Automatic Bayesian Cubature manuscript No.
(will be inserted by the editor)

Fast Automatic Bayesian Cubature Using Sobol Sampling

R. Jagadeeswaran · Fred J. Hickernell

Received: date / Accepted: date

Abstract Automatic cubatures approximate integrals to user-specified error tolerances. For high dimensional problems, it is difficult to adaptively change the sampling pattern, but one can automatically determine the sample size, n, given a reasonable, fixed sampling pattern. We take this approach here using a Bayesian perspective. We postulate that the integrand is an instance of a Gaussian stochastic process parameterized by a constant mean and a covariance kernel defined by a scale parameter times a parameterized function specifying how the integrand values at two different points in the domain are related. These hyperparameters are inferred or integrated out using integrand values via one of three techniques: empirical Bayes, full Bayes, or generalized cross-validation. The sample size, n, is increased until the half-width of the credible interval for the Bayesian posterior mean is no greater than the error tolerance.

The process outlined above typically requires a computational cost of $O(N_{\text{opt}}n^3)$, where N_{opt} is the number of optimization steps required to identify the hyperparameters. Our innovation is to pair low discrepancy nodes with matching covariance kernels to lower the computational cost to $O(N_{\text{opt}}n \log n)$. This approach is demonstrated explicitly with rank-1 lattice sequences and shift-invariant kernels. Our algorithm is implemented in the Guaranteed Automatic Integration Library (GAIL).

Keywords Bayesian cubature · Fast automatic cubature · GAIL · Probabilistic numeric methods

R. Jagadeeswaran
Department of Applied Mathematics,
Illinois Institute of Technology
10 W. 32nd St., Room 208, Chicago IL 60616
E-mail: jrathin1@iit.edu

Fred J. Hickernell
Center for Interdisciplinary Scientific Computation and
Department of Applied Mathematics
Illinois Institute of Technology
10 W. 32nd St., Room 208, Chicago IL 60616
E-mail: hickernell@iit.edu

1 Introduction

Your text comes here. Separate text sections with
JR: discuss prior work

2 Bayesian Cubature

JR: Briefly explain and cite our BayesLattice paper

1. automatic Bayesian cubature
2. fast automatic Bayesian cubature

The Bayesian approach for numerical analysis was popularized by Diaconis Diaconis [1988]. The earliest reference for such kind of approach dates back to Poincaré, where, the theory of interpolation was discussed. Diaconis motivates the reader by interpreting the most well known numerical methods, 1) trapezoidal rule and 2) splines, from the statistical point of view with whatever is known about the integrand as prior information. For example, the trapezoidal rule can be interpreted as a Bayesian method with prior information being modeled as a Brownian motion in the sample space $\mathcal{C}[0,1)$, the space of continuous functions.

This research is focused on the Bayesian approach for numerical integration that is known as Bayesian cubature as introduced by O'Hagan OHagan [1991]. Bayesian cubature returns a probability distribution, that expresses belief about the true value of integral, $\mu(f)$. This posterior probability distribution is based on a prior that depends on f, which is computed via Bayes' rule using the *data* contained in the function evaluations Briol et al. [2019]. The distribution in general captures numerical uncertainty due to the fact that we have only used a finite number of function values to evaluate the integral.

2.1 Bayesian Posterior Error

We assume the integrand, f, is an instance of a stochastic Gaussian process, i.e., $f \sim \mathcal{GP}(m, s^2 C_{\boldsymbol{\theta}})$. Specifically, f is a real-valued random function with constant mean m and covariance function $s^2 C_{\boldsymbol{\theta}}$, where s is a positive scale factor, and $C_{\boldsymbol{\theta}} : [0,1]^d \times [0,1]^d \to \mathbb{R}$ is a symmetric, positive-definite function and, parameterized by $\boldsymbol{\theta}$:

$$\mathsf{C}^T = \mathsf{C}, \quad \boldsymbol{a}^T \mathsf{C} \boldsymbol{a} > 0, \quad \text{where } \mathsf{C} = (C_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j=1}^n, \\ \text{for all } \boldsymbol{a} \neq 0,\ n \in \mathbb{N},\ \text{distinct } \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in [0,1]^d. \quad (1)$$

The covariance function, C, and the Gram matrix, C, depend implicitly on $\boldsymbol{\theta}$, but the notation may omit this for simplicity's sake. Procedures for estimating or integrating out the hyperparameters m, s, and $\boldsymbol{\theta}$ are explained later in this section.

For a Gaussian process, all vectors of linear functionals of f have a multivariate Gaussian distribution. For any deterministic sampling scheme with distinct nodes, $\{\boldsymbol{x}_i\}_{i=1}^n$, and defining $\boldsymbol{f} := (f(\boldsymbol{x}_i))_{i=1}^n$ as the multivariate Gaussian vector of function values, it follows from the definition of a Gaussian process that

$$\boldsymbol{f} \sim \mathcal{N}(m\mathbf{1}, s^2 \mathsf{C}), \quad (2a)$$

$$\mu \sim \mathcal{N}(m, s^2 c_0), \quad (2b)$$

$$\text{where } c_0 := \int_{[0,1]^d \times [0,1]^d} C_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{t}) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{t}, \tag{2c}$$

$$\mathrm{cov}(\boldsymbol{f}, \mu) = \left(\int_{[0,1]^d} C(\boldsymbol{t}, \boldsymbol{x}_i) \, \mathrm{d}\boldsymbol{t} \right)_{i=1}^{n} =: \boldsymbol{c}. \tag{2d}$$

Here, c_0 and $\boldsymbol{c}$ depend implicitly on $\boldsymbol{\theta}$. We assume the covariance function C is simple enough that the integrals in these definitions can be computed analytically. We need the following lemma to derive the posterior error of our cubature.

Lemma 1 *[Rasmussen and Williams, 2006, (A.6), (A.11–13)] If* $\boldsymbol{Y} = (\boldsymbol{Y}_1, \boldsymbol{Y}_2)^T \sim \mathcal{N}(\boldsymbol{m}, \mathsf{C})$*, where* $\boldsymbol{Y}_1$ *and* $\boldsymbol{Y}_2$ *are random vectors of arbitrary length, and*

$$\boldsymbol{m} = \begin{pmatrix} \boldsymbol{m}_1 \\ \boldsymbol{m}_2 \end{pmatrix} = \begin{pmatrix} \mathbb{E}(\boldsymbol{Y}_1) \\ \mathbb{E}(\boldsymbol{Y}_2) \end{pmatrix},$$

$$\mathsf{C} = \begin{pmatrix} \mathsf{C}_{11} & \mathsf{C}_{21}^T \\ \mathsf{C}_{21} & \mathsf{C}_{22} \end{pmatrix} = \begin{pmatrix} \mathrm{var}(\boldsymbol{Y}_1) & \mathrm{cov}(\boldsymbol{Y}_1, \boldsymbol{Y}_2) \\ \mathrm{cov}(\boldsymbol{Y}_2, \boldsymbol{Y}_1) & \mathrm{var}(\boldsymbol{Y}_2) \end{pmatrix}$$

then

$$\boldsymbol{Y}_1 | \boldsymbol{Y}_2 \sim \mathcal{N}\big(\boldsymbol{m}_1 + \mathsf{C}_{21}^T \mathsf{C}_{22}^{-1} (\boldsymbol{Y}_2 - \boldsymbol{m}_2), \quad \mathsf{C}_{11} - \mathsf{C}_{21}^T \mathsf{C}_{22}^{-1} \mathsf{C}_{21}\big).$$

Moreover, the inverse of the matrix C *may be partitioned as*

$$\mathsf{C}^{-1} = \begin{pmatrix} \mathsf{A}_{11} & \mathsf{A}_{21}^T \\ \mathsf{A}_{21} & \mathsf{A}_{22} \end{pmatrix},$$

$$\mathsf{A}_{11} = (\mathsf{C}_{11} - \mathsf{C}_{12} \mathsf{C}_{22}^{-1} \mathsf{C}_{21})^{-1}, \qquad \mathsf{A}_{21} = -\mathsf{C}_{22}^{-1} \mathsf{C}_{21} \mathsf{A}_{11},$$

$$\mathsf{A}_{22} = \mathsf{C}_{22}^{-1} + \mathsf{C}_{22}^{-1} \mathsf{C}_{21} \mathsf{A}_{11} \mathsf{C}_{21}^T \mathsf{C}_{22}^{-1}.$$

It follows from Lemma 1 that the *conditional* distribution of the integral given observed function values, $\boldsymbol{f} = \boldsymbol{y}$ is also Gaussian:

$$\mu | (\boldsymbol{f} = \boldsymbol{y}) \sim \mathcal{N}\big(m(1 - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{1}) + \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{y}, \;\; s^2 (c_0 - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{c})\big). \tag{3}$$

The natural choice for the cubature is the posterior mean of the integral, namely,

$$\widehat{\mu} | (\boldsymbol{f} = \boldsymbol{y}) = m(1 - \boldsymbol{1}^T \mathsf{C}^{-1} \boldsymbol{c}) + \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{y}, \tag{4}$$

which takes the form of (**??**). Under this definition, the cubature error has zero mean and a variance depending on the choice of nodes:

$$(\mu - \widehat{\mu}) | (\boldsymbol{f} = \boldsymbol{y}) \sim \mathcal{N}\left(0, \quad s^2 (c_0 - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{c})\right).$$

A credible interval for the integral is given by

$$\mathbb{P}_f \left[|\mu - \widehat{\mu}| \le \mathrm{err}_{\mathrm{CI}} \right] = 99\%, \tag{5a}$$

$$\mathrm{err}_{\mathrm{CI}} = 2.58 s \sqrt{c_0 - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{c}}. \tag{5b}$$

Naturally, 2.58 and 99% can be replaced by other quantiles and credible levels.

3 Hyperparameter estimation

JR: This section explains hyperparameter estimation using gradient descent

The credible interval in (5) suggests how our automatic Bayesian cubature proceeds. Integrand data is accumulated until the width of the credible interval, $\mathrm{err}_{\mathrm{CI}}$, is no greater than the error tolerance. As n increases, one expects $c_0 - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{c}$ to decrease for well-chosen nodes, $\{\boldsymbol{x}_i\}_{i=1}^n$. Please note that the credible interval depends on the parameters m, s, and $\boldsymbol{\theta}$

Note that $\mathrm{err}_{\mathrm{CI}}$ has no explicit dependence on the integrand values, even though one would intuitively expect that a larger integrand should imply a larger $\mathrm{err}_{\mathrm{CI}}$.

This is because the hyperparameters, m, s, and $\boldsymbol{\theta}$, have not yet been inferred from integrand data. After inferring the hyperparameters, $\mathrm{err}_{\mathrm{CI}}$ *does reflect the size* of the integrand values. The following next few sections describe three approaches to hyperparameter estimation.

4 Empirical Bayes

The first and a very straight forward approach is to estimate the parameters via maximum likelihood estimation. The log-likelihood function of the parameters given the function data $\boldsymbol{y}$ is:

$$\begin{aligned} l(s, m, \boldsymbol{\theta}|\boldsymbol{y}) = &-\frac{1}{2} s^{-2} (\boldsymbol{y} - m\mathbf{1})^T \mathsf{C}^{-1} (\boldsymbol{y} - m\mathbf{1}) \\ &- \frac{1}{2} \log(\det \mathsf{C}) - \frac{n}{2} \log(s^2) + \text{constants}. \end{aligned}$$

Maximizing the log-likelihood first with respect to m, then with respect to s, and finally with respect to $\boldsymbol{\theta}$ yields

$$\begin{aligned} m_{\mathrm{EB}} &= \frac{\mathbf{1}^T \mathsf{C}^{-1} \boldsymbol{y}}{\mathbf{1}^T \mathsf{C}^{-1} \mathbf{1}}, \\ s^2_{\mathrm{EB}} &= \frac{1}{n} (\boldsymbol{y} - m_{\mathrm{EB}} \mathbf{1})^T \mathsf{C}^{-1} (\boldsymbol{y} - m_{\mathrm{EB}} \mathbf{1}) \\ &= \frac{1}{n} \boldsymbol{y}^T \left[\mathsf{C}^{-1} - \frac{\mathsf{C}^{-1} \mathbf{1} \mathbf{1}^T \mathsf{C}^{-1}}{\mathbf{1}^T \mathsf{C}^{-1} \mathbf{1}} \right] \boldsymbol{y}, \\ \boldsymbol{\theta}_{\mathrm{EB}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\{ \log \left(\boldsymbol{y}^T \left[\mathsf{C}^{-1} - \frac{\mathsf{C}^{-1} \mathbf{1} \mathbf{1}^T \mathsf{C}^{-1}}{\mathbf{1}^T \mathsf{C}^{-1} \mathbf{1}} \right] \boldsymbol{y} \right) + \frac{1}{n} \log(\det(\mathsf{C})) \right\}. \end{aligned}$$

The empirical Bayes estimate of $\boldsymbol{\theta}$ balances minimizing the covariance scale factor, s^2_{EB}, against minimizing $\det(\mathsf{C})$.

Under these estimates of the parameters, the cubature (4) and the credible interval (5) simplify to

$$\begin{aligned} \widehat{\mu}_{\mathrm{EB}} &:= \left(\frac{(1 - \mathbf{1}^T \mathsf{C}^{-1} \boldsymbol{c}) \mathbf{1}}{\mathbf{1}^T \mathsf{C}^{-1} \mathbf{1}} + \boldsymbol{c} \right)^T \mathsf{C}^{-1} \boldsymbol{y}, \\ \mathrm{err}^2_{\mathrm{EB}} &:= \frac{2.58^2}{n} \boldsymbol{y}^T \left[\mathsf{C}^{-1} - \frac{\mathsf{C}^{-1} \mathbf{1} \mathbf{1}^T \mathsf{C}^{-1}}{\mathbf{1}^T \mathsf{C}^{-1} \mathbf{1}} \right] \boldsymbol{y} (c_0 - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{c}), \\ \mathbb{P}_f \left[|\mu - \widehat{\mu}_{\mathrm{EB}}| \le \mathrm{err}_{\mathrm{EB}} \right] &= 99\%. \end{aligned} \tag{6}$$

Here c_0, $\boldsymbol{c}$, and C are assumed implicitly to be based on $\boldsymbol{\theta} = \boldsymbol{\theta}_{\mathrm{EB}}$.

4.1 Gradient descent to find optimal shape parameter

The equation specifying $\boldsymbol{\theta}_{\mathrm{EB}}$ as defined in (**??**) does not say how the parameter search can be done. There exist empirical algorithms Brent [1973], Forsythe et al. [1976] that one could use to accomplish the same. Since the objective function is known we could compute the gradient. Using the gradient of $l(s, m, \boldsymbol{\theta}|\boldsymbol{y})$, one can apply optimization techniques such as gradient descent to find the optimal value faster. Let us define the objective function for the same purpose by excluding the negative sign, which modifies the problem to become a minimization of

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y}) := \frac{1}{n} \log(\det \mathsf{C}) + \log \left((\boldsymbol{y} - m_{\mathrm{EB}} \mathbf{1})^T \mathsf{C}^{-1} (\boldsymbol{y} - m_{\mathrm{EB}} \mathbf{1}) \right) + \text{constants}.$$

Taking derivative with respect to θ_ℓ, for $\ell = 1, \cdots, d$

$$\frac{\partial}{\partial \theta_\ell}\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{1}{n}\frac{\partial}{\partial \theta_\ell}\log(\det\,\mathsf{C}) + \frac{\partial}{\partial \theta_\ell}\log\Big((\boldsymbol{y}-m_{\mathrm{EB}}\mathbf{1})^T\mathsf{C}^{-1}(\boldsymbol{y}-m_{\mathrm{EB}}\mathbf{1})\Big)$$

$$= \frac{1}{n}\mathrm{trace}\left(\mathsf{C}^{-1}\frac{\partial \mathsf{C}}{\partial \theta_\ell}\right) - \frac{\left((\boldsymbol{y}-m_{\mathrm{EB}}\mathbf{1})^T\mathsf{C}^{-1}\right)^T\left(\frac{\partial \mathsf{C}}{\partial \theta_\ell}\right)\left((\boldsymbol{y}-m_{\mathrm{EB}}\mathbf{1})^T\mathsf{C}^{-1}\right)}{(\boldsymbol{y}-m_{\mathrm{EB}}\mathbf{1})^T\mathsf{C}^{-1}(\boldsymbol{y}-m_{\mathrm{EB}}\mathbf{1})}$$

where we used some of the results from Dong et al. [2017]. This can be used with gradient descent as follows,

$$\theta_\ell^{(j+1)} = \theta_\ell^{(j)} - \nu_\ell \frac{\partial}{\partial \theta_\ell}\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y}), \quad j = 0, 1, \cdots \tag{7}$$

where ν_ℓ is the step size for the gradient descent.

5 Cone of Functions and the Credible interval

In this research we assume that the integrand belongs to a cone of well-behaved functions, $\mathscr{C}$, to make the computations bounded in terms of function data. The concept of cone in general for cubature error analysis can be stated using the error bound definition. Suppose that

$$|\mu(f) - \widehat{\mu}_n(f)| \le \mathrm{err}_{\mathrm{CI}}(f(\boldsymbol{x}_1), \cdots, f(\boldsymbol{x}_n)) \tag{8}$$

for some f, which it is 99% of the time under our hypothesis. Also note that our $\mathrm{err}_{\mathrm{CI}}$ (**??**) (**??**) are positively homogeneous functions, meaning,

$$\mathrm{err}_{\mathrm{CI}}(ay_1, \cdots, ay_n) = |a|\,\mathrm{err}_{\mathrm{CI}}(y_1, \cdots, y_n).$$

One can verify the homogeneity of (**??**) and (**??**) easily. Thus if f satisfies (8), then

$$\begin{aligned}|\mu(af) - \widehat{\mu}_n(af)| &= |a|\,|\mu(f) - \widehat{\mu}_n(f)| \\ &\le |a|\,\mathrm{err}_{\mathrm{CI}}(f(\boldsymbol{x}_1), \cdots, f(\boldsymbol{x}_n)) \\ &= \mathrm{err}_{\mathrm{CI}}(af(\boldsymbol{x}_1), \cdots, af(\boldsymbol{x}_n))\end{aligned}$$

for all real a. Thus the set of all f satisfying (8) is a *cone*, $\mathscr{C}$. Cones of functions satisfy the property that if $f \in \mathscr{C}$ then $af \in \mathscr{C}$.

In the context of Bayesian cubature, one can explain the cone concept beginning with the definition of credible interval (5). Let $f \sim \mathcal{GP}$, be an instance of a Gaussian stochastic process:

$$\mathbb{P}_f\left[|\mu(f) - \widehat{\mu}_n(f)| \le \mathrm{err}_{\mathrm{CI}}(f)\right] \ge 99\%.$$

This can be interpreted as $|\mu(f) - \widehat{\mu}_n(f)| \le \mathrm{err}_{\mathrm{CI}}(f)$ with 99% confidence. If f is in the 99% middle of the sample space with $f(\boldsymbol{x}_i) = y_i$ then af is also in the middle 99% of the sample space with $af(\boldsymbol{x}_i) = ay_i$.

We demonstrate the credible interval using the following example. For this purpose, choose a smooth and periodic integrand $f_{\mathrm{smooth}}(\boldsymbol{x}) = \exp(\sum_{\ell=1}^d \cos(2\pi x_\ell))$ and another integrand $f_{\mathrm{peaky}}(\boldsymbol{x}) = f_{\mathrm{smooth}} + a_{\mathrm{peaky}} f_{\mathrm{noise}}$ where $a_{\mathrm{peaky}} \in \mathbb{R}$. Here $f_{\mathrm{noise}}(\boldsymbol{x}) = (1 - \exp(2\pi\sqrt{-1}\boldsymbol{x}^T\boldsymbol{\zeta}))$, $\boldsymbol{\zeta} \in \mathbb{R}^d$ is some d-dimensional vector belonging to the dual space of the lattice nodes for some $\{\boldsymbol{x}_i\}_{i=1}^n$. The $\boldsymbol{\zeta}$ in the dual space of lattice nodes implies that $f_{\mathrm{noise}}(\boldsymbol{x}_i) = 0$ at the sampling nodes $\{\boldsymbol{x}_i\}_{i=1}^n$. The f_{nice} is obtained by kernel interpolation of the n samples of f_{smooth} at $\{\boldsymbol{x}_i\}_{i=1}^n$. We chose the Matérn kernel (**??**) for the interpolation. Please note that $f_{\mathrm{peaky}}(\boldsymbol{x}_i) = f_{\mathrm{nice}}(\boldsymbol{x}_i) = f_{\mathrm{smooth}}(\boldsymbol{x}_i)$ for $i = 1, \cdots, n$.

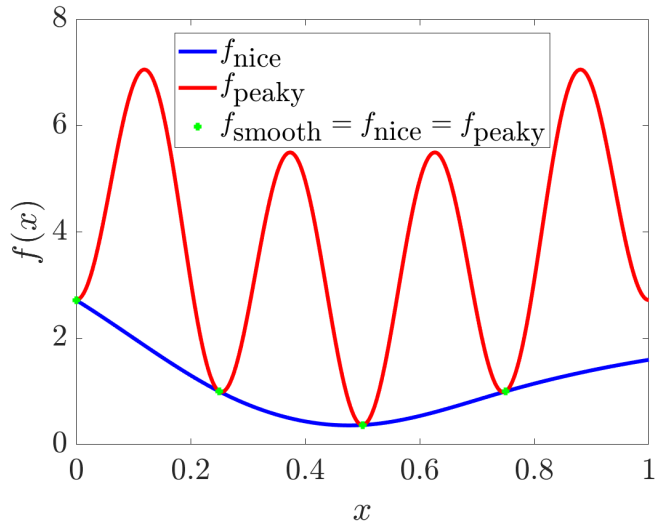

Fig. 1: Example integrands 1) f_{nice}, a smooth function, 2) f_{peaky}, a peaky function. The function values $f_{\text{peaky}}(\boldsymbol{x}_i) = f_{\text{nice}}(\boldsymbol{x}_i) = f_{\text{smooth}}(\boldsymbol{x}_i)$ for $i = 1, \cdots, n$. This plot can be conditionally reproduced using `DemoCone.m`

In Figure 1, the sampled function values are shown as dots. One can imagine these samples were obtained from f_{nice}, a moderately smoother function or from f_{peaky}, a highly oscillating function. In this example, we used $a_{\text{peaky}} = 2$.

When using $n = 16$ rank-1 lattice points, and $r = 1$ shift-invariant kernel, we get the posterior distribution of μ as shown in Figure 2. The true integral value is shown as μ_{smooth} which is at the center of the plot. The integral of the peaky function f_{peaky} lies outside of the 99% of the credible interval given by (6), whereas the μ_{nice} falls within.

Our Bayesian cubature algorithms compute the approximate integral using only the samples of the integrand. Estimated integral value of our algorithm closely matches the integral of a smooth function that falls within the middle of the confidence interval. If the true integrand were to resemble the smooth approximate function then the estimated integral will be accurate.

6 Sobol' Nets and Walsh Kernels

The previous section shows an automatic Bayesian cubature algorithm using rank-1 lattice nodes and shift-invariant kernels. In this chapter, we demonstrate a second approach to formulate fast Bayesian transform using matching kernel and point sets. Scrambled Sobol' nets and Walsh kernels are paired to achieve $\mathcal{O}(n^{-1+\epsilon})$ order error convergence where n is the sample size. Sobol' nets Sobol' [1967] are low discrepancy points, used extensively in numerical integration, simulation, and optimization. The results of this chapter can be summarized as a theorem,

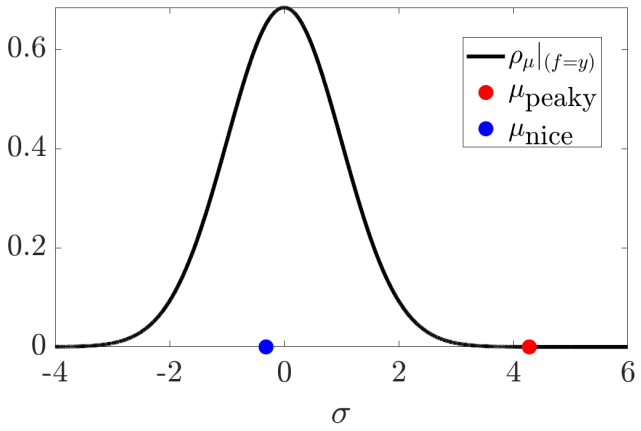

Fig. 2: Probability distributions showing the relative integral position of a smooth and a peaky function. f_{nice} lies within the center 99% of the confidence interval, and f_{peaky} lies on the outside of 99% of the confidence interval. This plot can be conditionally reproduced using `DemoCone.m`

Theorem 1 *Any symmetric, positive definite, digital shift-invariant covariance kernel of the form* (13) *scaled to satisfy* (**??**)*, when matched with digital net data-sites, satisfies assumptions* (**??**)*. The* fast Walsh-Hadamard transform *(FWHT) can be used to expedite the estimates of* $\boldsymbol{\theta}$ *in* (**??**) *and the credible interval widths* (**??**) *in* $\mathcal{O}(n \log n)$ *operations. The cubature,* $\widehat{\mu}$*, is just the sample mean.*

We introduce the necessary concepts and prove this theorem in the remaining of this chapter.

7 Sobol' Nets

Nets were developed to provide deterministic sample points for quasi-Monte Carlo rules Niederreiter [2005]. Nets are defined geometrically using elementary intervals, which are subintervals of the unit cube $[0,1)^d$. The (t,m,d)-nets in base b, introduced by Niederreiter, whose quality is governed by t. Lower values of t correspond to (t,m,d)-nets of higher quality Baldeaux [2010].

Definition 1 *Let* $\mathcal{A}$ *be the set of all elementary intervals* $\mathcal{A} \subset [0,1)^d$ *where* $\mathcal{A} = \prod_{\ell=1}^{d} [\alpha_\ell b^{-\gamma_\ell}, (\alpha_\ell + 1) b^{-\gamma_\ell})$*, with* $d, b, \gamma_\ell \in \mathbb{N}, b \geq 2$ *and* $b^{\gamma_\ell} > \alpha_\ell \geq 0$*. For* $m, t \in \mathbb{N}, m \geq t \geq 0$*, the point set* $\mathcal{P}_m \in [0,1)^d$ *with* $n = b^m$ *points is a* (t,m,d) *– net in base* b *if every* $\mathcal{A}$ *with volume* b^{t-m} *contains* b^t *points of* $\mathcal{P}_m$*.*

Digital (t,m,d)-nets are a special case of (t,m,d)-nets, constructed using matrix-vector multiplications over finite fields. Digital sequences are infinite length digital nets, i.e., the first $n = b^m$ points of a digital sequence comprise a digital net for all integer $m \in \mathbb{N}_0$.

Definition 2 *For any non-negative integer* $i = \dots i_3 i_2 i_1 (\text{base } b)$*, define the* $\infty \times 1$ *vector* $\boldsymbol{\imath}$ *as the vector of its digits, that is,* $\boldsymbol{\imath} = (i_1, i_2, \dots)^T$*. For any point* $z = 0.z_1 z_2 \dots (\text{base } b) \in [0,1)$*, define the* $\infty \times 1$ *vector of the digits of* z*, that is,* $\mathbf{z} = (z_1, z_2, \dots)^T$*. Let* $\mathsf{G}_1, \dots, \mathsf{G}_d$ *denote predetermined* $\infty \times \infty$ *generator matrices. The digital sequence in* base b *is* $\{\boldsymbol{z}_0, \boldsymbol{z}_1, \boldsymbol{z}_2, \dots\}$*, where each* $\boldsymbol{z}_i = (z_{i1}, \dots, z_{id})^T \in [0,1)^d$ *is defined by*

$$\mathbf{z}_{i\ell} = \mathsf{G}_\ell \, \boldsymbol{\imath}, \quad \ell = 1, \dots, d, \quad i = 0, 1, \dots \, .$$

The value of t *as mentioned in Definition 1 depends on the choice of* G_ℓ*.*

Digital nets have a group structure under digitwise addition, which is a very useful property exploited in our algorithm, especially to develop a fast Bayesian transform that speedups computations. Digitwise addition, $\oplus$, and subtraction $\ominus$, are defined in terms of b-ary expansions of points in $[0,1)^d$,

$$\boldsymbol{z} \oplus \boldsymbol{y} = \left(\sum_{j=1}^{\infty} [z_{\ell j} + y_{\ell j} \bmod b] b^{-j} \bmod 1 \right)_{\ell=1}^{d},$$

$$\boldsymbol{z} \ominus \boldsymbol{y} = \left(\sum_{j=1}^{\infty} [z_{\ell j} - y_{\ell j} \bmod b] b^{-j} \bmod 1 \right)_{\ell=1}^{d},$$

where

$$\boldsymbol{z} = \left(\sum_{j=1}^{\infty} z_{\ell j} b^{-j} \right)_{\ell=1}^{d}, \quad \boldsymbol{y} = \left(\sum_{j=1}^{\infty} y_{\ell j} b^{-j} \right)_{\ell=1}^{d}, \quad z_{\ell j}, y_{\ell j} \in \{0, \cdots, b-1\}.$$

Similarly for integer values in $\mathbb{N}_0^d$, the digitwise addition, $\oplus$, and subtraction $\ominus$, are defined in terms of their b-ary expansions,

$$\boldsymbol{k} \oplus \boldsymbol{l} = \left(\sum_{j=0}^{\infty} [k_{\ell j} + l_{\ell j} \bmod b] b^{j} \bmod 1 \right)_{\ell=1}^{d},$$

$$\boldsymbol{k} \ominus \boldsymbol{l} = \left(\sum_{j=0}^{\infty} [k_{\ell j} - l_{\ell j} \bmod b] b^{j} \bmod 1 \right)_{\ell=1}^{d},$$

where

$$\boldsymbol{k} = \left(\sum_{j=0}^{\infty} k_{\ell j} b^{j} \right)_{\ell=1}^{d}, \quad \boldsymbol{l} = \left(\sum_{j=0}^{\infty} l_{\ell j} b^{j} \right)_{\ell=1}^{d}, \quad \boldsymbol{k}_{\ell j}, \boldsymbol{l}_{\ell j} \in \{0, \cdots, b-1\}.$$

Let $\{\boldsymbol{z}_i\}_{i=0}^{b^m-1}$ be a digital net. Then

$\forall i_1, i_2 \in \{0, \cdots, b^m - 1\}, \quad \boldsymbol{z}_{j_1} \oplus \boldsymbol{z}_{i_2} = \boldsymbol{z}_{i_3}, \quad$ for some $i_3 \in \{0, \cdots, b^m - 1\}$.

The following very useful result, which will be further used to obtain the fast Bayesian transform, arises from the fundamental property of digital nets.

Lemma 2 *Let* $\{\boldsymbol{z}_i\}_{i=0}^{b^m-1}$ *be the digital-net and the corresponding digitally shifted net be* $\{\boldsymbol{x}_i\}_{i=0}^{b^m-1}$*, i.e.,*

$$\mathbf{x}_{i\ell} = \mathbf{z}_{i\ell} + \boldsymbol{\Delta}_l \bmod 1,$$

where $\mathbf{x}_{i\ell}$ *is the* ℓ*th component of* i*th digital net and* $\boldsymbol{\Delta}_\ell$ *is the digital shift for the* ℓ*th component. Then,*

$$\boldsymbol{x}_i \ominus \boldsymbol{x}_j = \boldsymbol{z}_i \ominus \boldsymbol{z}_j = \boldsymbol{z}_{i \ominus j}, \quad \forall i, j \in \mathbb{N}_0. \tag{9}$$

Also the digital subtraction is symmetric,

$$\boldsymbol{x}_i \ominus \boldsymbol{x}_i = \mathbf{0}, \qquad \boldsymbol{x}_i \ominus \boldsymbol{x}_j = \boldsymbol{x}_j \ominus \boldsymbol{x}_i, \quad \forall i, j \in \mathbb{N}_0. \tag{10}$$

Proof The proof can be obtained from the definition of digital nets which stated that the digital nets are obtained using generator matrices, $\mathbf{z}_{i\ell} = \mathsf{G}_\ell \boldsymbol{\imath} \bmod b$. Rewriting the subtraction using the generating matrix provides the result,

$$\begin{aligned}
\mathbf{z}_{i\ell} - \mathbf{z}_{j\ell} \bmod b &= (\mathsf{G}_\ell \boldsymbol{\imath} \bmod b) - (\mathsf{G}_\ell \boldsymbol{\jmath} \bmod b) \\
&= (\mathsf{G}_\ell \boldsymbol{\imath} - \mathsf{G}_\ell \boldsymbol{\jmath}) \bmod b \\
&= \mathsf{G}_\ell (\boldsymbol{\imath} - \boldsymbol{\jmath}) \bmod b \\
&= \mathsf{G}_\ell (\overrightarrow{i \ominus j}) \bmod b \\
&= \mathbf{z}_{i \ominus j \; \ell}.
\end{aligned}$$

The rest of the lemma is obvious from the definition of digital nets.

We chose digitally shifted and scrambled nets Hickernell and Yue [2000] for our Bayesian cubature algorithm. Digital shifts help to avoid having nodes at the origin, similar to the random shift used with lattice nodes. Scrambling helps to eliminate bias while retaining the low-discrepancy properties. A proof that a scrambled net preserves the property of (t, m, d)-net almost surely can be found in Owen Owen [1995]. The scrambling method proposed by Matoušek Matoušek [1998] is preferred since it is more efficient than the Owen's scrambling.

Sobol' nets Sobol' [1976] are a special case of (t, m, d)-nets when base $b = 2$. An example of 64 Sobol' nets in $d = 2$ is given in Figure **??**. The even coverage of the unit cube is ensured by a well chosen generating matrix. The choice of generating vector is typically done offline by computer search. See Kuo and Nuyens [2016] and Nuyens for more on generating matrices. We use randomly scrambled and digitally shifted Sobol' sequences in this research Hong and Hickernell [2003].

8 Walsh Kernels

Walsh kernels are product kernels based on the Walsh functions. We introduce the necessary concepts in this section.

8.1 Walsh functions

Like the Fourier transform used with lattice points (Section **??**), the Walsh-Hadamard transform, which we will simply call Walsh transform, is used for the digital nets. The Walsh transform is defined using Walsh functions. Recall $\mathbb{N}_0 := \{0, 1, 2, \cdots\}$. The one-dimensional Walsh functions in base b are defined as

$$\mathrm{wal}_{b,k}(x) := e^{2\pi\sqrt{-1}(x_1 k_0 + x_2 k_1 + \cdots)/b} = e^{2\pi\sqrt{-1}\mathbf{k}^T\mathbf{x}/b}, \tag{11}$$

for $x \in [0, 1)$ and $k \in \mathbb{N}_0$ and the unique base b expansions $x = \sum_{j \geq 1} x_j b^{-i} = (0.x_1 x_2 \cdots)_b$, $\mathbf{x} = (x_1, x_2, \cdots)^T$ $k = \sum_{j \geq 0} k_j b^j = (\cdots k_1 k_0)_b$, $\mathbf{k} = (k_0, k_1, \cdots)^T$, and $\mathbf{k}^T\mathbf{x} = x_1 k_0 + x_2 k_1 + \cdots$ where the number of digits used in (11) are limited to the length required to represent x or k, i.e., $\max\left(\lceil -\log_b x \rceil, \lceil \log_b k \rceil\right)$. Multivariate Walsh functions are defined as the product of the one-dimensional Walsh functions,

$$\mathrm{wal}_{b,\boldsymbol{k}}(\boldsymbol{x}) := \prod_{\ell=1}^{d} \mathrm{wal}_{b,k_\ell}(x_\ell)$$

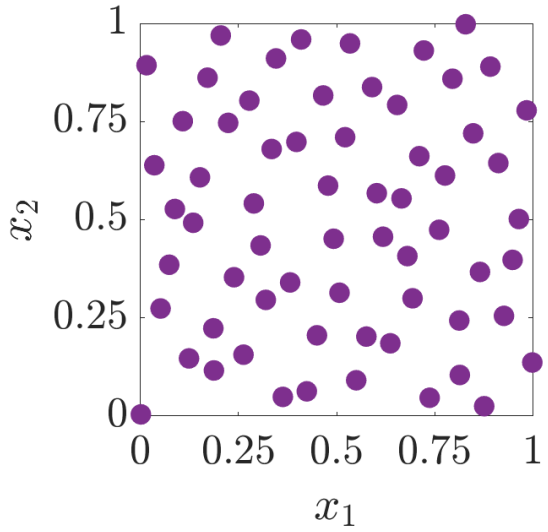

Fig. 3: Example of a scrambled Sobol' node set in $d = 2$. This plot can be reproduced using `PlotPoints.m`.

As shown in (11), for the case of $b = 2$, the Walsh functions only take the values in $\{1, -1\}$, i.e., $\mathrm{wal}_{b,\boldsymbol{k}} : [0,1)^d \to \{-1, 1\}$, $k \in \mathbb{N}_0^d$. Walsh functions form an orthonormal basis of the Hilbert space $L^2[0,1)^d$,

$$\int_{[0,1)^d} \mathrm{wal}_{b,\boldsymbol{l}}(\boldsymbol{x})\mathrm{wal}_{b,\boldsymbol{k}}(\boldsymbol{x})\mathrm{d}x = \delta_{\boldsymbol{l},\boldsymbol{k}}, \quad \forall \boldsymbol{l}, \boldsymbol{k} \in \mathbb{N}_0^d$$

Digital nets are designed to integrate certain Walsh functions without error. Thus our Bayesian cubature algorithm integrates linear combinations of certain Walsh functions without error. Functions that are well approximated by such linear combinations are then integrated with small errors.

In this research we use Sobol' nodes which are digital nets with base $b = 2$. So here afterwards base $b = 2$ is assumed. In this case, the Walsh function is simply

$$\mathrm{wal}_{2,\boldsymbol{k}}(\boldsymbol{x}) = (-1)^{\boldsymbol{k}^T\boldsymbol{x}}.$$

8.2 Walsh kernels

Consider the covariance kernels of the form,

$$C_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{t}) = K_{\boldsymbol{\theta}}(\boldsymbol{x} \ominus \boldsymbol{t}) \tag{12}$$

where $\ominus$ is bitwise subtraction. This is called a *digitally shift invariant kernel* because shifting both arguments of the covariance function by the same amount leaves the value unchanged. By a proper scaling of the function $K_{\boldsymbol{\theta}}$, it follows that assumption (**??**) is satisfied. The function $K_{\boldsymbol{\theta}}$ must be of the form that ensures that $C_{\boldsymbol{\theta}}$ is symmetric and positive definite, as assumed in (1). We drop the $\boldsymbol{\theta}$

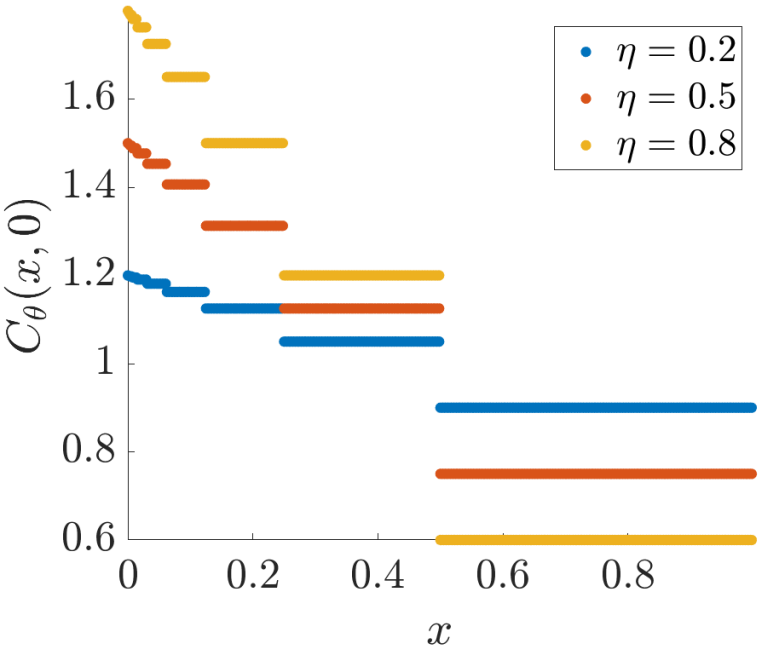

Fig. 4: Walsh kernel of order $r = 1$ in dimension $d = 1$. This figure can be reproduced using `plot_walsh_kernel.m`.

sometimes to make the notation simpler. The Walsh kernels are of the form,

$$K_{\boldsymbol{\theta}}(\boldsymbol{x} \ominus \boldsymbol{t}) = \prod_{\ell=1}^{d} 1 + \eta_\ell \omega_r(x_\ell \ominus t_\ell), \quad \boldsymbol{\eta} = (\eta_1, \cdots, \eta_d), \quad \boldsymbol{\theta} = (r, \boldsymbol{\eta}) \tag{13}$$

where r is the kernel order, $\boldsymbol{\eta}$ is the kernel shape parameter, and

$$\omega_r(x) = \sum_{k=1}^{\infty} \frac{\mathrm{wal}_{2,k}(x)}{2^{2r\lfloor \log_2 k \rfloor}}.$$

Explicit expression is available for ω_r in the case of order $r = 1$ Nuyens [2013],

$$\omega_1(x) = 6\left(\frac{1}{6} - 2^{\lfloor \log_2 x \rfloor - 1}\right). \tag{14}$$

The Figure 4 shows the Walsh kernel (13) of order $r = 1$ in the interval $[0, 1)$. Unlike the shift-invariant kernels used with lattice nodes, low order Walsh kernels are discontinuous and are only piecewise constant. Smaller η_ℓ implies lesser variation in the amplitude of the kernel. Also, the Walsh kernels are digitally shift invariant but not periodic.

9 Eigenvectors

We show the eigenvectors V in (**??**) of the Gram matrix formed by the covariance kernel (13) and Sobol' nets are the columns of the Walsh-Hadamard matrix. First we introduce the necessary concepts.

9.1 Walsh transform

The Walsh-Hadamard transform (WHT) is a generalized class of discrete Fourier transform (DFT) and is much simpler to compute than the DFT. The WHT matrices are comprised of only ± 1 values, so the computation usually involves only ordinary additions and subtractions. Hence, the WHT is also sometimes called the integer transform. In comparison, the DFT that was used with lattice nodes, uses complex exponential functions and the computation involves complex, non-integer multiplications.

The WHT involves multiplications by $2^m \times 2^m$ Walsh-Hadamard matrices, which is constructed recursively, starting with $\mathsf{H}^{(0)} = 1$,

$$\mathsf{H}^{(1)} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

$$\mathsf{H}^{(2)} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix},$$

$$\vdots$$

$$\mathsf{H}^{(m)} = \begin{pmatrix} \mathsf{H}^{(m-1)} & \mathsf{H}^{(m-1)} \\ \mathsf{H}^{(m-1)} & -\mathsf{H}^{(m-1)} \end{pmatrix} = \underbrace{\mathsf{H}^{(1)} \bigotimes \cdots \bigotimes \mathsf{H}^{(1)}}_{m \text{ times}} = \mathsf{H}^{(1)} \bigotimes \mathsf{H}^{(m-1)} \tag{15}$$

where $\bigotimes$ is Kronecker product. Alternatively for base $b = 2$, these matrices can be directly obtained by,

$$\mathsf{H}^{(m)} = \left((-1)^{(\boldsymbol{i}^T \boldsymbol{j})} \right)_{i,j=0}^{2^m - 1},$$

where the notation $\boldsymbol{i}^T \boldsymbol{j}$ indicates the bitwise dot product.

9.2 Eigenvectors of C are columns of Walsh-Hadamard matrix

The Gram matrix $\mathsf{C}_{\boldsymbol{\theta}}$ formed by Walsh kernels and Sobol' nodes have a special structure called block-Toeplitz, which can be used to construct the fast Bayesian transform. A Toeplitz matrix is a diagonal-constant matrix in which each descending diagonal from left to right is constant. A block Toeplitz matrix is a special block matrix, which contains blocks that are repeated down the diagonals of the matrix. We prove that the eigenvectors of $\mathsf{C}_{\boldsymbol{\theta}}$ are columns of a Walsh-Hadamard matrix in two theorems.

Theorem 2 *Let $(\boldsymbol{x}_i)_{i=0}^{n-1}$ be digitally shifted Sobol' nodes and K be any function, then the Gram matrix,*
$\mathsf{C}_{\boldsymbol{\theta}} = \big(C(\boldsymbol{x}_i, \boldsymbol{x}_j)\big)_{i,j=0}^{n-1} = \big(K(\boldsymbol{x}_i \ominus \boldsymbol{x}_j)\big)_{i,j=0}^{n-1}$,

$$\text{where} \qquad n = 2^m, \quad C(\boldsymbol{x}, \boldsymbol{t}) = K(\boldsymbol{x} \ominus \boldsymbol{t}), \quad \boldsymbol{x}, \boldsymbol{t} \in [0,1)^d,$$

is a 2×2 *block-Toeplitz matrix and all the sub-blocks and their sub-sub-blocks, etc. are also* 2×2 *block-Toeplitz.*

Proof We prove this theorem by induction. Let $\mathsf{C}_{\boldsymbol{\theta}}^{(m)}$ denote the Gram matrix of size $2^m \times 2^m$. The relation between sub-block matrices can be deciphered using the properties of digital nets. To help with the proof of block-Toeplitz structure, consider the digital net properties (9), (10), and notations,

$$\mathsf{K}^{(m)} := \big(K(\boldsymbol{z}_i \ominus \boldsymbol{z}_j)\big)_{i,j=0}^{2^m-1} = \big(K(\boldsymbol{z}_{i\ominus j})\big)_{i,j=0}^{2^m-1}, \quad m = 1, 2, \cdots,$$
$$\mathsf{K}^{(m,q)} := \big(K(\boldsymbol{z}_{i\ominus j+q2^m})\big)_{i,j=0}^{2^m-1}, \quad q = 0, 1, \cdots.$$

These two notations are related by $\mathsf{K}^{(m)} = \mathsf{K}^{(m,0)}$. Please note that $\mathsf{C}_{\boldsymbol{\theta}}^{(m)} = \mathsf{K}^{(m,0)}$. We will prove $\mathsf{K}^{(m,q)}$ is a 2×2 block-toeplitz matrix for all $m \in \mathbb{N}, q \in \mathbb{N}$.

As the first step, we verify the property holds for $m = 1$,

$$\mathsf{K}^{(1,q)} = \begin{pmatrix} K(\boldsymbol{z}_{0\ominus 0+q2^1}) & K(\boldsymbol{z}_{1\ominus 0+q2^1}) \\ K(\boldsymbol{z}_{0\ominus 1+q2^1}) & K(\boldsymbol{z}_{1\ominus 1+q2^1}) \end{pmatrix} = \begin{pmatrix} K(\boldsymbol{z}_{2q}) & K(\boldsymbol{z}_{1+2q}) \\ K(\boldsymbol{z}_{1+2q}) & K(\boldsymbol{z}_{2q}) \end{pmatrix}, \quad \text{by (9)}$$

has diagonal elements repeated. Thus by definition, it is a 2×2 block-Toeplitz.

Now assume that $\mathsf{K}^{(m,q)}$ is block-Toeplitz. We need to prove $\mathsf{K}^{(m+1,q)}$ is also a 2×2 block-Toeplitz. Let $n = 2^m$,

$$\mathsf{K}^{(m+1)} = \begin{pmatrix} K(\boldsymbol{z}_{0\ominus 0}) & \dots & K(\boldsymbol{z}_{0\ominus n-1}) & K(\boldsymbol{z}_{0\ominus n}) & \dots & K(\boldsymbol{z}_{0\ominus 2n-1}) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ K(\boldsymbol{z}_{n-1\ominus 0}) & \dots & K(\boldsymbol{z}_{n-1\ominus n-1}) & K(\boldsymbol{z}_{n-1\ominus n}) & \dots & K(\boldsymbol{z}_{n-1\ominus 2n-1}) \\ K(\boldsymbol{z}_{n\ominus 0}) & \dots & K(\boldsymbol{z}_{n\ominus n-1}) & K(\boldsymbol{z}_{n\ominus n}) & \dots & K(\boldsymbol{z}_{n\ominus 2n-1}) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ K(\boldsymbol{z}_{2n-1\ominus 0}) & \dots & K(\boldsymbol{z}_{2n-1\ominus n-1}) & K(\boldsymbol{z}_{2n-1\ominus n}) & \dots & K(\boldsymbol{z}_{2n-1\ominus 2n-1}) \end{pmatrix}$$

$$= \begin{pmatrix} \begin{pmatrix} K(\boldsymbol{z}_0) & \dots & K(\boldsymbol{z}_{n-1}) \\ \vdots & \vdots & \vdots \\ K(\boldsymbol{z}_{n-1}) & \dots & K(\boldsymbol{z}_0) \end{pmatrix} & \begin{pmatrix} K(\boldsymbol{z}_n) & \dots & K(\boldsymbol{z}_{2n-1}) \\ \vdots & \vdots & \vdots \\ K(\boldsymbol{z}_{2n-1}) & \dots & K(\boldsymbol{z}_n) \end{pmatrix} \\ \begin{pmatrix} K(\boldsymbol{z}_n) & \dots & K(\boldsymbol{z}_{2n-1}) \\ \vdots & \vdots & \vdots \\ K(\boldsymbol{z}_{2n-1}) & \dots & K(\boldsymbol{z}_n) \end{pmatrix} & \begin{pmatrix} K(\boldsymbol{z}_0) & \dots & K(\boldsymbol{z}_{n-1}) \\ \vdots & \vdots & \vdots \\ K(\boldsymbol{z}_{n-1}) & \dots & K(\boldsymbol{z}_0) \end{pmatrix} \end{pmatrix}$$

$$= \begin{pmatrix} \mathsf{K}^{(m)} & \mathsf{K}^{(m,1)} \\ \mathsf{K}^{(m,1)} & \mathsf{K}^{(m)} \end{pmatrix}$$

is a 2×2 block-Toeplitz, where we used the properties (9), (10) and facts $2n-1 \ominus n = n-1$, $2n-1 \ominus n-1 = n$, and $n \ominus n-1 = 2n-1$. Thus $\mathsf{K}^{(m+1)}$ is a 2×2 block-Toeplitz. Similarly

$$\mathsf{K}^{(m+1,q)} = \begin{pmatrix} \mathsf{K}^{(m,q)} & \mathsf{K}^{(m,q+1)} \\ \mathsf{K}^{(m,q+1)} & \mathsf{K}^{(m,q)} \end{pmatrix}$$

is a 2×2 block-Toeplitz. Thus $\mathsf{C}_{\boldsymbol{\theta}}^{(m)}$ of size $2^m \times 2^m$, for $m \in \mathbb{N}$, is a 2×2 block-Toeplitz and every block and it's sub-blocks of size 2^p, $p \in \mathbb{N}$, $p \le m$ are also 2×2 block-Toeplitz.

Theorem 3 *The Walsh-Hadamard matrix* $\mathsf{H}^{(m)}$ *factorizes* $\mathsf{C}_{\boldsymbol{\theta}}^{(m)}$*, so that the columns of Walsh-Hadamard matrix are the eigenvectors of* $\mathsf{C}_{\boldsymbol{\theta}}^{(m)}$*, i.e.,*

$$\mathsf{H}^{(m)} \mathsf{C}_{\boldsymbol{\theta}}^{(m)} = \Lambda^{(m)} \mathsf{H}^{(m)}, \quad m \in \mathbb{N}.$$

Proof Again, we use the proof-by-induction technique to show that the Walsh-Hadamard matrix factorizes $\mathsf{K}^{(m,q)}$. We can easily see the Hadamard matrix $\mathsf{H}^{(1)}$ diagonalizes $\mathsf{K}^{(1,q)}$,

$$\begin{aligned}
\mathsf{H}^{(1)}\mathsf{K}^{(1,q)} &= \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} K(\boldsymbol{z}_{0+q2^1}) & K(\boldsymbol{z}_{1+q2^1}) \\ K(\boldsymbol{z}_{1+q2^1}) & K(\boldsymbol{z}_{0+q2^1}) \end{pmatrix}, \quad \text{by Theorem 2} \\
&= \begin{pmatrix} K(\boldsymbol{z}_{2q}) + K(\boldsymbol{z}_{2q+1}) & K(\boldsymbol{z}_{2q}) + K(\boldsymbol{z}_{2q+1}) \\ K(\boldsymbol{z}_{2q}) - K(\boldsymbol{z}_{2q+1}) & K(\boldsymbol{z}_{2q+1}) - K(\boldsymbol{z}_{2q}) \end{pmatrix} \\
&= \begin{pmatrix} K(\boldsymbol{z}_{2q}) + K(\boldsymbol{z}_{2q+1}) & 0 \\ 0 & K(\boldsymbol{z}_{2q}) - K(\boldsymbol{z}_{2q+1}) \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\
&= \mathsf{\Lambda}^{(1,q)}\mathsf{H}^{(1)},
\end{aligned}$$

where $\mathsf{\Lambda}^{(1,q)}$ is a diagonal matrix, thus $\mathsf{H}^{(1)}$ factorizes $\mathsf{K}^{(1,q)}$.

Now assume $\mathsf{H}^{(m)}$ factorizes $\mathsf{K}^{(m,q)}$, so $\mathsf{H}^{(m)}\mathsf{K}^{(m,q)} = \mathsf{\Lambda}^{(m,q)}\mathsf{H}^{(m)}$ where $\mathsf{\Lambda}^{(m,q)}$ is diagonal. We need to prove $\mathsf{H}^{(m+1)}$ factorizes $\mathsf{K}^{(m+1,q)}$,

$$\begin{aligned}
\mathsf{H}^{(m+1)}\mathsf{K}^{(m+1,q)} &= \begin{pmatrix} \mathsf{H}^{(m)} & \mathsf{H}^{(m)} \\ \mathsf{H}^{(m)} & -\mathsf{H}^{(m)} \end{pmatrix} \begin{pmatrix} \mathsf{K}^{(m,q)} & \mathsf{K}^{(m,q+1)} \\ \mathsf{K}^{(m,q+1)} & \mathsf{K}^{(m,q)} \end{pmatrix}, \quad \text{by Theorem 2} \\
&= \begin{pmatrix} \mathsf{H}^{(m)}(\mathsf{K}^{(m,q)} + \mathsf{K}^{(m,q+1)}) & \mathsf{H}^{(m)}(\mathsf{K}^{(m,q)} + \mathsf{K}^{(m,q+1)}) \\ \mathsf{H}^{(m)}(\mathsf{K}^{(m,q)} - \mathsf{K}^{(m,q+1)}) & \mathsf{H}^{(m)}(\mathsf{K}^{(m,q+1)} - \mathsf{K}^{(m,q)}) \end{pmatrix} \\
&= \begin{pmatrix} (\mathsf{\Lambda}^{(m,q)} + \mathsf{\Lambda}^{(m,q+1)})\mathsf{H}^{(m)} & (\mathsf{\Lambda}^{(m,q)} + \mathsf{\Lambda}^{(m,q+1)})\mathsf{H}^{(m)} \\ (\mathsf{\Lambda}^{(m,q)} - \mathsf{\Lambda}^{(m,q+1)})\mathsf{H}^{(m)} & (\mathsf{\Lambda}^{(m,q+1)} - \mathsf{\Lambda}^{(m,q)})\mathsf{H}^{(m)} \end{pmatrix} \\
&= \begin{pmatrix} \mathsf{\Lambda}^{(m,q)} + \mathsf{\Lambda}^{(m,q+1)} & 0 \\ 0 & \mathsf{\Lambda}^{(m,q)} - \mathsf{\Lambda}^{(m,q+1)} \end{pmatrix} \begin{pmatrix} \mathsf{H}^{(m)} & \mathsf{H}^{(m)} \\ \mathsf{H}^{(m)} & -\mathsf{H}^{(m)} \end{pmatrix} \\
&= \mathsf{\Lambda}^{(m+1,q)}\mathsf{H}^{(m+1)}.
\end{aligned}$$

Thus, $\mathsf{H}^{(m+1)}$ factorizes $\mathsf{K}^{(m+1,q)}$ to a diagonal matrix $\mathsf{\Lambda}^{(m+1,q)}$. This implies $\mathsf{H}^{(p)}$ factorizes $\mathsf{C}_{\boldsymbol{\theta}}^{(p)}$ for $p \in \mathbb{N}$. Please recall $\mathsf{C}_{\boldsymbol{\theta}}^{(p)} = \mathsf{K}^{(p,0)}$. Here we used the fact that both H and K are symmetric positive definite.

9.3 Fast Bayesian transform

We can easily show that the Walsh-Hadamard matrices satisfy the assumptions of fast Bayesian transform (**??**). As shown in Section 9.2 the columns of $\mathsf{H}^{(m)}$ are the eigenvectors. Since the Gram matrix C is symmetric, the columns/rows of Walsh-Hadamard matrices are mutually orthogonal. Thus the Gram matrix can be written as

$$\mathsf{C}^{(m)} = \frac{1}{n}\mathsf{H}^{(m)}\mathsf{\Lambda}^{(m)}\mathsf{H}^{(m)}, \quad \text{where} \quad \mathsf{H}^{(m)} = \underbrace{\mathsf{H}^{(1)} \bigotimes \cdots \bigotimes \mathsf{H}^{(1)}}_{m \text{ times}}. \tag{16}$$

Assumption (**??**) follows automatically by the fact that Walsh-Hadamard matrices can be constructed analytically. Assumption (**??**) can also be verified as the first row/column are one vectors. Finally, assumption (**??**) is satisfied due to the fact that fast Walsh transform can be computed in $\mathcal{O}(n \log n)$ operations using fast Walsh-Hadamard transform. Thus the Walsh-Hadamard transform is a fast Bayesian transform, $\mathsf{V} := \mathsf{H}$, as per (**??**).

We have implemented a fast adaptive Bayesian cubature algorithm using the kernel (13) with $r = 1$ and Sobol' points Bratley and Fox [1988] in MATLAB as

part of the Guaranteed Adaptive Integration Library (GAIL) Choi et al. [2013–2017] as `cubBayesNet_g`. The Sobol' points used in this algorithm are generated using MATLAB's builtin function `sobolset` and scrambled using MATLAB function `scramble` Hong and Hickernell [2003]. The fast Walsh-Hadamard transform (16) is computed using MATLAB's builtin function `fwht` with *hadamard* ordering.

9.4 Iterative Computation of Walsh Transform

In every iteration of our algorithm, we double the number of function values. Using the technique described here, we have to only compute the Walsh transform for the newly added function values. Similar to the lattice points, Sobol' points are extensible by definition. This property is used in our algorithm to improve the integration accuracy till the required error tolerance is met. Sobol' nodes can be combined with Hadamard matrices as demonstrated here for iterative computation. Let $\widetilde{\boldsymbol{y}} = \mathsf{H}^{(m+1)}\boldsymbol{y}$ for some arbitrary $\boldsymbol{y} \in \mathbb{R}^{2n}$, $n = 2^m$. Define,

$$\boldsymbol{y} = \begin{pmatrix} [1.1]y_1 \\ \vdots \\ y_{2n} \end{pmatrix}, \quad \boldsymbol{y}^{(1)} = \begin{pmatrix} [1.1]y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad \boldsymbol{y}^{(2)} = \begin{pmatrix} [1.1]y_{n+1} \\ \vdots \\ y_{2n} \end{pmatrix},$$

$$\widetilde{\boldsymbol{y}}^{(1)} = \mathsf{H}^{(m)}\boldsymbol{y}^{(1)} = \begin{pmatrix} [1.0]\widetilde{y}_1^{(1)} \\ \widetilde{y}_2^{(1)} \\ \vdots \\ \widetilde{y}_n^{(1)} \end{pmatrix}, \quad \widetilde{\boldsymbol{y}}^{(2)} = \mathsf{H}^{(m)}\boldsymbol{y}^{(2)} = \begin{pmatrix} [1.0]\widetilde{y}_1^{(2)} \\ \widetilde{y}_2^{(2)} \\ \vdots \\ \widetilde{y}_n^{(2)} \end{pmatrix}.$$

Then,

$$\begin{aligned} \widetilde{\boldsymbol{y}} &= \mathsf{H}^{(m+1)}\boldsymbol{y} \\ &= \begin{pmatrix} \mathsf{H}^{(m)} & \mathsf{H}^{(m)} \\ \mathsf{H}^{(m)} & -\mathsf{H}^{(m)} \end{pmatrix} \begin{pmatrix} \boldsymbol{y}^{(1)} \\ \boldsymbol{y}^{(2)} \end{pmatrix}, \qquad \text{by (15)} \\ &= \begin{pmatrix} \mathsf{H}^{(m)}\boldsymbol{y}^{(1)} + \mathsf{H}^{(m)}\boldsymbol{y}^{(2)} \\ \mathsf{H}^{(m)}\boldsymbol{y}^{(1)} - \mathsf{H}^{(m)}\boldsymbol{y}^{(2)} \end{pmatrix} \\ &= \begin{pmatrix} \widetilde{\boldsymbol{y}}^{(1)} + \widetilde{\boldsymbol{y}}^{(2)} \\ \widetilde{\boldsymbol{y}}^{(1)} - \widetilde{\boldsymbol{y}}^{(2)} \end{pmatrix} =: \widetilde{\boldsymbol{y}} \quad . \end{aligned}$$

As before with the lattice nodes, the computational cost to compute $\mathsf{V}^{(m+1)H}\boldsymbol{y}$ is twice the cost of computing $\mathsf{V}^{(m)H}\boldsymbol{y}^{(1)}$ plus $2n$ additions, where $n = 2^m$. An inductive argument shows that for any $m \in \mathbb{N}$, $\mathsf{V}^{(m)H}\boldsymbol{y}$ requires only $\mathcal{O}(n \log n)$ operations. Usually the multiplications in $\mathsf{V}^{(m)H}\boldsymbol{y}^{(1)}$ are multiplications by -1 which are simply accomplished using sign change or negation, requiring no multiplications at all.

10 Higher Order Nets

Higher order digital nets are an extension of (t, m, d)-nets, introduced in Dick [2008]. They can be used to numerically integrate smoother functions which are not necessarily periodic, but have square integrable mixed partial derivatives of order α, at a rate of $\mathcal{O}(n^{-\alpha})$ multiplied by a power of a $\log n$ factor using rules

corresponding to the modified (t, m, d)-nets. We want to emphasize that quasi-Monte Carlo rules based on these point sets can achieve convergence rates faster than $\mathcal{O}(n^{-1})$. Higher order digital nets are constructed using matrix-vector multiplications over finite fields.

One could develop matching digitally shift invariant kernels to formulate the fast Bayesian cubature. Bayesian cubatures using higher order digital nets are a topic for future research.

11 Kernel Hyperparameters Search

JR: Explain the transformation used to make the search range positive, > 0 , etc.

The various hyperparameters introduced and used by our algorithms need to be optimally chosen. The parameter search can be done in two major ways. Bounded minima search, if the search interval is known, else unbounded search. Most of the scenarios, the search interval is unknown. So the natural choice is to use unbounded search over the unbound domain such as `fminsearch` provided by MATLAB. However hyperparameters need to live in a domain that is bounded or semi-bounded. There are some simple domain transformations available to achieve this.

11.1 Positive kernel shape parameter

The following parameter map is used to ensure that the shape parameter values are positive real numbers. For $\eta > 0$ as introduced in Section **??**, let

$$\eta(t_1) = \quad e^{t_1}, \qquad\qquad \eta : (-\infty, \infty) \to (0, \infty).$$

Instead of searching for $\eta \in (0, \infty)$, we may search for the optimal $t_1 = \log(\eta)$ over the whole real line $\mathbb{R}$. The optimal value $t_{1,\text{opt}}$ can be transformed back to the $(0, \infty)$ interval using

$$\eta_{\text{opt}} = e^{t_{1,\text{opt}}}.$$

11.2 Kernel order $1 < r < \infty$

The following map is used to ensure that the kernel order values are positive real number and greater than one, i.e., in the $(1, \infty)$ interval as required in Section **??**,

$$r(t_2) = \quad 1 + e^{t_2}, \qquad\qquad r : (-\infty, \infty) \to (1, \infty).$$

So one may search for the optimal $t_2 = -\log(r - 1)$ in the whole real line $\mathbb{R}$. The optimal value $t_{2,\text{opt}}$ can be transformed back to the desired interval $(0, 1)$ using

$$r_{\text{opt}} = 1 + e^{t_{2,\text{opt}}}.$$

11.3 Kernel order $0 < q < 1$

The following multivariate map is used to ensure that the kernel order values are positive real and less than one, i.e., in the $(0, 1)$ interval to use with exponentially decaying kernel, as introduced in Section **??**,

$$q(t_3) = \quad \frac{1}{1 + e^{t_3}}, \qquad\qquad q : (-\infty, \infty) \to (0, 1).$$

So one may search for the optimal $t_3 = \log(q^{-1} - 1)$ in the whole real line $\mathbb{R}$. The optimal value $t_{3,\text{opt}}$ can be transformed back to the desired interval $(0, 1)$ by using

$$q_{\text{opt}} = \frac{1}{1 + e^{t_{3,\text{opt}}}}.$$

11.4 Combined searching of kernel order r and shape parameter $\boldsymbol{\eta}$

Instead of searching $\boldsymbol{\eta}$ and r separately one would prefer to search them together so that the most optimal values can be obtained, where $\boldsymbol{\eta} = (\eta_1, \cdots, \eta_d)$ such that $\eta_l \neq \eta_k$, for $l \neq k$. We can combine the parameter maps used above to ensure that the kernel order values in $(1, \infty)$ and shape parameter $\boldsymbol{\eta}$ in $(0, \infty)^d$ as required in Section **??**,

$$\boldsymbol{\theta}(\boldsymbol{t}) = \begin{pmatrix} [1]r(t_1) \\ \eta(t_2) \\ \vdots \\ \eta(t_{d+1}) \end{pmatrix} = \begin{pmatrix} [1]1 + e^{t_1} \\ e^{t_2} \\ \vdots \\ e^{t_{d+1}} \end{pmatrix}, \quad \boldsymbol{\theta} : \mathbb{R}^{d+1} \to (1, \infty) \times (0, \infty)^d.$$

So instead of searching for $\boldsymbol{\theta}_{\text{opt}}$ in $(1, \infty) \times (0, \infty)^d$, one may search for the optimal

$$\boldsymbol{t} = \boldsymbol{t}^{-1}(\boldsymbol{\theta}) = \begin{pmatrix} [1]\log(r - 1) \\ \log(\eta_1) \\ \vdots \\ \log(\eta_d) \end{pmatrix}$$

in the whole real line $\mathbb{R}^{d+1}$. The optimal value $\boldsymbol{t}_{\text{opt}}$ can be transformed back to the desired interval $(1, \infty) \times (0, \infty)^d$ using

$$\boldsymbol{\theta}_{\text{opt}} = \begin{pmatrix} [1]1 + e^{t_{1,\text{opt}}} \\ e^{t_{2,\text{opt}}} \\ \vdots \\ e^{t_{d+1,\text{opt}}} \end{pmatrix}.$$

Similarly one can map the kernel order $q \in (0, 1)$ Section **??**, and $\boldsymbol{\eta}$ in to a multivariate hyperparameter search.

12 Numerical Experiments

JR: use uniformly randomly chosen ε instead 4 fixed

Fast Bayesian cubature algorithms developed in this research are demonstrated using three commonly used integration examples. These integrals were evaluated using both the algorithms `cubBayesLattice_g` and `cubBayesNet_g`. The first example shows evaluating a multivariate Gaussian probability given the interval. The second example shows integrating the Keister's function, and the final example shows computing an Asian arithmetic option pricing.

12.1 Testing Methodology

Four hundred different error tolerances, ε, were randomly chosen from a fixed interval for each example. The intervals for error tolerance were chosen depending

on the difficulty of the problem. The nodes used in `cubBayesLattice_g` were the randomly shifted lattice points supplied by GAIL, whereas the nodes used in `cubBayesNet_g` were the randomly scrambled and shifted Sobol' points supplied by MATLAB's Sobol' sequence generator.

For each integral example, and each stopping criteria—empirical Bayes, full Bayes, and generalized cross-validation—our algorithm is run with each randomly chosen error tolerance as mentioned above. For each test, the execution time is plotted against $|\mu - \widehat{\mu}|/\varepsilon$. We expect $|\mu - \widehat{\mu}|/\varepsilon$ to be no greater than one, but hope that it is not too much smaller than one, which would indicate a stopping criterion that is too conservative.

Periodization variable transforms are used in the examples with `cubBayesLattice_g`, which assumes the integrands to be periodic in $[0,1]^d$. But the `cubBayesNet_g` does not need this additional requirement, so the integrands are used directly.

12.2 Multivariate Gaussian Probability

This example is introduced in Section **??**, where we use the Matérn covariance kernel. We reuse f_{Genz} (**??**) and apply a periodization transform to obtain f_{GenzP} when required.

Here we use f_{Genz} (**??**) without any periodization, and chose $d = 3$ and $r = 1$. The simulation results for this example integrand are summarized in Figures 5, 6, and 7. In all cases, `cubBayesNet_g` returns an approximation within the prescribed error tolerance. We used the same setting as before with generic slow Bayesian cubature in Section **??** for comparision. For error threshold $\varepsilon = 10^{-5}$ with empirical stopping criterion, our fast algorithm takes about 2 seconds as shown in Figure **??** whereas the basic algorithm takes 30 seconds as shown in Figure **??**. `cubBayesNet_g` uses fast Walsh transform which is slower in MATLAB due to the way it was implemented. This is reason it takes more longer the `cubBayesLattice_g`. But comparing the number of samples, n, used for integration provides more insight which directly relates to alogrithm's computational cost. The `cubBayesLattice_g` used $n = 16384$ samples whereas `cubBayesNet_g` used $n = 32768$ samples even with $r = 1$ order kernel.

Amongst the three stopping criteria, GCV achieved the results faster than others but it is less conservative. One can also observe from the figures that the credible intervals are narrower than in Figure **??**. This shows that `cubBayesNet_g` with $r = 1$ kernel more accurately approximates the integrand.

12.3 Keister's Example

This multidimensional integral function comes from Keister [1996] and is inspired by a physics application:

$$\begin{aligned} \mu &= \int_{\mathbb{R}^d} \cos(\|\boldsymbol{t}\|) \exp(-\|\boldsymbol{t}\|^2)\, \mathrm{d}\boldsymbol{t} \\ &= \int_{[0,1]^d} f_{\text{Keister}}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}, \end{aligned} \tag{17}$$

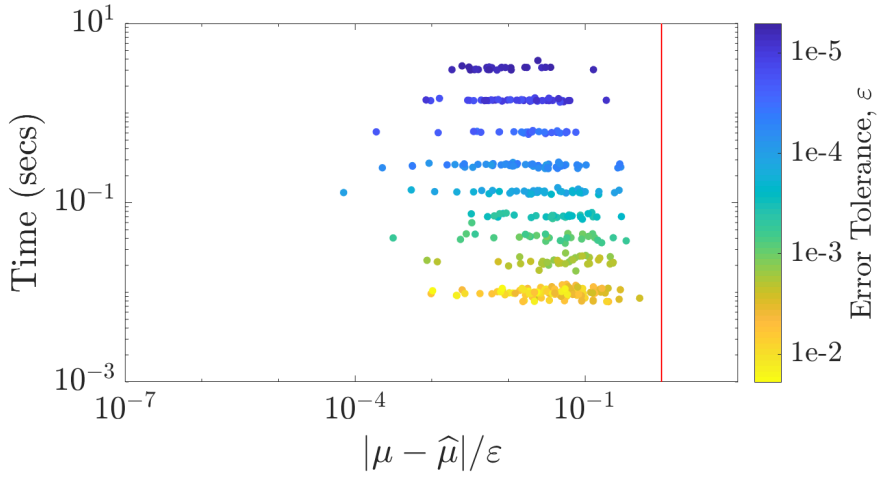

Fig. 5: Multivariate normal probability example with empirical Bayes stopping criterion.

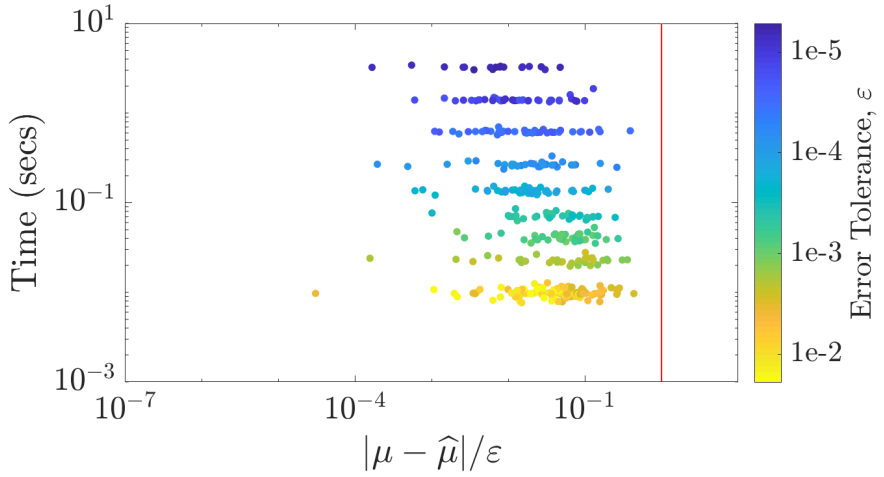

Fig. 6: Multivariate normal probability example with the full-Bayes stopping criterion.

where

$$f_{\text{Keister}}(\boldsymbol{x}) = \pi^{d/2} \cos\left(\left\|\Phi^{-1}(\boldsymbol{x})/2\right\|\right),$$

and Φ is the standard normal distribution. The true value of μ can be calculated iteratively in terms of a quadrature as follows:

$$\mu = \frac{2\pi^{d/2} I_c(d)}{\Gamma(d/2)}, \quad d = 1, 2, \ldots$$

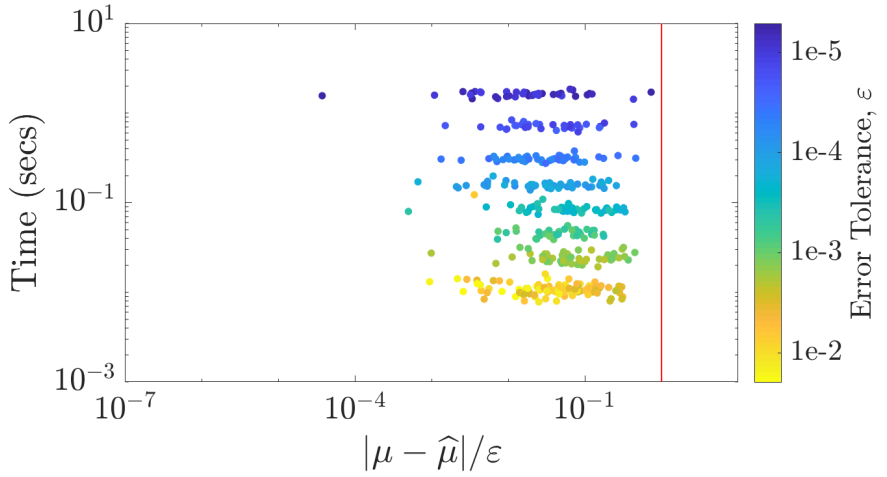

Fig. 7: Multivariate normal probability example with the GCV stopping criterion.

where Γ denotes the gamma function, and

$$
\begin{aligned}
I_c(1) &= \frac{\sqrt{\pi}}{2\exp(1/4)},\\
I_s(1) &= \int_{x=0}^{\infty} \exp(-\boldsymbol{x}^T\boldsymbol{x})\sin(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\\
&= 0.4244363835020225,\\
I_c(2) &= \frac{1-I_s(1)}{2}, \qquad I_s(2) = \frac{I_c(1)}{2}\\
I_c(j) &= \frac{(j-2)I_c(j-2) - I_s(j-1)}{2}, \qquad j = 3,4,\ldots\\
I_s(j) &= \frac{(j-2)I_s(j-2) - I_c(j-1)}{2}, \qquad j = 3,4,\ldots.
\end{aligned}
$$

Figures 8, 9 and 10 summarize the numerical tests for this case. We used dimension $d = 4$, and $r = 1$. No periodization transform was used as the integrand need not be periodic. In this example, we use $r = 1$ order kernel whereas in Section **??**, $r = 2$ kernel was used. This necessitates `cubBayesNet_g` to use more samples for integration. As observed from the figures, the GCV stopping criterion achieved the results faster than the others but it is less conservative which is also the case with the multivariate Gaussian example.

12.4 Option Pricing

The price of financial derivatives can often be modeled by high dimensional integrals. If the underlying asset is described in terms of a discretized geometric Brownian motion, then the fair price of the option is:

$$
\mu = \int_{\mathbb{R}^d} \text{payoff}(\boldsymbol{z}) \frac{\exp(\frac{1}{2}\boldsymbol{z}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{z})}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}}\,\mathrm{d}\boldsymbol{z} = \int_{[0,1]^d} f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x},
$$

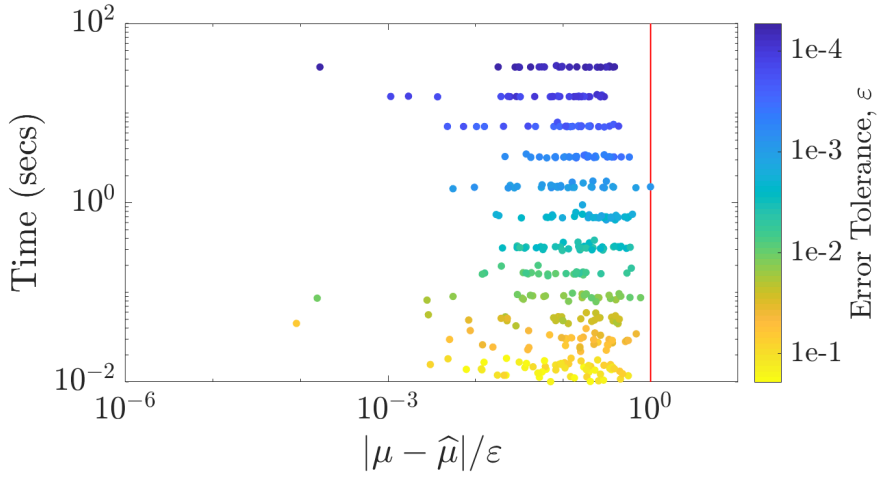

Fig. 8: Keister example using the empirical Bayes stopping criterion.

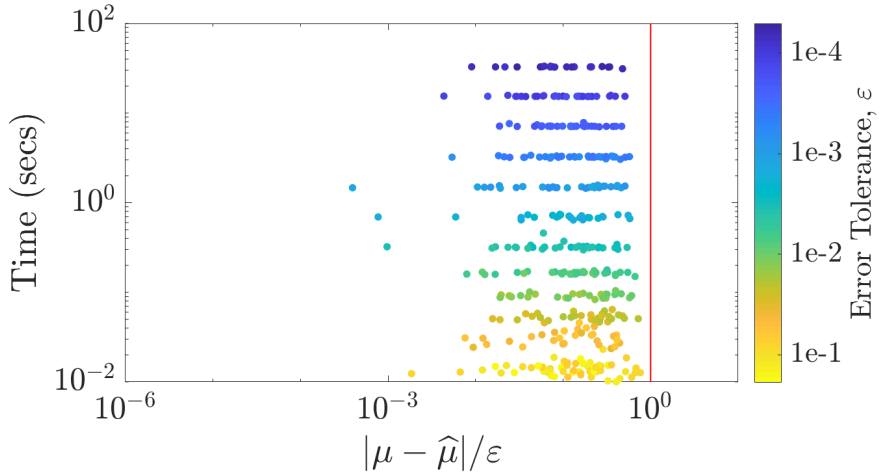

Fig. 9: Keister example using the full-Bayes stopping criterion.

where payoff$(\cdot)$ defines the discounted payoff of the option,

$$\Sigma = (T/d)\big(\min(j,k)\big)_{j,k=1}^{d} = \mathsf{L}\mathsf{L}^T,$$

$$f(\boldsymbol{x}) = \text{payoff}\left(\mathsf{L}\begin{pmatrix}\Phi^{-1}(x_1)\\ \vdots \\ \Phi^{-1}(x_d)\end{pmatrix}\right).$$

The Asian arithmetic mean call option has a payoff of the form

$$\text{payoff}(\boldsymbol{z}) = \max\left(\frac{1}{d}\sum_{j=1}^{d} S_j(\boldsymbol{z}) - K, 0\right)\mathrm{e}^{-rT},$$

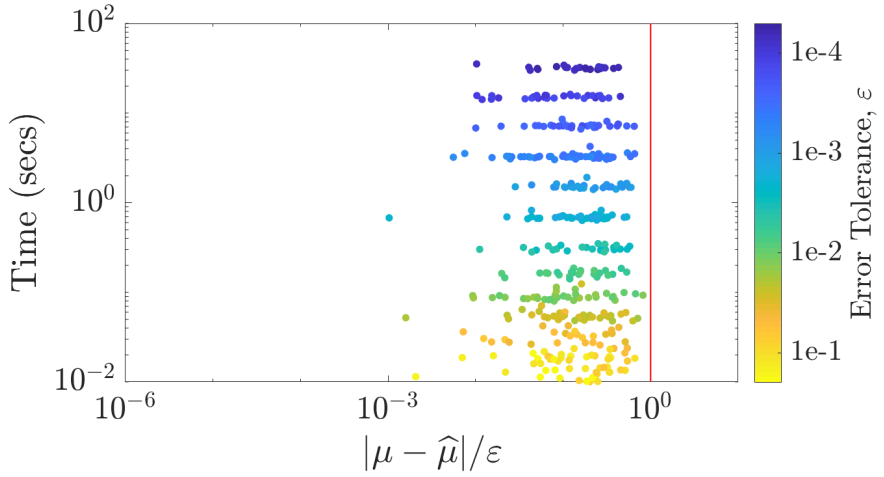

Fig. 10: Keister example using the GCV stopping criterion.

$$S_j(\boldsymbol{z}) = S_0 \exp\bigl((r - \sigma^2/2)jT/d + \sigma\sqrt{T/d}z_j\bigr).$$

Here, T denotes the time to maturity of the option, d the number of time steps, S_0 the initial price of the stock, r the interest rate, σ the volatility, and K the strike price.

The Figures 11, 12 and 13 summarize the numerical results for the option pricing example using the same values for, T, d, S_0, r, σ, K, as in Section **??**. As mentioned before, this integrand has a kink caused by the max function, so, `cubBayesNet_g` could be more efficient than `cubBayesLattice_g`, as no periodization transform is required. This can be observed from the number of samples used for intgration to meet the same error threshold. For the error tolerance, $\varepsilon = 10^{-3}$, `cubBayesLattice_g` used $n = 2^{20}$ samples, whereas `cubBayesNet_g` used $n = 2^{17}$ samples.

12.5 Discussion

JR: Move this to end of chapter

As shown in Figures **??** to 13, both the algorithms computed the integral within user specified threshold most of the time except on a few occasions. This is especially the case with option pricing example due to the complexity and high dimension of the integrand. Also notice that the `cubBayesLattice_g` algorithm finished within 10 seconds for Keister and multivariate Gaussian. Option pricing took closer to 70 seconds due to the complexity of the integrand.

Another noticeable aspect from the plots of `cubBayesLattice_g` is how much the error bounds differ from the true error. For option pricing example, the error bound is not as conservative as it is for the multivariate Gaussian and Keister examples. A possible reason is that the latter integrands are significantly smoother than the covariance kernel. This is a matter for further investigation.

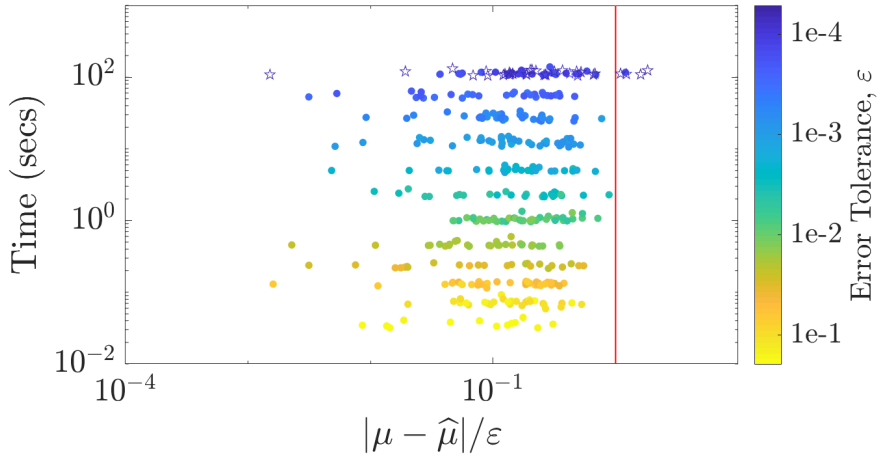

Fig. 11: Option pricing using the empirical Bayes stopping criterion. The hollow stars indicate the algorithm has not met the error threshold ϵ even with using maximum n.

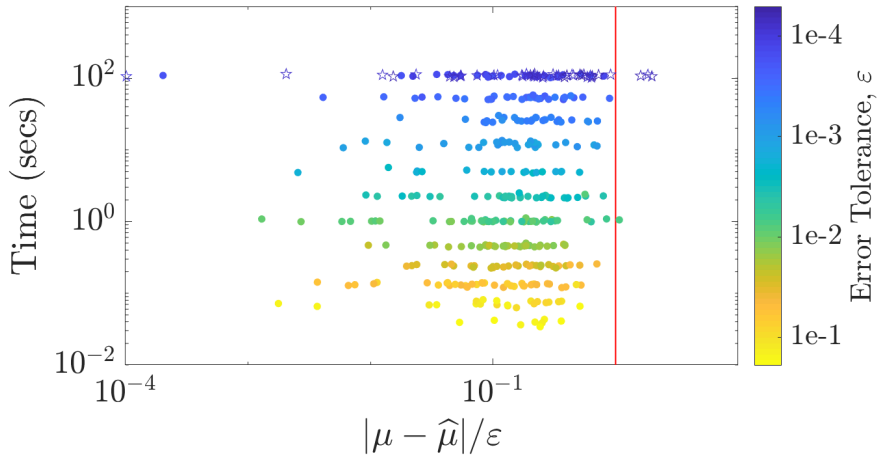

Fig. 12: Option pricing using the full-Bayes stopping criterion. The hollow stars indicate the algorithm has not met the error threshold ϵ even with using maximum n.

Most noticeable aspect from the plots of `cubBayesNet_g` is how closer the error bounds are to the true error. This shows that the `cubBayesNet_g`'s estimation of expected error in the stopping criterion is very accurate. Similar to `cubBayesLattice_g`, it missed meeting the given error threshold for the option pricing example, as marked by the hollow stars, for $\varepsilon = 10^{-4}$. The algorithm

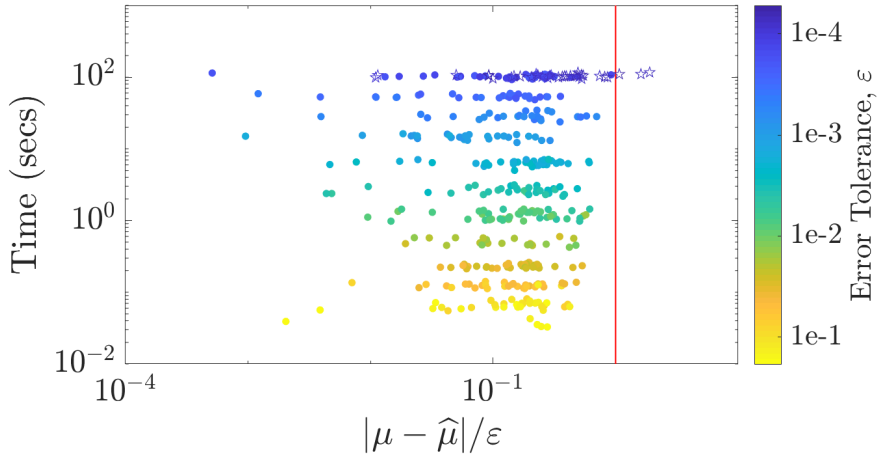

Fig. 13: Option pricing using the GCV stopping criterion. The hollow stars indicate the algorithm has not met the error threshold ϵ even with using maximum n.

reached max allowed number of samples, $n = 2^{20}$ due to the complexity of the integrand.

12.6 Comparison with `cubMC_g`, `cubLattice_g` and `cubSobol_g`

GAIL library provides variety of numerical integration algorithms based on different theoretical foundations, We would like to compare how our algorithms perform relatively to these. We consider three GAIL algorithms 1) `cubMC_g`, a simple Monte-Carlo method for multi-dimensional integration, 2) `cubLattice_g`, a quasi-Monte-Carlo method using Lattice points, and 3) `cubSobol_g`, a quasi-Monte-Carlo method using Sobol points.

Keister integral The Table 1 summarizes the performance of the methods MC, Lattice, Sobol, BayesLat, and BayesSob—which refer to the GAIL cubatures, `cubMC_g`, `cubLattice_g`, `cubSobol_g`, `cubBayesLattice_g`, `cubBayesNet_g`, respectively for estimating Keister integral defined in (17). We conducted two simulations with $d = 3$ and 8. In the case of $d = 3$, all five methods succeeded completely, meaning, the absolute error is less than given tolerance, i.e., $|\mu - \hat{\mu}| \leq \varepsilon$, where $\hat{\mu}$ is a cubature's approximated value. The fastest method was `cubBayesLattice_g`. In the case of $d = 8$, `cubSobol_g` achieved 100% success rate and was the fastest. But `cubBayesLattice_g` was competitive and had the smallest average absolute error. `cubBayesNet_g` used lowest number of samples but was slower than `cubSobol_g`.

JR: avoid Exp notation for 2 decimals

Multivariate Gaussian The Table 2 summarizes the performance of the methods MC, Lattice, Sobol, BayesLat, and BayesSob for estimating the multi-dimensional

Table 1: Comparison of average performance of cubatures for estimating the Keister integral (17) for 1000 independent runs. These results can be conditionally reproduced with the script, `KeisterCubatureExampleBayes.m`, in GAIL.

$d = 3,\ \varepsilon = 0.005$					
Method	MC	Lattice	Sobol	BayesLat	BayesSobol
Absolute Error	0.001 100	0.000 510	0.000 520	0.000 430	0.000 560
Tolerance Met	100%	100%	100%	100%	100%
n	2 500 000	4100	3900	1000	1900
Time (seconds)	0.1800	0.0069	0.0054	0.0029	0.0700
$d = 8,\ \varepsilon = 0.050$					
Method	MC	Lattice	Sobol	BayesLat	BayesSobol
Absolute Error	0.012 000	0.015 000	0.007 300	0.001 800	0.008 300
Tolerance Met	100%	99%	100%	100%	100%
n	7 400 000	15 000	16 000	66 000	8200
Time (seconds)	1.2000	0.0220	0.0160	0.2100	0.3500

Gaussian probability $\mathbf{X} \sim \mathbf{N}(\mu, \mathbf{\Sigma})$. This experiment demonstrates our algorithm's ability to handle high-dimensional integral.

We conducted two simulations with different Σ and estimation intervals $(\mathbf{a}, \mathbf{b})$ but fixed $\mu = 0$ and required error threshold, $\varepsilon = 10^{-3}$. In the first case, all five methods succeeded completely. The fastest method was `cubBayesLattice_g` but `cubBayesNet_g` used the lowest number of samples. In the second case also, all five methods succeeded, but `cubLattice_g` was the fastest. The `cubBayesNet_g` was competitive and had the smallest average absolute error using lowest number of samples. The `cubBayesLattice_g` achieved the next lowest average error but was slower than `cubSobol_g`.

Table 2: Comparison of average performance of cubatures for estimating the $d = 20$ Multivariate Normal (**??**) for 1000 independent runs with $\varepsilon = 10^{-3}$. These results can be conditionally reproduced with the script, `MVNCubatureExampleBayes.m`, in GAIL.

$\Sigma = \mathsf{I}_d,\ \boldsymbol{b} = -\boldsymbol{a} = (3.5, \cdots, 3.5)$					
Method	MC	Lattice	Sobol	BayesLat	BayesSobol
Absolute Error	2.20×10^{-16}	2.70×10^{-14}	2.70×10^{-14}	2.20×10^{-16}	2.20×10^{-16}
Tolerance Met	100%	100%	100%	100%	100%
n	10 000	1000	1000	1000	260
Time (seconds)	0.0410	0.0820	0.0710	0.0650	0.0790
$\Sigma = 0.4\ \mathsf{I}_d + 0.6\ \mathbf{1}\mathbf{1}^T,\ \boldsymbol{a} = (-\infty, \cdots, -\infty),\ \boldsymbol{b} = \sqrt{d}(U_1, \cdots, U_d)$					
Method	MC	Lattice	Sobol	BayesLat	BayesSobol
Absolute Error	2.30×10^{-4}	2.10×10^{-4}	4.40×10^{-4}	1.00×10^{-4}	4.80×10^{-5}
Tolerance Met	100%	100%	100%	100%	100%
n	10 000	1000	1000	1000	260
Time (seconds)	0.0350	0.0120	0.0140	0.0150	0.0300

Shape Parameter Fine-tuning JR: Numerical examples for the case of shape parameter per dimension

Allowing the kernel shape parameter to vary for each dimension could improve the accuracy of numerical integration when the integrand under consideration has only very low effective dimension as in the Option Pricing example we demonstrated. We demonstrate this advantage by integrating a function that is not symmetric across dimensions,

$$f(\boldsymbol{x}) = \sum_{j=1}^{d} \upsilon_j \sin(2\pi x_j^2) \tag{18}$$

which has known integral

$$\int_{[0,1)^d} f(\boldsymbol{x}) = \frac{1}{2}\,\texttt{fresnels}(d) \sum_{j=1}^{d} \upsilon_j$$

where `fresnels` is the Fresnel Sine integral,

$$\texttt{fresnels}(z) = \int_0^z \sin\left(\frac{\pi t^2}{2}\right) \mathrm{d}t.$$

Table 3: Comparison of average performance of Bayesian Cubature with common shape parameter vs dimension specific shape parameter for estimating the $d = 3$ Fresnel Sine integral. These results can be conditionally reproduced with the script, `demoMultiTheta.m`, in GAIL.

	Fresnel Sine Integral in $d = 3$	
Method	`OneTheta`	`MultiTheta`
Absolute Error	0.000 23	0.063 00
n	4100	260
Time (seconds)	0.0270	0.0230

The results are summarized from the two different approaches in Table 3. The first method, called `OneTheta`, uses common shape parameter across all the dimensions, whereas the second method, called `MultiTheta`, allows the shape parameters to vary across the dimensions. In the `MultiTheta` method, the shape parameter search is multivariate, so the magnitude of shape parameter depends on the integrand's magnitude in each dimension. We have chosen an integrand particularly to demonstrate this aspect (18) where we used $d = 3$ and the constants $\boldsymbol{\upsilon} = (10^{-4}, 1, 10^4)$. The choice of magnitude variations in constants $\boldsymbol{\upsilon}$ allows to make the integrand varies significantly across dimensions.

We ran this test for 1000 times. In comparison, both the methods successfully computed the integral all the time but `MultiTheta` was slightly faster. The `MultiTheta` method used less number of samples but the integration error was bigger than the `OneTheta`. For the same number of samples, the `OneTheta` method will be much faster since the shape parameter search is faster. The `MultiTheta` method is useful in scenarios where we want to use smaller size, n, and the integrand varies significantly across dimensions.

References

J. F. Baldeaux. *Higher order nets and sequences.* PhD thesis, The School of Mathematics and Statistics at The University of New South Wales, June 2010.

P. Bratley and B. L. Fox. Algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Trans. Math. Software*, 14:88–100, 1988.

R. Brent. *Algorithms for Minimization Without Derivatives.* Prentice-Hall, 1973.

F.-X. Briol, C. J. Oates, M. Girolami, M. A. Osborne, and D. Sejdinovic. Probabilistic integration: A role in statistical computation? *Statist. Sci.*, 2019. to appear.

S.-C. T. Choi, Y. Ding, F. J. Hickernell, L. Jiang, Ll. A. Jiménez Rugama, D. Li, R. Jagadeeswaran, X. Tong, K. Zhang, Y. Zhang, and X. Zhou. GAIL: Guaranteed Automatic Integration Library (versions 1.0–2.2). MATLAB software, 2013–2017. URL `http://gailgithub.github.io/GAIL_Dev/`.

P. Diaconis. Bayesian numerical analysis. In S. S. Gupta and J. O. Berger, editors, *Statistical Decision Theory and Related Topics IV, Papers from the 4th Purdue Symp., West Lafayette, Indiana 1986*, volume 1, pages 163–175. Springer-Verlag, New York, 1988.

J. Dick. Walsh spaces containing smooth functions an quasi-Monte Carlo rules of arbitrary high order. *SIAM J. Numer. Anal.*, 46(1519–1553), 2008.

K. Dong, D. Eriksson, H. Nickisch, D. Bindel, and A. G. Wilson. Scalable log determinants for gaussian process kernel learning. *NIPS*, 2017. in press.

G. Forsythe, M. Malcolm, and C. Moler. *Computer methods for mathematical computations.* Prentice-Hall, 1976.

F. J. Hickernell and R. X. Yue. The mean square discrepancy of scrambled (t, s)-sequences. *SIAM J. Numer. Anal.*, 38:1089–1112, 2000. doi: 10.1137/S0036142999358019.

H. S. Hong and F. J. Hickernell. Algorithm 823: Implementing scrambled digital nets. *ACM Trans. Math. Software*, 29:95–109, 2003. doi: 10.1145/779359.779360.

B. D. Keister. Multidimensional quadrature algorithms. *Computers in Physics*, 10:119–122, 1996. doi: 10.1063/1.168565.

F. Y. Kuo and D. Nuyens. Application of quasi-Monte Carlo methods to elliptic pdes with random diffusion coefficients a survey of analysis and implementation. *Foundations of Computational Mathematics*, 16(6):1631–1696, 2016.

J. Matoušek. On the L_2-discrepancy for anchored boxes. *J. Complexity*, 14:527–556, 1998.

H. Niederreiter. Constructions of (t, m, s)-nets and (t, s)-sequences. *Finite Fields Appl.*, 11:578–600, 2005.

D. Nuyens. URL `https://people.cs.kuleuven.be/~dirk.nuyens/`.

D. Nuyens. The construction of good lattice rules and polynomial lattice rules. Aug 2013.

A. OHagan. Bayes-hermite quadrature. *Journal of Statistical Planning and Inference*, 29(3):245260, 1991.

A. B. Owen. Randomly permuted (t, m, s)-nets and (t, s)-sequences. In H. Niederreiter and P. J.-S. Shiue, editors, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, volume 106 of *Lecture Notes in Statistics*, pages 299–317. Springer-Verlag, New York, 1995.

C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning.* MIT Press, Cambridge, Massachusetts, 2006. (online version available

at `http://www.gaussianprocess.org/gpml/`).

I. M. Sobol'. The distribution of points in a cube and the approximate evaluation of integrals. *U.S.S.R. Comput. Math. and Math. Phys.*, 7:86–112, 1967.

I. M. Sobol'. Uniformly distributed sequences with an additional uniformity property. *Zh. Vychisl. Mat. i Mat. Fiz.*, 16:1332–1337, 1976.