this section refers to the given covariance kernel of GP at each step. We discuss the convergence of BASQ in one iteration. We consider the importance sampling: Let f be a probability density on $\mathbb{R}^d$ and g be another density such that $f = \lambda g$ with a nonnegative function λ. Let us call such a pair, (f, g), a density pair with weight λ.

We approximate the kernel K with $K_0 = \sum_{i=1}^{n-1} c_i \varphi_i(x)\varphi_i(y)$. In general, we can apply the kernel recombination algorithm [46, 62] with the weighted sample $(\mathbf{w}_{\text{rec}}, \mathbf{x}_{\text{rec}})$ to obtain a weighted point set $(\mathbf{w}_{\text{quad}}, \mathbf{x}_{\text{quad}})$ of size n satisfying $\mathbf{w}_{\text{quad}}^\top \varphi_i(\mathbf{x}_{\text{quad}}) = \mathbf{w}_{\text{rec}}^\top \varphi_i(\mathbf{x}_{\text{rec}})$ $(i = 1, \ldots, n-1)$ and $\mathbf{w}_{\text{quad}}^\top \mathbf{1} = \mathbf{w}_{\text{rec}}^\top \mathbf{1}$. By modifying the kernel recombination algorithm, we can require $\mathbf{w}_{\text{quad}}^\top k_1^{1/2}(\mathbf{x}_{\text{quad}}) \leq \mathbf{w}_{\text{rec}}^\top k_1^{1/2}(\mathbf{x}_{\text{rec}})$, where $k_1^{1/2}(x) := \sqrt{K(x,x) - K_0(x,x)}$ [37]. We call such $(\mathbf{w}_{\text{quad}}, \mathbf{x}_{\text{quad}})$ a *proper* kernel recombination of $(\mathbf{w}_{\text{rec}}, \mathbf{x}_{\text{rec}})$ with K_0.[8] We have the following guarantee (proved in Supplementary):

Theorem 1. *Suppose $\int \sqrt{K(x,x)} f(x)\,\mathrm{d}x < \infty$, $\ell \sim \mathcal{GP}(m, K)$, and we are given an $(n-1)$-dimensional kernel K_0 such that $K_1 := K - K_0$ is also a kernel. Let (f, g) be a density pair with weight λ. Let $\mathbf{x}_{\text{rec}}$ be an N-point independent sample from g and $\mathbf{w}_{\text{rec}} := \lambda(\mathbf{x}_{\text{rec}})$. Then, if $(\mathbf{w}_{\text{quad}}, \mathbf{x}_{\text{quad}})$ is a proper kernel recombination of $(\mathbf{w}_{\text{rec}}, \mathbf{x}_{\text{rec}})$ for K_0, it satisfies*

$$\mathbb{E}_{\mathbf{x}_{\text{rec}}}\left[\sqrt{\text{var}[Z_f \mid \mathbf{x}_{\text{quad}}]}\right] \leq 2\left(\int K_1(x,x) f(x)\,\mathrm{d}x\right)^{1/2} + \sqrt{\frac{C_{K,f,g}}{N}}, \tag{12}$$

where $Z_f := \int \ell(x) f(x)\,\mathrm{d}x$ and $C_{K,f,g} := \int K(x,x)\lambda(x) f(x)\,\mathrm{d}x - \iint K(x,y) f(x) f(y)\,\mathrm{d}x\,\mathrm{d}y$.

The above approximation has one source of randomness which stems from sampling N points $\mathbf{x}_{\text{rec}}$ from g. One can also apply this estimate with a random kernel and thereby introduce another source of randomness. In particular, when we use the Nyström approximation for K_0 (that ensures K_1 is a kernel [37]), then one can show that $\int K_1(x,x) f(x)\,\mathrm{d}x$ can be bounded by

$$\int K_1(x,x) f(x)\,\mathrm{d}x \leq n\sigma_n + \sum_{m=n+1}^{\infty} \sigma_m + \mathcal{O}_p\left(\frac{nK_{\max}}{\sqrt{M}}\right), \tag{13}$$

where σ_n is the n-th eigenvalue of the integral operator $L^2(f) \ni h \mapsto \int K(\cdot, y) h(y) f(y)\,\mathrm{d}x$, $K_{\max} := \sup_x K(x,x)$. However, note that unlike Eq. (12), this inequality only applies with high probability due to the randomness of K_0; see Supplementary for details.

If, for example, K is a Gaussian kernel on $\mathbb{R}^d$ and f is a Gaussian distribution, we have $\sigma_n = \mathcal{O}(\exp(-cn^{1/d}))$ for some constant $c > 0$ (see Supplementary). So in (12) we also achieve an exponential rate as well. It also works well with a moderate M in practise. Note that unlike the previous analysis [42], we do not have to assume that the space is compact.

7 Discussion

We introduced a batch BQ approach, BASQ, capable of simultaneous calculation of both model evidence and posteriors. BASQ demonstrated faster convergence (in wall-clock time) on both synthetic and real-world datasets, when compared against existing BQ approaches and state-of-the-art NS. Further, BASQ was proven to converge exponentially-fast under natural assumptions. Although it scales to the number of data seen in large-scale GP experiments, practical BASQ usage is limited to fewer than 16 dimensions (similar to many GP-based algorithms). This is because filling high dimensional space requires exponentially many samples. Employing a more efficient sampler, kernel learning, or importing ideas from coresets research may be productive lines of research. As for the mathematical proof, we do not incorporate batch and hyperparameter updates, which should be addressed in future work. The generality of our theoretical guarantee with respect to kernel and distribution should be useful for extending the analysis to the whole algorithm.

[8] Note that the inequality constraint on the diagonal value here is only needed for theoretical guarantee, and skipping it does not reduce the empirical performance [37].

References

[1] A. Aitio and D. A. Howey. Predicting battery end of life from solar off-grid system field data using machine learning. *Joule*, 5(12):3204–3220, 2021.

[2] A. Aitio, S. G. Marquis, P. Ascencio, and D. A. Howey. Bayesian parameter estimation applied to the Li-ion battery single particle model with electrolyte dynamics. *IFAC-PapersOnLine*, 53(2):12497–12504, 2020.

[3] F. Bach. On the equivalence between kernel quadrature rules and random feature expansions. *Journal of Machine Learning Research*, 18:714, 2017.

[4] F. Bach, S. Lacoste-Julien, and G. Obozinski. On the equivalence between herding and conditional gradient algorithms. In *International Conference on Machine Learning (ICML)*, 2012.

[5] A. Belhadji, R. Bardenet, and P. Chainais. Kernel quadrature with DPPs. In *International Conference on Neural Information Processing Systems (NeurIPS)*, 2019. URL: `https://proceedings.neurips.cc/paper/2019/file/7012ef0335aa2adbab58bd6d0702ba41-Paper.pdf`.

[6] C. H. Bennett. Efficient estimation of free energy differences from Monte Carlo data. *Journal of Computational Physics*, 22:245 – 268, 1976.

[7] A. M. Bizeray, J. Reniers, and D. A. Howey. Spectral Li-ion SPM. `"https://doi.org/10.5281/zenodo.212178"`, 2016.

[8] A. M. Bizeray, S. Zhao, S. R. Duncan, and D. A. Howey. Lithium-ion battery thermal-electrochemical model-based state estimation using orthogonal collocation and a modified extended Kalman filter. *Journal of Power Sources*, 296:400–412, 2015.

[9] F. X. Briol, C. J. Oates, M. Girolami, and M. A. Osborne. Frank-Wolfe Bayesian quadrature: Probabilistic integration with theoretical guarantees. In *International Conference on Neural Information Processing Systems (NeurIPS)*, 2015. URL: `https://proceedings.neurips.cc/paper/2015/file/ba3866600c3540f67c1e9575e213be0a-Paper.pdf`.

[10] J. Buchner. A statistical test for nested sampling algorithms. *Statistics and Computing*, 26(1):383–392, 2016.

[11] J. Buchner. Collaborative nested sampling: Big data versus complex physical models. *Publications of the Astronomical Society of the Pacific*, 131(1004):108005, 2019.

[12] J. Buchner. Nested sampling methods. *arXiv preprint arXiv:2101.09675*, 2021.

[13] J. Buchner. UltraNest–a robust, general purpose Bayesian inference engine. *arXiv preprint arXiv:2101.09604*, 2021.

[14] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5):1190–1208, 1995.

[15] T. Campbell and B. Beronov. Sparse variational inference: Bayesian coresets from scratch. *International Conference on Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[16] T. Campbell and T. Broderick. Automated scalable Bayesian inference via Hilbert coresets. *The Journal of Machine Learning Research*, 20(1):551–588, 2019.

[17] H. Chai, J. F. Ton, M. A. Osborne, and R. Garnett. Automated model selection with Bayesian quadrature. In *International Conference on Machine Learning (ICML)*, volume 97, pages 931–940, 2019. URL: `https://proceedings.mlr.press/v97/chai19a.html`.

[18] H. R. Chai and R. Garnett. Improving quadrature for constrained integrands. In *In IEEE 16th International Conference on Data Mining (ICDM)*, page 1107, 2016. URL: `https://doi.org/10.1109/ICDM.2016.0144`.

[19] Y. Chen, M. Welling, and A. Smola. Super-samples from kernel herding. In *International Conference on Uncertainty in Artificial Intelligence (UAI)*, 2010.

[20] F. Cosentino, H. Oberhauser, and A. Abate. A randomized algorithm to reduce the support of discrete measures. *International Conference on Neural Information Processing Systems (NeurIPS)*, 33:15100–15110, 2020.

[21] K. Cranmer, J. Brehmer, and G. Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.

[22] X. Didelot, R. G. Everitt, A. M. Johansen, and D. J. Lawson. Likelihood-free estimation of model evidence. *Bayesian analysis*, 6(1):49–76, 2011.

[23] G. Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, 99(C5):10143–10162, 1994.

[24] F. Feroz, M. P. Hobson, and M. Bridges. MultiNest: an efficient and robust Bayesian inference tool for cosmology and particle physics. *Monthly Notices of the Royal Astronomical Society*, 398(4):1601–1614, 2009.

[25] C. Fröhlich, A. Gessner, P. Hennig, B. Schölkopf, and G. Arvanitidis. Bayesian quadrature on Riemannian data manifolds. In *International Conference on Machine Learning (ICML)*, 2021. URL: `https://icml.cc/virtual/2021/poster/9655`.

[26] A. Genz. Testing multidimensional integration routines. In *In Proc. of international conference on Tools, methods and languages for scientific and engineering computation*, pages 81–94, 1984.

[27] F. J. G. Gisbert. Weighted samples, kernel density estimators and convergence. *Empirical Economics*, 28:335–351, 2003. URL: `https://doi.org/10.1007/s001810200134`.

[28] GPy. GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`, since 2012.

[29] D. Greenberg, M. Nonnenmacher, and J. Macke. Automatic posterior transformation for likelihood-free inference. In *International Conference on Machine Learning (ICML)*, pages 2404–2414, 2019.

[30] T. Gunter, M. A. Osborne, R. Garnett, P. Hennig, and S. J. Roberts. Sampling for inference in probabilistic models with fast Bayesian quadrature. In *International Conference on Neural Information Processing Systems (NeurIPS)*, 2014. URL: `https://proceedings.neurips.cc/paper/2014/file/e94f63f579e05cb49c05c2d050ead9c0-Paper.pdf`.

[31] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022. URL: `https://www.gurobi.com`.

[32] M. U. Gutmann, J. Corander, et al. Bayesian optimization for likelihood-free inference of simulator-based statistical models. *Journal of Machine Learning Research*, 2016.

[33] N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[34] S. Hamid, S. Schulze, M. A. Osborne, and S. J. Roberts. Marginalising over stationary kernels with Bayesian quadrature. 2021. URL: `https://proceedings.mlr.press/v151/hamid22a/hamid22a.pdf`.

[35] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. Kerkwijk, M. Brett, A. Haldane, J. F. Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020. `doi:10.1038/s41586-020-2649-2`.

[36] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. URL: `https://doi:10.1093/biomet/57.1.97`.

[37] S. Hayakawa, H. Oberhauser, and T. Lyons. Positively weighted kernel quadrature via subsampling. arXiv preprint, 2021. URL: `https://doi.org/10.48550/arXiv.2107.09597`.

[38] E. Higson, W. Handley, M. Hobson, and A. Lasenby. Dynamic nested sampling: an improved algorithm for parameter estimation and evidence calculation. *Stat. Comput.*, 29:891–913, 2019. URL: `https://doi.org/10.1007/s11222-018-9844-0`.

[39] M. D. Hoffman and A. Gelman. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

[40] F. Huszár and D. Duvenaud. Optimally-weighted herding is Bayesian quadrature. In *International Conference on Uncertainty in Artificial Intelligence (UAI)*, 2012.

[41] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21:345–383, 2001.

[42] M. Kanagawa and P. Hennig. Convergence guarantees for adaptive Bayesian quadrature methods. *Advances in Neural Information Processing Systems*, 32, 2019.

[43] G. Kitagawa. A Monte Carlo filtering and smoothing method for non-Gaussian nonlinear state space models. In *Proceedings of the 2nd U.S.-Japan Joint Seminar on Statistical Time Series Analysis*, page 110, 1993. URL: `https://www.ism.ac.jp/~kitagawa/1993_US-Japan.pdf`.

[44] T. Koyama. *Computational Engineering for Materials Design, Computational Microstructures, New and Expanded Edition: Microstructure Formation Analysis by the Phase-Field Method*. Uchida Rokakuho, Tokyo, Japan, 2019.

[45] S. Kumar, M. Mohri, and A. Talwalkar. Sampling methods for the Nyström method. *The Journal of Machine Learning Research*, 13(1):981–1006, 2012.

[46] C. Litterer and T. Lyons. High order recombination and an application to cubature on Wiener space. *The Annals of Applied Probability*, 22(4):1301–1327, 2012.

[47] D. Manousakas, Z. Xu, C. Mascolo, and T. Campbell. Bayesian pseudocoresets. *International Conference on Neural Information Processing Systems (NeurIPS)*, 33:14950–14960, 2020.

[48] S. G. Marquis, V. Sulzer, R. Timms, C. P. Please, and S. J. Chapman. An asymptotic derivation of a single particle model with electrolyte. *Journal of the Electrochemical Society*, 166(15):A3693–A3706, 2019. URL: `https://doi.org/10.1149/2.0341915jes`.

[49] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[50] P. Mukherjee, D. Parkinson, and A. R. Liddle. A nested sampling algorithm for cosmological model selection. *The Astrophysical Journal*, 638(2):L51, 2006.

[51] R.M. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

[52] A. O'Hagan. Bayes-Hermite quadrature. *Journal of Statistical Planning and Inference*, 29:245, 1991. URL: `https://doi.org/10.1016/0378-3758(91)90002-V`.

[53] A. O'Hagan. Bayesian quadrature with non-normal approximating functions. *Statistics and Computing*, 8:365, 1998. URL: `https://doi.org/10.1023/A:1008832824006`.

[54] M. A. Osborne, D. Duvenaud, R. Garnett, C. Rasmussen, S. Roberts, and Z. Ghahramani. Active learning of model evidence using Bayesian quadrature. In *International Conference on Neural Information Processing Systems (NeurIPS)*, 2012. URL: `https://proceedings.neurips.cc/paper/2012/file/6364d3f0f495b6ab9dcf8d3b5c6e0b01-Paper.pdf`.

[55] R. Pinsler, J. Gordon, E. Nalisnick, and J. M. Hernández-Lobato. Bayesian batch active learning as sparse subset approximation. *International Conference on Neural Information Processing Systems (NeurIPS)*, 32, 2019.

[56] C. E. Rasmussen and Z. Ghahramani. Bayesian Monte Carlo. In *International Conference on Neural Information Processing Systems (NeurIPS)*, 2003. URL: `https://mlg.eng.cam.ac.uk/zoubin/papers/RasGha03.pdf`.

[57] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain. Gaussian processes for time-series modelling. *Phil. Trans. R. Soc. A.*, 371:20110550, 2013.

[58] S. Sarkka and J. Hartikainen. Infinite-dimensional Kalman filtering approach to spatio-temporal Gaussian process regression. In *Artificial Intelligence and Statistics*, pages 993–1001, 2012.

[59] J. Skilling. Nested sampling for general Bayesian computation. *Bayesian Analysis*, 1(4):833 – 859, 2006. URL: `https://doi.org/10.1214/06-BA127`.

[60] J. S. Speagle. dynesty: a dynamic nested sampling package for estimating Bayesian posteriors and evidences. *Monthly Notices of the Royal Astronomical Society*, 493(3):3132–3158, 2020. URL: `https://doi.org/10.1093/mnras/staa278`.

[61] S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets. Retrieved May 12, 2022, from `http://www.sfu.ca/~ssurjano`.

[62] M. Tchernychova. *Carathéodory cubature measures*. PhD thesis, University of Oxford, 2015.

[63] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. `doi:10.1038/s41592-019-0686-2`.

[64] E. Wagstaff. Bayesian quadrature library. `https://github.com/OxfordML/bayesquad`, since 2019.

[65] E. Wagstaff, S. Hamid, and M. A. Osborne. Batch selection for parallelisation of Bayesian quadrature. arXiv preprint, 2018. URL: `https://doi.org/10.48550/arXiv.1812.01553`.

[66] J. Wilson, F. Hutter, and M. Deisenroth. Maximizing acquisition functions for Bayesian optimization. *International Conference on Neural Information Processing Systems (NeurIPS)*, 31, 2018.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See section 7
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] see section 6 and Supplementary
   (b) Did you include complete proofs of all theoretical results? [Yes] see section 6 and Supplementary
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] in the supplemental material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] see section 5 and Supplementary
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] see section 5
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] see section 5
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] see section 5
   (b) Did you mention the license of the assets? [Yes] see section 5
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] see section 5
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]