Noname manuscript No.
(will be inserted by the editor)

Fast Automatic Bayesian Cubature Using Lattice Sampling

R. Jagadeeswaran · Fred. J. Hickernell

Received: date / Accepted: date

Abstract Automatic cubatures provide approximations to multidimensional integrals that satisfy user-specified error tolerances. For multidimensional problems, the sampling density is fixed, but the sample size, n, is determined automatically. Bayesian cubature postulates that the integrand is an instance of a stochastic process. Prior information about mean and covariance of this process is used to form data-driven error bounds. However, the process of inferring the mean and covariance governing the stochastic process from n integrand values involves computing matrix inverses and determinants, which are in general time-consuming $O(n^3)$ operations. Our work employs low discrepancy data sites and matching kernels that lower the computational cost to $O(n \log n)$. The confidence interval for the Bayesian posterior error is used to choose n automatically to satisfy the user-defined error tolerance. This approach is demonstrated using rank-1 lattice sequences and shift-invariant kernels.

Keywords Bayesian cubature · Probabilistic numeric methods · GAIL

1 Introduction

Cubature is the problem of inferring a numerical value for the integral $\mu := \int_{\mathbb{R}^d} g(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$ where μ has no closed form analytic expression. Typically, the multivariate integrand g is only accessible in the form of a black-box function routine. Cubature is a key component of many problems in scientific computing, finance, statistical modeling, and machine learning.

The integral may often be expressed as

$$\mu := \mu(f) := \mathbb{E}[f(\boldsymbol{X})] = \int_{[0,1]^d} f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}, \tag{1}$$

where $f : [0,1]^d \to \mathbb{R}$ is the integrand, and $\boldsymbol{X} \sim \mathcal{U}[0,1]^d$. The process of transforming the original integral into the form of (1) is not addressed here. The cubature algorithm may take the form

$$\widehat{\mu} := \widehat{\mu}(f) := w_0 + \sum_{i=1}^{n} f(\boldsymbol{x}_i) w_i, \tag{2}$$

where the weights, w_0, and $\boldsymbol{w} = (w_i)_{i=1}^n \in \mathbb{R}^n$, and the nodes, $\{\boldsymbol{x}_i\}_{i=1}^n \subset [0,1]^d$, are chosen to make the error, $|\mu - \widehat{\mu}|$, small. The integration domain $[0,1]^d$ is convenient for the low discrepancy node sets that we use. The nodes are assumed to be deterministic.

We will construct a reliable stopping criterion that determines the number of integrand values required, n, to ensure that the error is no greater than a user-defined error tolerance, i.e.,

$$|\mu - \widehat{\mu}| \le \varepsilon \tag{3}$$

Rather than relying on strong assumptions about the integrand, such as its variance or total variation, we construct a stopping criterion that is data-driven, and based on a credible interval arising from a Bayesian approach to the problem. We build upon the ideas of Briol et al. [1], Diaconis [3], O'Hagan [12], Ritter [16], Rasmussen and Ghahramani [14], and others. Our algorithm is an example of *probabilistic numerics*.

Our primary contribution here is to demonstrate how the choice of a family of covariance kernels that match the low discrepancy sampling nodes facilitates

F. Author
first address
Tel.: +123-45-678910
Fax: +123-45-678910
E-mail: fauthor@example.com

Fred J. Hickernell
Center for Computational Science and Department of Applied Mathematics, Illinois Institute of Technology
PS 106, 3105 S. Dearborn St., Chicago IL 60616
E-mail: hickernell@iit.edu

fast computation of the cubature and the data-driven stopping criterion. Our cubature requires n function values—at a cost of \$(f) each—plus $\mathcal{O}(n\log(n))$ operations to check whether the error tolerance is satisfied. The total cost of our algorithm is then $\mathcal{O}(n[\$(f)+\log(n)])$. This is significantly fewer operations than the $\mathcal{O}(n^3)$ typically required for Bayesian cubature. If function evaluation is expensive, then \$(f) might be similar in magnitude to $\log(n)$.

Hickernell [7] compares different approaches to cubature error analysis depending on whether the rule is deterministic or random and whether the integrand is assumed to be deterministic or random. Error analyis that assumes a deterministic integrand lying in a Banach space, leads to an error bound that is typically impractical for deciding how large n must be to satisfy (3). The deterministic error bound includes a (semi-) norm of the integrand, which is often more complex to compute than the original integral.

Hickernell and Jiménez-Rugama [8,10] have developed stopping criteria for cubature rules based on low discrepancy nodes by tracking the discrete Fourier coefficients of the integrand. The algorithm proposed here also depends on the discrete Fourier coefficients of the integrand, but in a different way.

Section 2 explains the Bayesian approach to estimate the posterior cubature error and defines our automatic Bayesian cubature . Although much of this material is not new, it is included for completeness. We end Section 2 by demonstrating why Bayesian cubature is typically quite computationally expensive. Section 3 introduces the concept of covariance kernels that match well the nodes and expedites the computations required by our automatic Bayesian cubature. Section 4 implements this idea for shift invariant kernels and rank-1 lattice nodes. Section 5 covers further enhancements to the algorithm to avoid cancellation error and make it faster. Finally, numerical examples are shown using the faster algorithm developed.

2 Bayesian cubature

2.1 Bayesian posterior error

Suppose that the integrand, f, is drawn from a Gaussian process, i.e., $f \sim \mathcal{GP}(m, s^2 C_{\boldsymbol{\theta}})$. Specifically, f has real-valued mean m and covariance function $s^2 C_{\boldsymbol{\theta}}$, where s is a non-negative scale factor s $C_{\boldsymbol{\theta}} : [0,1]^d \times [0,1]^d \to \mathbb{R}$ is a symmetric, positive-definite function and parameterized by $\boldsymbol{\theta}$:

$$\mathsf{C}^T = \mathsf{C}, \qquad \boldsymbol{a}^T \mathsf{C} \boldsymbol{a} > 0 \quad \forall \boldsymbol{a} \neq \boldsymbol{0},$$

where $\mathsf{C} = (C_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j=1}^n$,

$$\forall n \in \mathbb{N},\ \boldsymbol{t}, \boldsymbol{x}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in [0,1]^d. \quad (4)$$

The covariance function, C, and the Gram matrix, C depend implicitly on $\boldsymbol{\theta}$, but sometimes this is omitted in the notation for simplicity's sake.

For a Gaussian process, all vectors of linear functionals of f have a multivariate Gaussian distribution. Defining $\boldsymbol{f} := (f(\boldsymbol{x}_i))_{i=1}^n$ as the multivariate normal vector of function values, it follows that

$$\boldsymbol{f} \sim \mathcal{N}(m\boldsymbol{1}, s^2\mathsf{C}), \quad (5a)$$

$$\mu \sim \mathcal{N}(m, s^2 c_0), \quad (5b)$$

$$\text{where } c_0 = \int_{[0,1]^{2d}} C_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{t})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{t}, \quad (5c)$$

$$\mathrm{cov}(\boldsymbol{f}, \mu) = \left(\int_{[0,1]^d} C(\boldsymbol{t}, \boldsymbol{x}_i)\, \mathrm{d}\boldsymbol{t} \right)_{i=1}^n =: \boldsymbol{c}. \quad (5d)$$

We need the following Lemma 1 to derive the posterior.

Lemma 1 *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

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

Inverse of the matrix C *may be partitioned as [15] eqns (A.11, A.12 and A.13):*

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

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

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

For more details of the proof refer to [15] eqn (A.6).

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

$$\mu | (\boldsymbol{f} = \boldsymbol{y}) \sim \mathcal{N}\big(m(1 - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{1}) + \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{y}, \ s^2(c_0 - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c})\big). \quad (6)$$

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

$$\widehat{\mu}|(\boldsymbol{f} = \boldsymbol{y}) = m(1 - \boldsymbol{1}^T\mathsf{C}^{-1}\boldsymbol{c}) + \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{y}. \quad (7)$$

Under this definition, the cubature error has zero mean and a variance depending on the choice of nodes:

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

A credible interval for the integral is given given by

$$\mathbb{P}_f\left[|\mu - \widehat{\mu}| \le 2.58 s\sqrt{c_0 - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c}}\right] = 99\%. \quad (8)$$

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

2.2 Parameter estimation

The credible interval in (8) suggests how our automatic Bayesian cubature proceeds. Function data is accumulated until we can be confident that the error is no greater than the error tolerance. However, the credible interval still depends on the parameters m, s, and $\boldsymbol{\theta}$. These should not be assumed a priori but be inferred from the data. This section describes three approaches.

2.2.1 Empirical Bayes

One approach to estimate the parameters is via maximum likelihood estimation (MLE). The log-likelihood function of the parameters given the function data $\boldsymbol{y}$ is:

$$l(s,m,\boldsymbol{\theta}|\boldsymbol{y}) = -\frac{1}{2}s^{-2}(\boldsymbol{y}-m\mathbf{1})^T\mathsf{C}^{-1}(\boldsymbol{y}-m\mathbf{1}) - \frac{1}{2}\log(\det\,\mathsf{C}) - \frac{n}{2}\log(s^2) + \text{constants}. \quad (9)$$

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

$$m_{\text{MLE}} = \frac{\mathbf{1}^T\mathsf{C}^{-1}\boldsymbol{y}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}} \quad (10)$$

$$s^2_{\text{MLE}} = \frac{1}{n}(\boldsymbol{y}-m_{\text{MLE}}\mathbf{1})^T\mathsf{C}^{-1}(\boldsymbol{y}-m_{\text{MLE}}\mathbf{1}) = \frac{1}{n}\boldsymbol{y}^T\left[\mathsf{C}^{-1}-\frac{\mathsf{C}^{-1}\mathbf{1}\mathbf{1}^T\mathsf{C}^{-1}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}\right]\boldsymbol{y}, \quad (11)$$

$$\boldsymbol{\theta}_{\text{MLE}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left\{\log\left(\boldsymbol{y}^T\left[\mathsf{C}^{-1}-\frac{\mathsf{C}^{-1}\mathbf{1}\mathbf{1}^T\mathsf{C}^{-1}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}\right]\boldsymbol{y}\right) + \frac{1}{n}\log(\det(\mathsf{C}))\right\}. \quad (12)$$

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

Under these estimates of the parameters, the cubature (7) and the credible interval (8) simplify to

$$\widehat{\mu}_{\text{MLE}} = \left(\frac{(1-\mathbf{1}^T\mathsf{C}^{-1}\boldsymbol{c})\mathbf{1}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}+\boldsymbol{c}\right)^T\mathsf{C}^{-1}\boldsymbol{y}, \quad (13)$$

$$\text{err}^2_{\text{MLE}} := \frac{2.58^2}{n}\boldsymbol{y}^T\left[\mathsf{C}^{-1}-\frac{\mathsf{C}^{-1}\mathbf{1}\mathbf{1}^T\mathsf{C}^{-1}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}\right]\boldsymbol{y} \quad (14)$$

$$\times\,(c_0-\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c}), \quad (15)$$

$$\mathbb{P}_f\left[|\mu-\widehat{\mu}_{\text{MLE}}|\le \text{err}_{\text{MLE}}\right] = 99\%. \quad (16)$$

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

2.2.2 Full Bayes

Rather than use maximum likelihood to determine m and s one treat them as hyper-parameters and may a non-informative, conjugate prior, namely $\boldsymbol{\rho}_{m,s^2}(\xi,\lambda)\propto 1/\lambda$. Then the posterior density for the integral given the data using Bayes theorem is

$$\begin{aligned}
&\rho_\mu(z|\boldsymbol{f}=\boldsymbol{y})\\
&\quad\propto\int_0^\infty\int_{-\infty}^\infty \rho_\mu(z|\boldsymbol{f}=\boldsymbol{y},m=\xi,s^2=\lambda)\\
&\qquad\times\rho_{\boldsymbol{f}}(\boldsymbol{y}|\xi,\lambda)\rho_{m,s^2}(\xi,\lambda)\,\mathrm{d}\xi\mathrm{d}\lambda\\
&\quad\propto\int_0^\infty\frac{1}{\lambda^{(n+3)/2}}\\
&\quad\times\int_{-\infty}^\infty\exp\Bigg(-\frac{1}{2\lambda}\Bigg\{\frac{[z-\xi(1-\boldsymbol{c}^T\mathsf{C}^{-1}\mathbf{1})-\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{y}]^2}{c_0-\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c}}\\
&\qquad+(\boldsymbol{y}-\xi\mathbf{1})^T\mathsf{C}^{-1}(\boldsymbol{y}-\xi\mathbf{1})\Bigg\}\Bigg)\,\mathrm{d}\xi\mathrm{d}\lambda\\
&\qquad\text{by (5a), (6) and } \rho_{m,s^2}(\xi,\lambda)\propto 1/\lambda\\
&\quad\propto\int_0^\infty\frac{1}{\lambda^{(n+3)/2}}\cdots\\
&\qquad\cdots\int_{-\infty}^\infty\exp\left(-\frac{\alpha\xi^2-2\beta\xi+\gamma}{2\lambda(c_0-\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c})}\right)\mathrm{d}\xi\mathrm{d}\lambda,
\end{aligned}$$

where

$$\begin{aligned}
\alpha &= (1-\boldsymbol{c}^T\mathsf{C}^{-1}\mathbf{1})^2+\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}(c_0-\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c}),\\
\beta &= (1-\boldsymbol{c}^T\mathsf{C}^{-1}\mathbf{1})(z-\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{y})\\
&\quad+\mathbf{1}^T\mathsf{C}^{-1}\boldsymbol{y}(c_0-\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c}),\\
\gamma &= (z-\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{y})^2+\boldsymbol{y}^T\mathsf{C}^{-1}\boldsymbol{y}(c_0-\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c}).
\end{aligned}$$

In the derivation above and below, factors that are independent of ξ, λ, or z can be discarded since we only need to preserve the proportion, however, factors that depend on ξ, λ, or z must be kept. Completing the square $\alpha\xi^2-2\beta\xi+\gamma=\alpha(\xi-\beta/\alpha)^2-(\beta^2/\alpha)+\gamma$, allows us to evaluate the integrals with respect to ξ and λ:

$$\begin{aligned}
&\rho_\mu(z|\boldsymbol{f}=\boldsymbol{y})\\
&\quad\propto\int_0^\infty\frac{1}{\lambda^{(n+3)/2}}\exp\left(-\frac{\gamma-\beta^2/\alpha}{2\lambda(c_0-\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c})}\right)\cdots\\
&\qquad\cdots\int_{-\infty}^\infty\exp\left(-\frac{\alpha(\xi-\beta/\alpha)^2}{2\lambda(c_0-\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c})}\right)\mathrm{d}\xi\mathrm{d}\lambda\\
&\quad\propto\int_0^\infty\frac{1}{\lambda^{(n+2)/2}}\exp\left(-\frac{\gamma-\beta^2/\alpha}{2\lambda(c_0-\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c})}\right)\mathrm{d}\lambda\\
&\quad\propto\left(\gamma-\frac{\beta^2}{\alpha}\right)^{-n/2}\propto\left(\alpha\gamma-\beta^2\right)^{n/2}.
\end{aligned}$$

Finally, we simplify the key term via straightforward calculations to the following:

$$\alpha\gamma-\beta^2\propto 1+\frac{(z-\widehat{\mu}_{\text{MLE}})^2}{(n-1)\widehat{\sigma}^2_{\text{full}}},$$

where

$$\widehat{\sigma}^2_{\text{full}} := \frac{1}{n-1}\boldsymbol{y}^T\left[\mathsf{C}^{-1}-\frac{\mathsf{C}^{-1}\mathbf{1}\mathbf{1}^T\mathsf{C}^{-1}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}\right]\boldsymbol{y} \times\left[\frac{(1-\boldsymbol{c}^T\mathsf{C}^{-1}\mathbf{1})^2}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}+(c_0-\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c})\right]. \quad (17)$$

This means that $\mu|(\boldsymbol{f}=\boldsymbol{y})$, properly centered and scaled, has a Student's t-distribution with $n-1$ degrees of freedom. The estimated integral is the same as in the empirical Bayes case, but the confidence interval is wider:

$$\mathbb{P}_f\left[|\mu-\widehat{\mu}_{\mathrm{MLE}}|\le \mathrm{err}_{\mathrm{full}}\right]=99\%, \tag{18}$$

where

$$\mathrm{err}_{\mathrm{full}} := t_{n_j-1,0.995}\widehat{\sigma}_{\mathrm{full}} > \mathrm{err}_{\mathrm{MLE}}. \tag{19}$$

Here $t_{n-1,0.995}$ denotes the 99.5 percentile of a standard Student's t-distribution with $n-1$ degrees of freedom.

Because the shape parameter, $\boldsymbol{\theta}$, enters the definition of the covariance kernel in a non-trivial way, the only way to treat it as a hyperparameter and assign a tractable prior would be for the prior to be discrete. We believe in practice that choosing such a prior involves more guesswork than using the empirical Bayes estimate of $\boldsymbol{\theta}$ in (12) or the cross-validation approached described next.

2.2.3 Generalized Cross-validation

A third parameter optimization is *leave-one-out cross-validation*. Let $\widetilde{y}_i = \mathbb{E}[f(\boldsymbol{x}_i)|\boldsymbol{f}_{-i}=\boldsymbol{y}_{-i}]$, where the subscript $-i$ denotes the vector excluding the i^{th} component. This is the conditional expectation of $f(\boldsymbol{x}_i)$ given all data but the function value at $\boldsymbol{x}_i$. The cross-validation criterion, which is to be minimized, is sum of squares of the difference between these conditional expectations and the observed values:

$$\mathrm{CV}=\sum_{i=1}^{n}(y_i-\widetilde{y}_i)^2. \tag{20}$$

Let $\mathsf{A}=\mathsf{C}^{-1}$, let $\boldsymbol{\zeta}=\mathsf{A}(\boldsymbol{y}-m\boldsymbol{1})$, and partition C, A, and $\boldsymbol{\zeta}$ as

$$\mathsf{C}=\begin{pmatrix} c_{ii} & \boldsymbol{C}_{-i,i}^T \\ \boldsymbol{C}_{-i,i} & \mathsf{C}_{-i,-i}\end{pmatrix}, \qquad \mathsf{A}=\begin{pmatrix} a_{ii} & \boldsymbol{A}_{-i,i}^T \\ \boldsymbol{A}_{-i,i} & \mathsf{A}_{-i,-i}\end{pmatrix},$$

$$\boldsymbol{\zeta}=\begin{pmatrix}\zeta_i\\ \boldsymbol{\zeta}_{-i}\end{pmatrix},$$

where the subscript i denotes the i^{th} row or column, and the subscript $-i$ denotes all rows or columns except the i^{th}. Following this notation, Lemma 1 implies that

$$\begin{aligned}\widetilde{y}_i &= m+\boldsymbol{C}_{-i,i}^T\mathsf{C}_{-i,-i}^{-1}(\boldsymbol{y}_{-i}-m\boldsymbol{1})\\ \zeta_i &= a_{ii}(y_i-m)+\boldsymbol{A}_{-i,i}^T(\boldsymbol{y}_{-i}-m\boldsymbol{1})\\ &= a_{ii}[(y_i-m)-\boldsymbol{C}_{-i,i}^T\mathsf{C}_{-i,-i}^{-1}(\boldsymbol{y}_{-i}-m\boldsymbol{1})]\\ &= a_{ii}(y_i-\widetilde{y}_i).\end{aligned}$$

Thus, (20) may be re-written as

$$\mathrm{CV}=\sum_{i=1}^{n}\left(\frac{\zeta_i}{a_{ii}}\right)^2, \qquad \boldsymbol{\zeta}=\mathsf{C}^{-1}(\boldsymbol{y}-m\boldsymbol{1}).$$

Wahba [17] recommended a generalized cross-validation criterion, which replaces the i^{th} diagonal element of A in the denominator by the average diagonal element of A:

$$\begin{aligned}\mathrm{GCV} &= \frac{\sum_{i=1}^n \zeta_i^2}{\left(\frac{1}{n}\sum_{i=1}^n a_{ii}\right)^2}\\ &= \frac{(\boldsymbol{y}-m\boldsymbol{1})^T\mathsf{C}^{-2}(\boldsymbol{y}-m\boldsymbol{1})}{\left(\frac{1}{n}\mathrm{trace}(\mathsf{C}^{-1})\right)^2}.\end{aligned} \tag{21}$$

The loss function GCV depends on m and $\boldsymbol{\theta}$, but not on s. Minimizing the GCV yields

$$m_{\mathrm{GCV}}=\frac{\boldsymbol{1}^T\mathsf{C}^{-2}\boldsymbol{y}}{\boldsymbol{1}^T\mathsf{C}^{-2}\boldsymbol{1}}, \tag{22}$$

$$\begin{aligned}\boldsymbol{\theta}_{\mathrm{GCV}}=\operatorname*{argmin}_{\boldsymbol{\theta}}\Bigg\{&\log\left(\boldsymbol{y}^T\left[\mathsf{C}^{-2}-\frac{\mathsf{C}^{-2}\boldsymbol{1}\boldsymbol{1}^T\mathsf{C}^{-2}}{\boldsymbol{1}^T\mathsf{C}^{-2}\boldsymbol{1}}\right]\boldsymbol{y}\right)\\ &-2\log\left(\mathrm{trace}(\mathsf{C}^{-1})\right)\Bigg\}.\end{aligned} \tag{23}$$

Plugging this value of m into (7) yields

$$\widehat{\mu}_{\mathrm{GCV}}=\left(\frac{(1-\boldsymbol{1}^T\mathsf{C}^{-1}\boldsymbol{c})\mathsf{C}^{-1}\boldsymbol{1}}{\boldsymbol{1}^T\mathsf{C}^{-2}\boldsymbol{1}}+\boldsymbol{c}\right)^T\mathsf{C}^{-1}\boldsymbol{y}. \tag{24}$$

An estimate for s may be obtained by noting that by Lemma 1,

$$\mathrm{var}[f(\boldsymbol{x}_i)|\boldsymbol{f}_{-i}=\boldsymbol{y}_{-i}]=s^2a_{ii}^{-1},$$

Thus, we may estimate s using an argument similar to that used in deriving the GCV and then substituting m_{GCV} for m:

$$\begin{aligned}s^2 &= \mathrm{var}[f(\boldsymbol{x}_i)|\boldsymbol{f}_{-i}=\boldsymbol{y}_{-i}]a_{ii}\\ &\approx \frac{1}{n}\sum_{i=1}^n(y_i-\widetilde{y}_i)^2a_{ii}=\frac{1}{n}\sum_{i=1}^n\frac{\zeta_i^2}{a_{ii}}\\ &\approx \frac{\frac{1}{n}\sum_{i=1}^n\zeta_i^2}{\frac{1}{n}\sum_{i=1}^n a_{ii}}=\frac{(\boldsymbol{y}-m\boldsymbol{1})^T\mathsf{C}^{-2}(\boldsymbol{y}-m\boldsymbol{1})}{\mathrm{trace}(\mathsf{C}^{-1})}\\ &\approx s_{\mathrm{GCV}}^2\end{aligned}$$

where

$$\begin{aligned}s_{\mathrm{GCV}}^2 := \boldsymbol{y}^T&\left[\mathsf{C}^{-2}-\frac{\mathsf{C}^{-2}\boldsymbol{1}\boldsymbol{1}^T\mathsf{C}^{-2}}{\boldsymbol{1}^T\mathsf{C}^{-2}\boldsymbol{1}}\right]\boldsymbol{y}\\ &\times\left[\mathrm{trace}(\mathsf{C}^{-1})\right]^{-1}\end{aligned} \tag{25}$$

The confidence interval based on generalized cross-validation corresponds to (8) with the GCV estimates for m, s, and $\boldsymbol{\theta}$:

$$\mathrm{err}_{\mathrm{GCV}}=2.58s_{\mathrm{GCV}}\sqrt{c_0-\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c}} \tag{26}$$

$$\mathbb{P}_f\left[|\mu-\widehat{\mu}_{\mathrm{GCV}}|\le \mathrm{err}_{\mathrm{GCV}}\right]=99\%. \tag{27}$$

Looking back over the results of Sections 2.2.1–2.2.3, it is noted that if the original covariance function, C, is replaced by $b\mathsf{C}$ for some positive constant b, the cubature, $\widehat{\mu}$, the estimates of $\boldsymbol{\theta}$, and the credible interval widths, err, all remain unchanged. The estimates of s^2 are multiplied by b^{-1}, as would be expected.

2.3 The automatic Bayesian cubature algorithm

The previous section presents three credible intervals, (16), (18), and (27), for the μ, the desired integral. Each credible interval is based on different assumptions

about the hyperparameters m, s, and $\boldsymbol{\theta}$. We stress that one must estimate these hyperparameters or assume a prior distribution on them because the credible intervals are used as stopping criteria for our cubature rule. Since a credible intervals makes a statement about a typical function—not an outlier—one must try to ensure that the integrand is a typical draw from the assumed Gaussian process.

Our Bayesian cubature algorithm increases the sample size until the width of the credible interval is small enough. This is accomplished through successfully doubling the sample size. The steps are detailed in Algorithm 1.

Algorithm 1 Automatic Bayesian Cubature

Require: a generator for the sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$; a black-box function, f; an absolute error tolerance, $\varepsilon > 0$; the positive initial sample size, n_0

1: $n \leftarrow n_0$, $n' \leftarrow 0$, err $= \infty$
2: **while** err $> \varepsilon$ **do**
3: **if** $n' > 0$ **then** $n' \leftarrow n$, $n \leftarrow 2n'$
4: **end if**
5: Generate $\{\boldsymbol{x}_i\}_{i=n'+1}^{n}$ and sample $\{f(\boldsymbol{x}_i)\}_{i=n'+1}^{n}$
6: Compute $\boldsymbol{\theta}$ by (12) or (23)
7: Compute err, the width of the credible interval, according to (14), (19), or (26)
8: **end while**
9: Compute $\widehat{\mu}$, the approximate integral, according to (10) or (24)
10: **return** $\widehat{\mu}$ and err

2.4 Example with the Matern kernel

FJH: What is your integrand? d? Does θ really depend on k? We would like to demonstrate and test numerical accuracy and computational cost of the automatic Bayesian cubature algorithm using the results obtained so far. For this example, we use the Matern kernel:

$$C_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{t}) = \prod_{k=1}^{d} \exp(-\theta_k |\boldsymbol{x}_k - \boldsymbol{t}_k|)(1 + \theta_k |\boldsymbol{x}_k - \boldsymbol{t}_k|) \tag{28}$$

On our test computer with Intel i7 3630QM and 16GB RAM memory, it took 118 minutes to compute $\widehat{\mu}_n$ with $n = 2^{13}$. As shown in Figure 1, computation time increases rapidly with n. Especially, Maximum-likelihood-estimation of $\boldsymbol{\theta}$ which needs the loss function, is the most time consuming of all. Because the loss functions need to be computed multiple times in every iteration to choose the optimal shape parameter till its minimum is found. Not only the complexity increases, also the kernel matrix becomes highly ill-conditioned with increasing n number of data-points. We must use

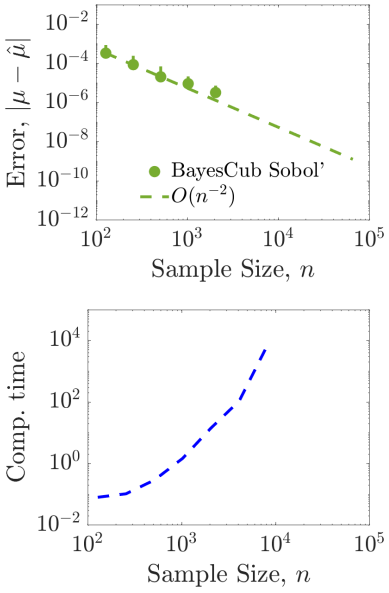

Fig. 1: MVN for d=2 with Matern kernel

alternative techniques to overcome this problem. So, our algorithm in the current form is not straightaway usable for any practical applications.

3 Fast Automatic Bayesian Cubature

The generic automatic Bayesian cubature algorithm described in the previous section requires $\mathcal{O}(n^3)$ operations to estimate $\boldsymbol{\theta}$, compute the credible interval width, and compute the cubature. There is also the potential ill-condtioning that can arise. This section explains how speed up the calculations and avoid ill-conditioning. A key is to choose kernels that match the design, $\{\boldsymbol{x}_i\}_{i=1}^{n}$, so that the vector-matrix operations required by Bayesian cubature can be accomplished using fast transforms at a cost of $\mathcal{O}(n \log(n))$.

3.1 Fast transform kernel

We make some assumptions about the relationship between the covariance kernel and the design, which will be shown to hold in Section 4 for rank-1 lattices and shift-invariant kernels. First we introduced the notation

$$\begin{aligned} \mathsf{C} &= \left(C_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)_{i,j=1}^{n} = (\boldsymbol{C}_1, \ldots, \boldsymbol{C}_n) \\ &= \frac{1}{n}\mathsf{V}\Lambda\mathsf{V}^H = \frac{1}{n}\mathsf{V}^*\Lambda\mathsf{V}^T, \qquad \mathsf{V}^H = n\mathsf{V}^{-1}, \\ \mathsf{C}^{-1} &= \frac{1}{n}\mathsf{V}\Lambda^{-1}\mathsf{V}^H = \frac{1}{n}\mathsf{V}^*\Lambda^{-1}\mathsf{V}^T, \end{aligned} \tag{29}$$

where V^H is the Hermitian of V. The columns of matrix V are eigenvectors of C, and Λ is a diagonal matrix of

eigenvalues of C.
$\mathsf{V} = (\boldsymbol{V}_1, ..., \boldsymbol{V}_n) = (\boldsymbol{v}_1, ..., \boldsymbol{v}_n)^T$, also $\boldsymbol{v}_1 = \boldsymbol{V}_1 = \boldsymbol{1}$
$\Lambda = \operatorname{diag}(\boldsymbol{\lambda}), \quad \boldsymbol{\lambda} = (\lambda_1, ...\lambda_n)$,
$\mathsf{V}^{-1} = \frac{1}{n}\mathsf{V}^H$, $\mathsf{C}^{-1} = \frac{1}{n}\mathsf{V}\Lambda^{-1}\mathsf{V}^H = \frac{1}{n}\mathsf{V}^*\Lambda^{-1}\mathsf{V}^T$.
For any $n \times n$ vector $\boldsymbol{b}$, define the notation $\widetilde{\boldsymbol{b}} := \mathsf{V}^T\boldsymbol{b}$.

We make three assumptions that allow the fast computation:
V may be identified analytically, (30a)
$\boldsymbol{v}_1 = \boldsymbol{V}_1 = \boldsymbol{1}$, (30b)
$\mathsf{V}^T\boldsymbol{b}$ requires only $\mathcal{O}(n\log(n))$ operations $\forall \boldsymbol{b}$. (30c)
We call the transformation $\boldsymbol{z} \mapsto \mathsf{V}^T\boldsymbol{z}$ a *fast transform.*

Under assumptions (30) the eigenvalues may be identified as the fast transform of the first column of C:

$$\boldsymbol{\lambda} = \begin{pmatrix}\lambda_1\\ \vdots \\ \lambda_n\end{pmatrix} = \Lambda\boldsymbol{1} = \Lambda\boldsymbol{v}_1 = \underbrace{\left(\frac{1}{n}\mathsf{V}^T\mathsf{V}^*\right)}\Lambda\boldsymbol{v}_1$$

$$= \mathsf{V}^T\left(\frac{1}{n}\mathsf{V}^*\Lambda\boldsymbol{v}_1\right) = \mathsf{V}^T\boldsymbol{C}_1 = \widetilde{\boldsymbol{C}}_1. \tag{31}$$

Also note that the fast transform of $\boldsymbol{1}$ has a simple form

$$\widetilde{\boldsymbol{1}} = \mathsf{V}^T\boldsymbol{1} = \mathsf{V}^T\boldsymbol{V}_1^* = \begin{pmatrix} n\\ 0\\ \vdots \\ 0\end{pmatrix}. \tag{32}$$

Many of the terms that arise in the calculations in Algorithm 1 are of the form $\boldsymbol{a}^T\mathsf{C}^p\boldsymbol{b}$ for integer p. These can be calculated via the transforms $\widetilde{\boldsymbol{a}} = \mathsf{V}^T\boldsymbol{a}$ and $\widetilde{\boldsymbol{b}} = \mathsf{V}^T\boldsymbol{b}$ as

$$\boldsymbol{a}^T\mathsf{C}^p\boldsymbol{b} = \frac{1}{n}\boldsymbol{a}^T\mathsf{V}\Lambda^p\mathsf{V}^H\boldsymbol{b} = \frac{1}{n}\widetilde{\boldsymbol{a}}^T\Lambda^p\widetilde{\boldsymbol{b}}^* = \frac{1}{n}\sum_{i=1}^n \lambda_i^p\widetilde{a}_i\widetilde{b}_i^*,$$

In particular,

$$\boldsymbol{1}^T\mathsf{C}^{-p}\boldsymbol{1} = \frac{n}{\lambda_1^p}, \qquad \boldsymbol{1}^T\mathsf{C}^{-p}\boldsymbol{y} = \frac{\widetilde{y}_1}{\lambda_1^p},$$

$$\boldsymbol{y}^T\mathsf{C}^{-p}\boldsymbol{y} = \frac{1}{n}\sum_{i=1}^n \frac{|\widetilde{y}_i|^2}{\lambda_i^p}, \qquad \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{1} = \frac{\widetilde{c}_1}{\lambda_1},$$

$$\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{y} = \frac{1}{n}\sum_{i=1}^n \frac{\widetilde{c}_i\widetilde{y}_i^*}{\lambda_i}, \qquad \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c} = \frac{1}{n}\sum_{i=1}^n \frac{|\widetilde{c}_i|^2}{\lambda_i},$$

where $\widetilde{\boldsymbol{y}} = \mathsf{V}^T\boldsymbol{y}$ and $\widetilde{\boldsymbol{c}} = \mathsf{V}^T\boldsymbol{c}$.

The covariance kernel used in practice also may satisfy an additional assumption:

$$\int_{[0,1]^d} C(\boldsymbol{t}, \boldsymbol{x})\,\mathrm{d}\boldsymbol{t} = 1 \qquad \forall \boldsymbol{x} \in [0,1]^d, \tag{33}$$

which implies that $c_0 = 1$ and $\boldsymbol{c} = \boldsymbol{1}$. Under (33), the expressions above may be further simplified:
$\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{1} = \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c} = \frac{n}{\lambda_1}$.

3.2 Empirical Bayes

Under assumptions (30), the empirical Bayes parameters in (10), (11), (12) (13), and (14) can be expressed in terms of the fast transforms of the function data, the first column of the Gram matrix, and $\boldsymbol{c}$ as follows:

$$m_{\mathrm{MLE}} = \frac{\widetilde{y}_1}{n} = \frac{1}{n}\sum_{i=1}^n y_i,$$

$$s^2_{\mathrm{MLE}} = \frac{1}{n^2}\sum_{i=2}^n \frac{|\widetilde{y}_i|^2}{\lambda_i}$$

$$\boldsymbol{\theta}_{\mathrm{MLE}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\log\left(\sum_{i=2}^n \frac{|\widetilde{y}_i|^2}{\lambda_i}\right) + \frac{1}{n}\sum_{i=1}^n \log(\lambda_i)\right] \tag{34}$$

$$\widehat{\mu}_{\mathrm{MLE}} = \frac{\widetilde{y}_1}{n} + \frac{1}{n}\sum_{i=2}^n \frac{\widetilde{c}_i\widetilde{y}_i^*}{\lambda_i} \tag{35}$$

$$\mathrm{err}_{\mathrm{MLE}} = \frac{2.58}{n}\sqrt{\sum_{i=2}^n \frac{|\widetilde{y}_i|^2}{\lambda_i}\left(c_0 - \frac{1}{n}\sum_{i=1}^n \frac{|\widetilde{c}_i|^2}{\lambda_i}\right)}, \tag{36}$$

Since all the quantities on the right hand sides can be obtained in in $\mathcal{O}(n\log(n))$ operations by fast transforms, the left hand sides are all computable using the asymptotic computational cost.

Under the further assumption (33) it follows that

$$\widehat{\mu}_{\mathrm{MLE}} = \frac{\widetilde{y}_1}{n} = \frac{1}{n}\sum_{i=1}^n y_i,$$

$$\mathrm{err}_{\mathrm{MLE}} = \frac{2.58}{n}\sqrt{\sum_{i=2}^n \frac{|\widetilde{y}_i|^2}{\lambda_i}\left(1 - \frac{n}{\lambda_1}\right)}. \tag{37}$$

Thus, in this case $\widehat{\mu}$ is simply the sample mean.

3.3 Full Bayes

For the full Bayes approach the cubature is the same as for empirical Bayes. We also defer to empirical Bayes to estimate the parameter $\boldsymbol{\theta}$. The width of the confidence interval is $\mathrm{err}_{\mathrm{full}} := t_{n_j-1,0.995}\widehat{\sigma}_{\mathrm{full}}$, where $\widehat{\sigma}^2_{\mathrm{full}}$ can also be computed swiftly under assumptions (30):

$$\widehat{\sigma}^2_{\mathrm{full}} = \frac{1}{n(n-1)}\sum_{i=2}^n \frac{|\widetilde{y}_i|^2}{\lambda_i} \times \left[\frac{\lambda_1}{n}\left(1 - \frac{\widetilde{c}_1}{\lambda_1}\right)^2 + \left(c_0 - \frac{1}{n}\sum_{i=1}^n \frac{|\widetilde{c}_i|^2}{\lambda_i}\right)\right].$$

Under assumption (33) further simplification can be made:

$$\widehat{\sigma}^2_{\mathrm{full}} = \frac{1}{n(n-1)}\sum_{i=2}^n \frac{|\widetilde{y}_i|^2}{\lambda_i}\left(\frac{\lambda_1}{n} - 1\right). \tag{38}$$

3.4 Generalized Cross-Validation

GCV yields a different cubature, which nevertheless can also be computed quickly using the fast transform. Under assumptions (30):

$$m_{\mathrm{GCV}} = m_{\mathrm{MLE}} = \frac{\widetilde{y}_1}{n} = \frac{1}{n}\sum_{i=1}^{n} y_i,$$

$$s^2_{\mathrm{GCV}} := \frac{1}{n}\sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i^2}\left[\sum_{i=1}^{n}\frac{1}{\lambda_i}\right]^{-1},$$

$$\boldsymbol{\theta}_{\mathrm{GCV}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\log\left(\sum_{i=2}^{n}\frac{|\widetilde{y}_i|^2}{\lambda_i^2}\right) - 2\log\left(\sum_{i=1}^{n}\frac{1}{\lambda_i}\right)\right], \tag{39}$$

$$\widehat{\mu}_{\mathrm{GCV}} = \widehat{\mu}_{\mathrm{MLE}} = \frac{\widetilde{y}_1}{n} + \frac{1}{n}\sum_{i=2}^{n}\frac{\widetilde{c}_i\widetilde{y}_i^*}{\lambda_i}, \tag{40}$$

$$\mathrm{err}_{\mathrm{GCV}} = \frac{2.58}{n}\left\{\sum_{i=2}^{n}\frac{|\widetilde{y}_i|^2}{\lambda_i^2}\left[\sum_{i=1}^{n}\frac{1}{\lambda_i}\right]^{-1}\right. \tag{41}$$

$$\left.\times\left(c_0 - \frac{1}{n}\sum_{i=1}^{n}\frac{|\widetilde{c}_i|^2}{\lambda_i}\right)\right\}^{1/2}. \tag{42}$$

Moreover, under further assumption (33) it follows that

$$\widehat{\mu}_{\mathrm{GCV}} = \widehat{\mu}_{\mathrm{MLE}} = \frac{\widetilde{y}_1}{n} = \frac{1}{n}\sum_{i=1}^{n} y_i,$$

$$\mathrm{err}_{\mathrm{GCV}} = \frac{2.58}{n}\left\{\sum_{i=2}^{n}\frac{|\widetilde{y}_i|^2}{\lambda_i^2}\left[\sum_{i=1}^{n}\frac{1}{\lambda_i}\right]^{-1}\right.$$

$$\left.\times\left(1 - \frac{n}{\lambda_1}\right)\right\}^{1/2}. \tag{43}$$

Thus, in this case $\widehat{\mu}$ is simply the sample mean.

4 Integration Lattices and Shift Invariant Kernels

The preceding sections lay out an automatic Bayesian cubature algorithm whose computational cost is only $\mathcal{O}(n\log(n))$ if n function values are used. However, this algorithm relies on covariance kernel functions, C and designs, $\{\boldsymbol{x}_i\}_{i=1}^n$ that satisfy assumptions (30). We will also satisfy assumption (33). To facilitate the fast transform, it is assumed in this section and the next that n is power of 2.

4.1 Extensible Integration Lattice Node Sets

The design or set of data sites used is defined by a shifted extensible integration lattice node sequence, which takes the form

$$\boldsymbol{x}_i = \boldsymbol{h}\phi(i-1) + \boldsymbol{\Delta} \mod \mathbf{1}, \qquad i \in \mathbb{N}. \tag{44}$$

Here $\boldsymbol{h}$ is a d-dimensional generating vector of positive integers, $\boldsymbol{\Delta}$ is some point in $[0,1)^d$, often chosen at random, and $\{\phi(i)\}_{i=0}^n$ is the van der Corput sequence, defined by reflecting the binary digits of the integer about the decimal point, i.e.,

i	0	1	2	3	4	5	6	7	$\cdots$
i	0_2	1_2	10_2	11_2	100_2	101_2	110_2	111_2	$\cdots$
$\phi(i)$	$_2.0$	$_2.1$	$_2.01$	$_2.11$	$_2.001$	$_2.101$	$_2.011$	$_2.111$	$\cdots$
$\phi(i)$	0	0.5	0.25	0.75	0.125	0.625	0.375	0.875	$\cdots$

(45)

An example of 64 data sites is given in Figure 2. The even coverage of the unit cube is ensured by a well chosen generating vector. The choice of generating vector is typically done offline by computer search. See [4] and [9] for more on extensible integration lattices

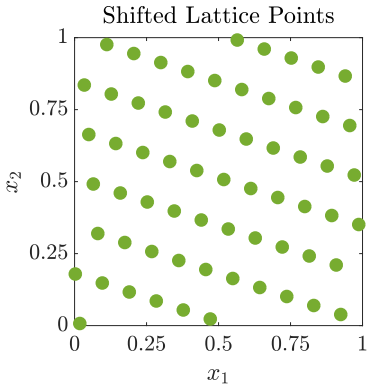

Fig. 2: Example of a shifted integration lattice node set in $d = 2$

4.2 Shift Invariant Kernels

The covariance functions C that match integration lattice node sets have the form

$$C(\boldsymbol{t},\boldsymbol{x}) = K(\boldsymbol{t} - \boldsymbol{x} \bmod \mathbf{1}). \tag{46}$$

This is called a *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 kernel K it follows that assumption (33) is satisfied. Of course, K must also be of the form that ensures that C is symmetric and positive definite, as assumed in (4).

An family of shift invariant kernels is constructed via even degree Bernoulli polynomials:

$$C_{\boldsymbol{\theta}}(\boldsymbol{t},\boldsymbol{x}) = \prod_{l=1}^{d} 1 - (-1)^r \gamma B_{2r}(|x_l - t_l|), \quad \forall \boldsymbol{t},\boldsymbol{x} \in [0,1]^d,$$

$$\boldsymbol{\theta} = (r,\gamma),\ r \in \mathbb{N},\ \gamma > 0. \tag{47}$$

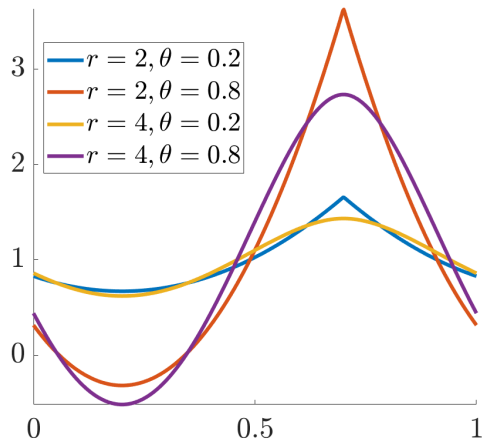

Fig. 3: Shift invariant kernel in 1D shifted by 0.3 to show the discontinuity

Symmetric, positive definite kernels of this form appear in [4] and [6]. The Bernoulli polynomials are described in [13, Chapter 24].

Larger r implies a greater degree of smoothness of the kernel. Larger θ implies greater fluctuations of the output with respect to the input. Plots of $C(\cdot, 0.3)$ are given in Figure 3 for various r and γ values.

4.3 Eigenvectors

For general shift-invariance covariance functions the Gram matrix takes the form

$$\mathsf{C} = \big(C(\boldsymbol{x}_i, \boldsymbol{x}_j)\big)_{i,j=1}^n \tag{48}$$

$$= \Big(K(\boldsymbol{h}(\phi(i-1) - \phi(j-1)) \bmod \mathbf{1})\Big)_{i,j=1}^n. \tag{49}$$

We now demonstrate that the eigenvector matrix for C is

$$\mathsf{V} = \Big(\mathrm{e}^{2\pi n\sqrt{-1}\phi(i-1)\phi(j-1)}\Big)_{i=1}^n. \tag{50}$$

Assumption (30b) follows automatically. Now, note that the k, j element of $\mathsf{V}^H\mathsf{V}$ is

$$\sum_{i=1}^n \mathrm{e}^{2\pi n\sqrt{-1}\phi(i-1)[\phi(j-1)-\phi(k-1)]}.$$

Noting that the sequence $\{\phi(i-1)\}_{i=1}^n$ is a re-ordering of $0, \dots, 1 - 1/n$ for n a power of 2, this sum may be re-written by replacing $\phi(i-1)$ by $(i-1)/n$:

$$\sum_{i=1}^n \mathrm{e}^{2\pi\sqrt{-1}(i-1)[\phi(j-1)-\phi(k-1)]}.$$

Since $\phi(j-1)-\phi(k-1)$ is some integer multiple of $1/n$, it follows that this sum is $n\delta_{j,k}$, where δ is the Kroneker delta function. This establishes that $\mathsf{V}^H = n\mathsf{V}^{-1}$ as in (29).

Next, let $\omega_{k,\ell}$ denote the k, ℓ element of $\mathsf{V}^H\mathsf{C}\mathsf{V}$, which is given by the double sum

$$\omega_{k,\ell} = \sum_{i,j=1}^n K(\boldsymbol{h}(\phi(i-1) - \phi(j-1)) \bmod \mathbf{1}) \times \mathrm{e}^{-2\pi n\sqrt{-1}\phi(k-1)\phi(i-1)}\mathrm{e}^{2\pi n\sqrt{-1}\phi(j-1)\phi(m-1)}$$

Noting that the sequence $\{\phi(i-1)\}_{i=1}^n$ is a re-ordering of $0, \dots, 1 - 1/n$ for n a power of 2, this sum may be re-written by replacing $\phi(i-1)$ by $(i-1)/n$ and $\phi(j-1)$ by $(j-1)/n$:

$$\omega_{k,\ell} = \sum_{i,j=1}^n K\left(\boldsymbol{h}\left(\frac{i-j}{n}\right) \bmod \mathbf{1}\right) \times \mathrm{e}^{-2\pi\sqrt{-1}\phi(k-1)(i-1)}\mathrm{e}^{2\pi\sqrt{-1}(j-1)\phi(\ell-1)}.$$

This sum also remains unchanged if i is replaced by $i+m$ and j is replaced by $j+m$ for any integer m:

$$\omega_{k,\ell} = \sum_{i,j=1}^n K\left(\boldsymbol{h}\left(\frac{i-j}{n}\right) \bmod \mathbf{1}\right) \times \mathrm{e}^{-2\pi\sqrt{-1}\phi(k-1)(i+m-1)}\mathrm{e}^{2\pi\sqrt{-1}(j+m-1)\phi(\ell-1)} = \omega_{k,\ell}\mathrm{e}^{2\pi\sqrt{-1}m(\phi(\ell-1)-\phi(k-1))}.$$

For this last equality to hold for all integers m, we must have $k = \ell$ or $\omega_{k,\ell} = 0$. Thus,

$$\omega_{k,\ell} = \delta_{k,\ell}\sum_{i,j=1}^n K\left(\boldsymbol{h}\left(\frac{i-j}{n}\right) \bmod \mathbf{1}\right) \times \mathrm{e}^{-2\pi\sqrt{-1}(i-j)\phi(k-1)} = n\delta_{k,\ell}\sum_{i=1}^n K\left(\left(\frac{i\boldsymbol{h}}{n}\right) \bmod \mathbf{1}\right)\mathrm{e}^{-2\pi\sqrt{-1}i\phi(k-1)}.$$

This establishes $\mathsf{V}^H\mathsf{C}\mathsf{V}$ as a diagonal matrix whose diagonal elements are n times the eigenvalues, i.e., $\lambda_k = \omega_{k,k}/n$. Furthermore, V is the matrix of eigenvectors, which satisfies assumption (30a).

4.4 Iterative Computation of the Fast Transform

Assumption (30a) is that computing $\mathsf{V}^T\boldsymbol{b}$ requires only $\mathcal{O}(n\log(n))$ operations. Recall that we assume that n is a power of 2. This can be accomplished by an iterative algorithm. Let $\mathsf{V}^{(n)}$ denote the $n\times n$ matrix V defined in (50). We show how to compute $\mathsf{V}^{(2n)T}\boldsymbol{b}$ quickly for all $\boldsymbol{b} \in \mathbb{R}^{2n}$ assuming that $\mathsf{V}^{(n)T}\boldsymbol{b}$ can be computed quickly for all $\boldsymbol{b} \in \mathbb{R}^n$.

From the definition of the van der Corput sequence in (45) it follows that

$$\phi(2i) = \phi(i)/2,\ \phi(2i+1) = [\phi(i)+1]/2, \quad i \in \mathbb{N}_0 \tag{51}$$

$$\phi(i+n) = \phi(i) + 1/(2n), \qquad i = 0, \dots, n-1, \tag{52}$$

$$n\phi(i) \in \mathbb{N}_0, \qquad i = 0, \dots, n-1, \tag{53}$$

still assuming that n is an integer power of two. Let $\widetilde{\boldsymbol{b}} = \mathsf{V}^{(2n)T}\boldsymbol{b}$ for some arbitrary $\boldsymbol{b} \in \mathbb{R}^{2n}$, and define

$$\boldsymbol{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_{2n} \end{pmatrix}, \quad \boldsymbol{b}^{(1)} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}, \quad \boldsymbol{b}^{(2)} = \begin{pmatrix} b_{n+1} \\ \vdots \\ b_{2n} \end{pmatrix},$$

$$\widetilde{\boldsymbol{b}} = \begin{pmatrix} \widetilde{b}_1 \\ \vdots \\ \widetilde{b}_{2n} \end{pmatrix}, \quad \widetilde{\boldsymbol{b}}^{(1)} = \begin{pmatrix} b_1 \\ b_3 \\ \vdots \\ b_{2n-1} \end{pmatrix}, \quad \widetilde{\boldsymbol{b}}^{(2)} = \begin{pmatrix} b_2 \\ b_4 \\ \vdots \\ b_{2n} \end{pmatrix}.$$

It follows from these definitions and the definition of V in (50) that

$$\begin{aligned}
\widetilde{\boldsymbol{b}}^{(1)} &= \left(\sum_{j=1}^{2n} \mathrm{e}^{4\pi n\sqrt{-1}\phi(2i-2)\phi(j-1)} b_j \right)_{i=1}^{n} \\
&= \left(\sum_{j=1}^{2n} \mathrm{e}^{2\pi n\sqrt{-1}\phi(i-1)\phi(j-1)} b_j \right)_{i=1}^{n} \qquad \text{by (51)} \\
&= \left(\sum_{j=1}^{n} \mathrm{e}^{2\pi n\sqrt{-1}\phi(i-1)\phi(j-1)} b_j \right)_{i=1}^{n} \\
&\qquad + \left(\sum_{j=1}^{n} \mathrm{e}^{2\pi n\sqrt{-1}\phi(i-1)\phi(n+j-1)} b_{n+j} \right)_{i=1}^{n} \\
&= \mathsf{V}^{(n)}\boldsymbol{b}^{(1)} \\
&\quad + \left(\mathrm{e}^{\pi\sqrt{-1}\phi(i-1)} \sum_{j=1}^{n} \mathrm{e}^{2\pi n\sqrt{-1}\phi(i-1)\phi(j-1)} b_{n+j} \right)_{i=1}^{n} \\
&\qquad\qquad \text{by (52)} \\
&= \mathsf{V}^{(n)}\boldsymbol{b}^{(1)} + \left(\mathrm{e}^{\pi\sqrt{-1}\phi(i-1)}\right)_{i=1}^{n} \odot \left(\mathsf{V}^{(n)}\boldsymbol{b}^{(2)}\right),
\end{aligned}$$

where $\odot$ denotes the Hadamard (term-by-term) product. By a similar argument,

$$\begin{aligned}
\widetilde{\boldsymbol{b}}^{(2)} &= \left(\sum_{j=1}^{2n} \mathrm{e}^{4\pi n\sqrt{-1}\phi(2i-1)\phi(j-1)} b_j \right)_{i=1}^{n} \\
&= \left(\sum_{j=1}^{2n} \mathrm{e}^{2\pi n\sqrt{-1}[\phi(i-1)+1]\phi(j-1)} b_j \right)_{i=1}^{n} \qquad \text{by (51)} \\
&= \left(\sum_{j=1}^{n} \mathrm{e}^{2\pi n\sqrt{-1}[\phi(i-1)+1]\phi(j-1)} b_j \right)_{i=1}^{n} \\
&\qquad + \left(\sum_{j=1}^{n} \mathrm{e}^{2\pi n\sqrt{-1}[\phi(i-1)+1]\phi(n+j-1)} b_{n+j} \right)_{i=1}^{n} \\
&= \mathsf{V}^{(n)}\boldsymbol{b}^{(1)} + \Big(\mathrm{e}^{\pi\sqrt{-1}[\phi(i-1)+1]} \\
&\qquad\qquad \times \sum_{j=1}^{n} \mathrm{e}^{2\pi n\sqrt{-1}\phi(i-1)\phi(j-1)} b_{n+j} \Big)_{i=1}^{n} \\
&\qquad\qquad \text{by (52) and (53)} \\
&= \mathsf{V}^{(n)}\boldsymbol{b}^{(1)} - \left(\mathrm{e}^{\pi\sqrt{-1}\phi(i-1)}\right)_{i=1}^{n} \odot \left(\mathsf{V}^{(n)}\boldsymbol{b}^{(2)}\right).
\end{aligned}$$

The computational cost to compute $\mathsf{V}^{(2n)T}\boldsymbol{b}$ is then twice the cost of computing $\mathsf{V}^{(n)T}\boldsymbol{b}^{(1)}$ plus $2n$ multiplications plus $2n$ additions/subtractions. An inductive argument establishes that $\mathsf{V}^{(n)T}\boldsymbol{b}$ requires only $\mathcal{O}(n\log(n))$ operations.

4.5 Overcoming Cancellation Error

For the kernels used in our computation it may happen that n/λ_1 is close to 1. Thus, the term $1 - n/\lambda_1$, which appears in the credible interval widths, $\mathrm{err}_{\mathrm{MLE}}$, $\mathrm{err}_{\mathrm{full}}$, and $\mathrm{err}_{\mathrm{GCV}}$, may suffer from cancellation error. We can avoid this cancellation error by modifying how we compute the Gram matrix and its eigenvalues.

Define a new function $\mathring{C} := C - 1$, and its associated Gram matrix $\mathring{\mathsf{C}} = \mathsf{C} - \mathbf{1}\mathbf{1}^T$. Note that $\mathring{C}$ inherits the shift-invariant properties of C. Since $\mathbf{1}$ is the first eigenvector of C it follows that the eigenvectors of $\mathring{\mathsf{C}}$ are $\mathring{\lambda}_1 = \lambda_1 - n, \lambda_2, \ldots, \lambda_n$. Moreover,

$$1 - \frac{n}{\lambda_1} = \frac{\lambda_1 - n}{\lambda_1} = \frac{\mathring{\lambda}_1}{\mathring{\lambda}_1 + n}, \tag{54}$$

where now the right hand side is free of cancellation error.

We show how to compute $\mathring{C}$ without introducing round-off error. The covariance functions that we use are of product form, namely,

$$C(\boldsymbol{t}, \boldsymbol{x}) = \prod_{\ell=1}^{d} \left[1 + \mathring{C}_\ell(t_\ell, x_\ell)\right], \qquad \mathring{C}_\ell : [0,1]^2 \to \mathbb{R}.$$

Direct computation of $\mathring{C}(\boldsymbol{t}, \boldsymbol{x}) = C(\boldsymbol{t}, \boldsymbol{x}) - 1$ introduces cancellation error if the $\mathring{C}_\ell$ are small. So, we employ the iteration

$$\begin{aligned}
&\mathring{C}^{(1)} = \mathring{C}_1(t_1, x_1), \\
&\mathring{C}^{(\ell)} = \mathring{C}^{(\ell-1)}[1 + \mathring{C}_\ell(t_\ell, x_\ell)] + \mathring{C}_\ell(t_\ell, x_\ell), \\
&\qquad\qquad\qquad\qquad\qquad\qquad \ell = 2, \ldots, d, \\
&\mathring{C}(\boldsymbol{t}, \boldsymbol{x}) = \mathring{C}^{(d)}.
\end{aligned}$$

In this way, the Gram matrix $\mathring{\mathsf{C}}$, whose i,j-element is $\mathring{C}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ can be constructed with minimal round-off error.

Computing the eigenvalues of $\mathring{\mathsf{C}}$ via the procedure given in (31) yields $\mathring{\lambda}_1 = \lambda_1 - n, \lambda_2, \ldots, \lambda_n$. The estimates of $\boldsymbol{\theta}$ are computed in terms of the eigenvalues of $\mathring{\mathsf{C}}$, so (34) and (39) become

$$\begin{aligned}
\boldsymbol{\theta}_{\mathrm{MLE}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \Bigg[&\log\left(\sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i} \right) \\
&+ \frac{\log(n + \mathring{\lambda}_1)}{n} + \frac{1}{n}\sum_{i=2}^{n} \log(\lambda_i) \Bigg],
\end{aligned} \tag{55}$$

$$\boldsymbol{\theta}_{\text{GCV}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left[\log\left(\sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i^2} \right) - 2\log\left(\frac{1}{n+\mathring{\lambda}_1} + \sum_{i=2}^{n} \frac{1}{\lambda_i} \right) \right]. \tag{56}$$

The widths of the credible intervals in (37), (38), and (43) become

$$\text{err}_{\text{MLE}} = \frac{2.58}{n} \sqrt{ \frac{\mathring{\lambda}_1}{n+\mathring{\lambda}_1} \sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i} },$$

$$\text{err}_{\text{full}} := \frac{t_{n_j-1,0.995}}{n} \sqrt{ \frac{\mathring{\lambda}_1}{n-1} \sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i} },$$

$$\text{err}_{\text{GCV}} = \frac{2.58}{n} \sqrt{ \frac{\mathring{\lambda}_1}{n+\mathring{\lambda}_1} \sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i^2} \left[\frac{1}{n+\mathring{\lambda}_1} + \sum_{i=2}^{n} \frac{1}{\lambda_i} \right]^{-1} }.$$

4.6 Variable transforms

Fast transform kernel technique discussed with shift invariant kernel eqn(**??**) and lattice points so far, assumes the integrand is periodic and has continuous derivatives on the boundaries of the domain $[0,1]^d$. Non-periodic functions do not live in the space spanned by the kernel. Variable transformation or periodization transform techniques are typically used to improve the accuracy in multi-dimensional numerical integrations where boundary conditions needs to be enforced. These transformations could be either polynomial, exponential and also trigonometric nature.

$$\text{Baker's} : \tilde{f}(\boldsymbol{t}) = f\left(\left(1 - 2\left| t_j - \frac{1}{2} \right| \right)_{j=1}^{d} \right)$$

$$\text{C0} : \tilde{f}(\boldsymbol{t}) = f(\boldsymbol{g}_0) \prod_{j=1}^{d} g_0'(t_j),\ \boldsymbol{g}_0 = (g_0(t_j))_{j=1}^{d}$$

$$\text{C1} : \tilde{f}(\boldsymbol{t}) = f(\boldsymbol{g}_1) \prod_{j=1}^{d} g_1'(t_j),\ \boldsymbol{g}_1 = (g_1(t_j))_{j=1}^{d}$$

$$\text{Sidi's C1} : \tilde{f}(\boldsymbol{t}) = f(\boldsymbol{\psi}_2) \prod_{j=1}^{d} \psi_2'(t_j),\ \boldsymbol{\psi}_2 = (\psi_2(t_j))_{j=1}^{d}$$

$$\text{Sidi's C2} : \tilde{f}(\boldsymbol{t}) = f(\boldsymbol{\psi}_3) \prod_{j=1}^{d} \psi_3'(t_j),\ \boldsymbol{\psi}_3 = (\psi_3(t_j))_{j=1}^{d}$$

where

$$g_0(t) = 3t^2 - 2t^3, g_0'(t) = 6t(1-t))$$

$$g_1(t) = t^3(10 - 15t + 6t^2), g_1'(t) = 30t^2(1-t)^2$$

$$\psi_2(t) = \left(t - \frac{1}{2\pi}\sin(2\pi t) \right), \psi_2'(t) = (1 - \cos(2\pi t))$$

$$\psi_3(t) = \frac{1}{16}\left(8 - 9\cos(\pi t) + \cos(3\pi t)\right),$$

$$\psi_3'(t) = \frac{1}{16}\left(9\sin(\pi t)\pi - \sin(3\pi t)3\pi\right)$$

These transforms vary in terms of computational complexity and accuracy, shall be chosen on a need basis. Such as:

1. Baker's : Baker's transform or called tent map in each coordinate. It preserves only continuity but it is easier to compute.
2. C0 : Polynomial transformation only ensures periodicity of function.
3. C1 : Polynomial transformation preserving the first derivative.
4. C1sin : Sidi's transform with Sine, preserving the first derivative. This is, in general, a better option than 'C1'.
5. C2sin : Sidi's transform with Sine, preserving upto second derivative. We use this when smoothness of 'C1sin' is not sufficient and need to preserve upto second derivative.

4.7 Validating Gaussian process assumption

We begin developing the Bayesian cubature with the assumption that the integrand arises from a Gaussian process. How can we check if $m, s, \boldsymbol{\theta}$ and C are chosen appropriately so that the $\boldsymbol{f}$ is a draw from the Gaussian process?. Here is an attempt to validate that assumption. Let,

$$\boldsymbol{f} = (f(\boldsymbol{x}_i))_{i=1}^{n} \sim \mathcal{N}(m\mathbf{1}, \mathsf{C}),$$

$$\text{where} \quad \mathsf{C} = \frac{1}{n}\mathsf{V}\Lambda\mathsf{V}^H, \quad \mathsf{V}^H\mathsf{V} = n$$

Then,

$$\begin{aligned} \mathbb{E}\left[\frac{1}{\sqrt{n}}\Lambda^{-\frac{1}{2}}\mathsf{V}^H\boldsymbol{f} \right] &= \frac{1}{\sqrt{n}}\Lambda^{-\frac{1}{2}}\mathsf{V}^H\mathbb{E}[\boldsymbol{f}] \\ &= \frac{1}{\sqrt{n}}\Lambda^{-\frac{1}{2}}\mathsf{V}^H m\mathbf{1} \\ &= m\sqrt{\frac{n}{\lambda_1}} \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \end{aligned}$$

Let $\boldsymbol{f}' = \frac{1}{\sqrt{n}}\Lambda^{-\frac{1}{2}}\mathsf{V}^H\boldsymbol{f}$,

$$\begin{aligned} \text{cov}\left[\boldsymbol{f}'\right] &= \frac{1}{n}\mathbb{E}\left[\Lambda^{-\frac{1}{2}}\mathsf{V}^H(\boldsymbol{f} - m\mathbf{1})(\boldsymbol{f} - m\mathbf{1})^T\mathsf{V}\Lambda^{-\frac{1}{2}} \right] \\ &= \frac{1}{n}\Lambda^{-\frac{1}{2}}\mathsf{V}^H\mathbb{E}\left[(\boldsymbol{f} - m\mathbf{1})(\boldsymbol{f} - m\mathbf{1})^T \right]\mathsf{V}\Lambda^{-\frac{1}{2}} \\ &= \frac{1}{n}\Lambda^{-\frac{1}{2}}\mathsf{V}^H\frac{1}{n}\mathsf{V}\Lambda\mathsf{V}^H\mathsf{V}\Lambda^{-\frac{1}{2}} \\ &= \mathsf{I} \end{aligned}$$

Thus

$$\boldsymbol{f}' \sim \mathcal{N}(m'\boldsymbol{e}_1, \mathsf{I}),$$

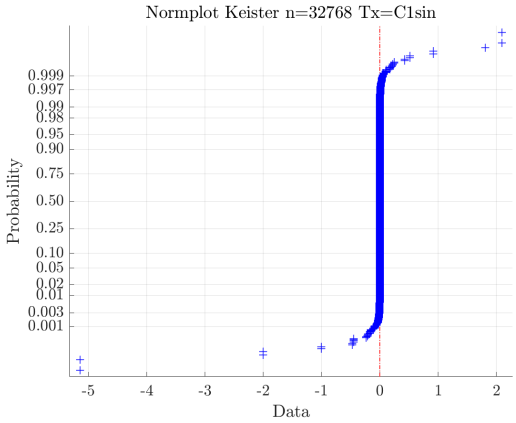

Fig. 4: Normal plot : Keister function with arbMean assumption

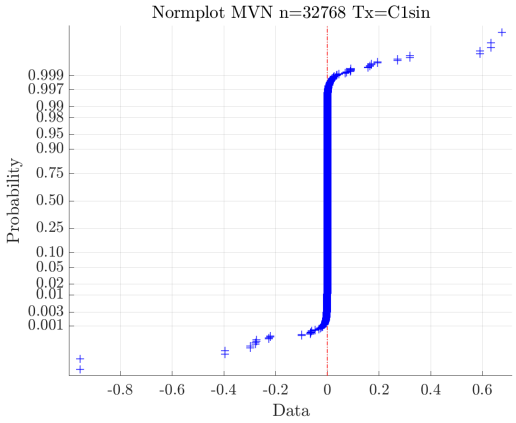

Fig. 5: Normal plot : MVN with arbMean assumption

Where $m' = m\sqrt{\frac{n}{\lambda_1}}$. If we can verify the sample distribution of $\boldsymbol{f}'$ is approximately $\mathcal{N}(m'\boldsymbol{e}_1, \mathsf{I})$ by using Normal plots, could validate our assumption.

We plotted the Normal plots of $\boldsymbol{f}'$ for Keister and Multivariate Normal functions as shown in Figure 5 and Figure 4. These plots show the transformed $\boldsymbol{f}'$ follows approximately standard normal with a constant mean.

4.8 Numerical Experiments

Having all the tools and optimization done handy, now we shall run the numerical simulations to see the performance.

4.8.1 Test Functions

The following test functions were used to test the integration speed and accuracy of *Fast Bayesian cubature algorithm* that we developed so far

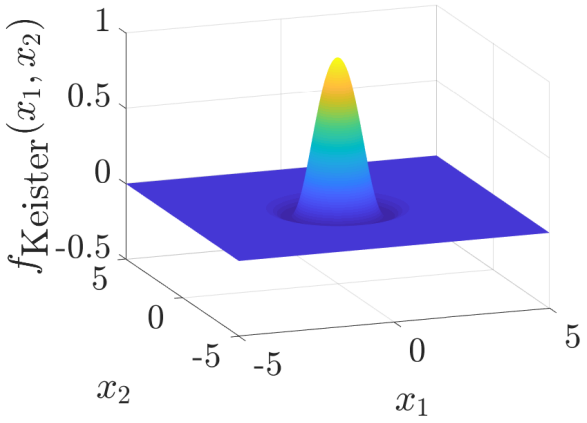

Fig. 6: Keister function in d=2

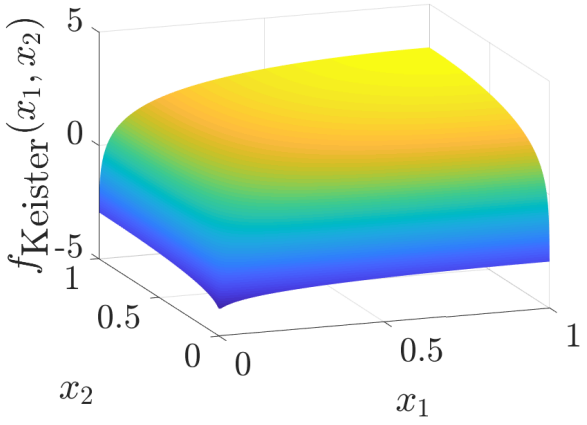

Fig. 7: Keister function transformed to $[0,1]^2$

1. Keister function
   The following multidimensional integral example is based on the paper [11], inspired by a physics application. There exist an iterative algorithm to compute the true integral of this function. But the domain of this function is $\mathbb{R}^d$ so variation transformation must be used to use with $[0,1]^d$.

$$f(\boldsymbol{x}) = \cos(\|\boldsymbol{x}\|)\exp(-\|\boldsymbol{x}\|^2)\mathrm{d}\boldsymbol{x},$$

$$\int_{\mathbb{R}^d} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \frac{2\pi^{\frac{d}{2}}}{\Gamma(\frac{d}{2})}\mathrm{CI}(d), \quad d = 1, 2, 3, \ldots$$

where, $\Gamma :=$ gamma function, and

$$\mathrm{CI}(1) = \frac{\sqrt{\pi}}{2\exp(1/4)},$$

$$\mathrm{SI}(1) = \int_{x=0}^{\infty} \exp(-\boldsymbol{x}^T\boldsymbol{x})\sin(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$$
$$= 0.4244363835020225,$$

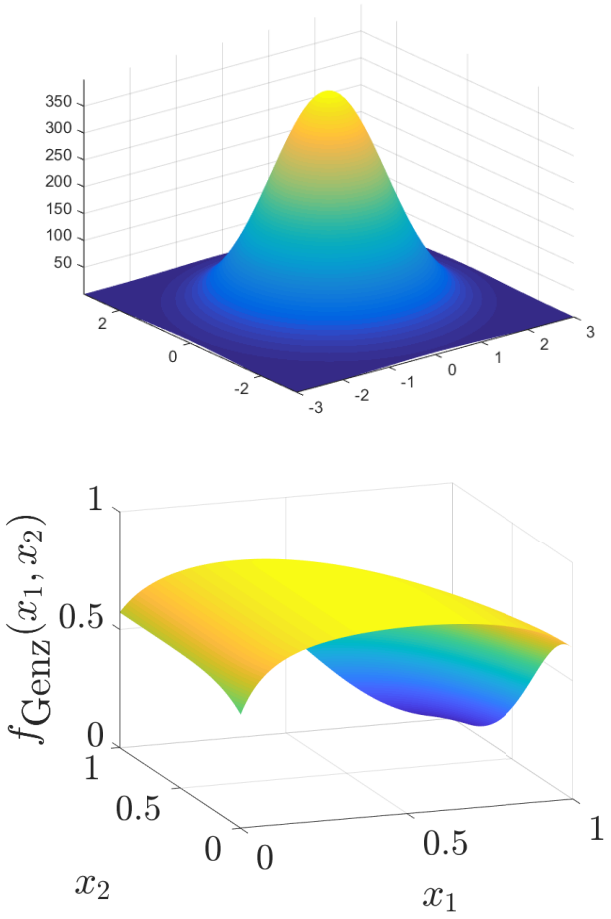

Fig. 8: Mutivariate Normal and Genz function

$$\text{CI}(2) = \frac{1 - \text{SI}(1)}{2}, \ \text{SI}(2) = \frac{\text{CI}(1)}{2}$$
$$\text{CI}(j) = \frac{(j-2)\text{CI}(j-2) - \text{SI}(j-1)}{2}, \ j = 3, 4, \ldots$$
$$\text{SI}(j) = \frac{(j-2)\text{SI}(j-2) - \text{CI}(j-1)}{2}, \ j = 3, 4, \ldots.$$

2. Multivariate Normal:
We use the Genz's method to compute Multivariate normal probability as explained below. This method reduces the original dimension of the problem by 1.

$$\mu = \int_{[\boldsymbol{a},\boldsymbol{b}]\in\mathbb{R}^d} \frac{\exp\left(-\frac{1}{2}\boldsymbol{t}^T\Sigma^{-1}\boldsymbol{t}\right)}{\sqrt{(2\pi)^d \det(\Sigma)}}\,\mathrm{d}\boldsymbol{t} \overset{[5]}{=} \int_{[0,1]^{d-1}} f_{\text{Genz}}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$$

where $\Sigma = \mathsf{L}\mathsf{L}^T$ is the Cholesky decomposition of the covariance matrix, $\mathsf{L} = (l_{jk})_{j,k=1}^d$ is a lower triangular matrix, and

$$\boldsymbol{\alpha}_1 = \Phi(a_1), \qquad \boldsymbol{\beta}_1 = \Phi(b_1)$$

$$\boldsymbol{\alpha}_j(x_1, \ldots, x_{j-1}) = \Phi\left(\frac{1}{l_{jj}}\left(a_j - \sum_{k=1}^{j-1} l_{jk}\Phi^{-1}(\boldsymbol{\alpha}_k + x_k(\boldsymbol{\beta}_k - \boldsymbol{\alpha}_k))\right)\right), \quad j = 2, \ldots, d,$$

$$\boldsymbol{\beta}_j(x_1, \ldots, x_{j-1}) = \Phi\left(\frac{1}{l_{jj}}\left(b_j - \sum_{k=1}^{j-1} l_{jk}\Phi^{-1}(\boldsymbol{\alpha}_k + x_k(\boldsymbol{\beta}_k - \boldsymbol{\alpha}_k))\right)\right), \quad j = 2, \ldots, d,$$

$$f_{\text{Genz}}(\boldsymbol{x}) = \prod_{j=1}^{d} [\boldsymbol{\beta}_j(\boldsymbol{x}) - \boldsymbol{\alpha}_j(\boldsymbol{x})].$$

As we see from the figure, Genz function is not periodic, So we need to make it periodic to get the best accuracy with Bayesian cubature.
We use the following parameter values for the numerical examples

	a	b	L
$d=2$	$\begin{pmatrix}-6\\-2\\-2\end{pmatrix}$	$\begin{pmatrix}5\\2\\1\end{pmatrix}$	$\begin{pmatrix}4&1&1\\0&1&0.5\\0&0&0.25\end{pmatrix}$
$d=3$	$\begin{pmatrix}-6\\-2\\-2\\2\end{pmatrix}$	$\begin{pmatrix}5\\2\\1\\2\end{pmatrix}$	$\begin{pmatrix}4&1&1&1\\0&1&0.5&0.5\\0&0&0.25&0.25\\0&0&0&0.25\end{pmatrix}$

3. 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 Brownian motion, B, at time $t_1, \ldots, t_d$, then $Z = (B(t_1), \ldots, B(t_d)) \sim \mathcal{N}(\boldsymbol{0}, \Sigma)$, where $\Sigma = (\min(t_j, t_k))_{j,k=1}^d$, and the fair price of the option is:

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

where the function payoff(.) defines the discounted payoff of the option,

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

and L is any square matrix satisfying $\Sigma = \mathsf{L}\mathsf{L}^T$.
For the Asian arithmetic mean call option:

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

Where $S_j = S_0 \exp((r - \sigma^2/2)t_j + \sigma z_j$, d - number of dimensions and T, S, S_0, K, r, σ are the parameters to be specified.

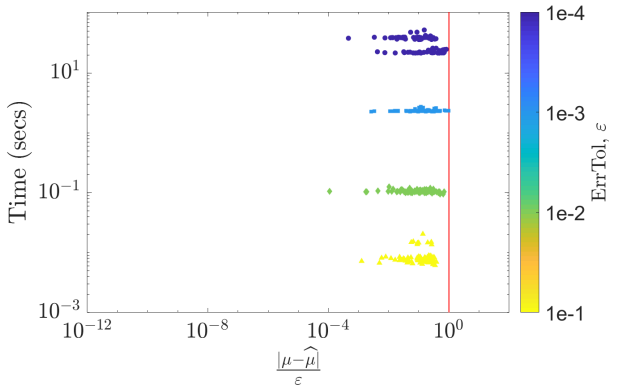

Fig. 9: Option pricing estimated within ε using empirical Bayes stopping criterion

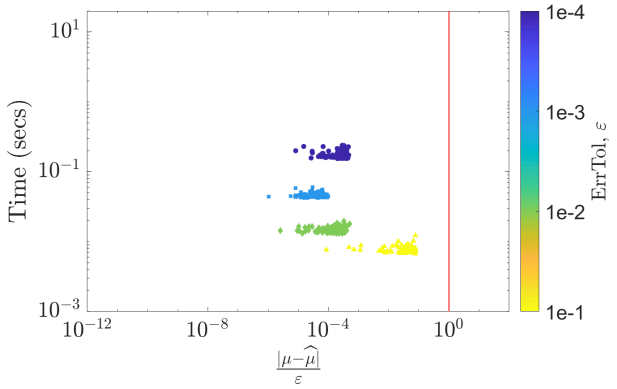

Fig. 10: Keister function integrated within ε using empirical Bayes stopping criterion

4.9 Results and observation

We tested our cubature to integrate the above specified functions. The following plots summarize the results. We used a random shift of $\delta \sim \mathcal{U}[0,1]^d$ to randomly shift the rank-1 Lattice points. This provides randomness in the cubature's behavior. Then we run the cubature for 100 times and take the mean of it as the final result. We integrated with the error thresholds $\varepsilon \in \{1E-2, 1E-3, 1E-4, 1E-5\}$. We use $\frac{|\mu-\widehat{\mu}|}{\varepsilon}$ as x-axis so that a single plot can include all these ε.

The figures Figure 9, Figure 10 and Figure 11 show the integration using empirical Bayes stopping criterion within the error tolerance ε. And, the figures Figure 12, Figure 13 and Figure 14 show the integration using full Bayes stopping criterion within the error tolerance ε. Finally, the figures Figure 15, Figure 16 and Figure 17 show the integration using full Bayes stopping criterion and generalized cross validation (GCV) within the error tolerance ε.

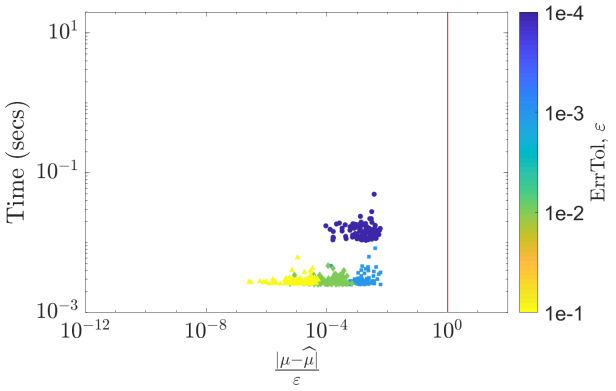

Fig. 11: MVN integrated within ε using empirical Bayes stopping criterion

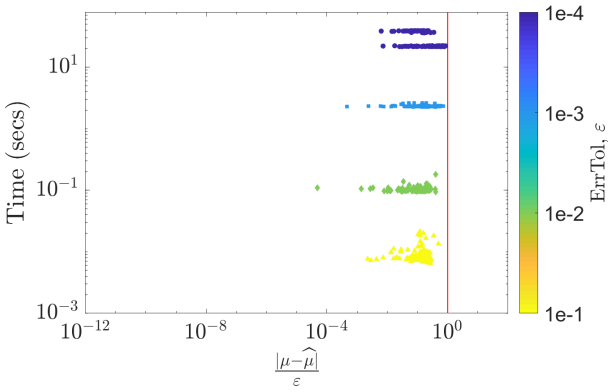

Fig. 12: Option pricing estimated within ε using Full Bayes stopping criterion

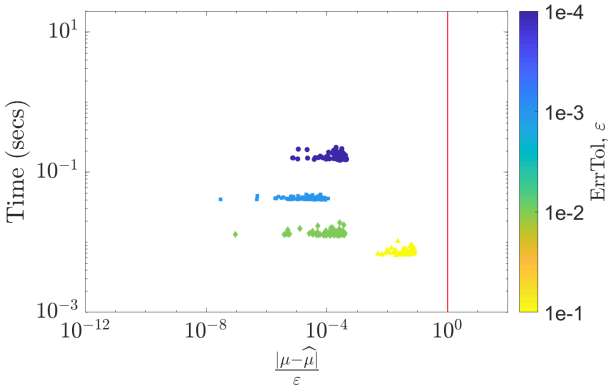

Fig. 13: Keister integrated within ε using Full Bayes stopping criterion

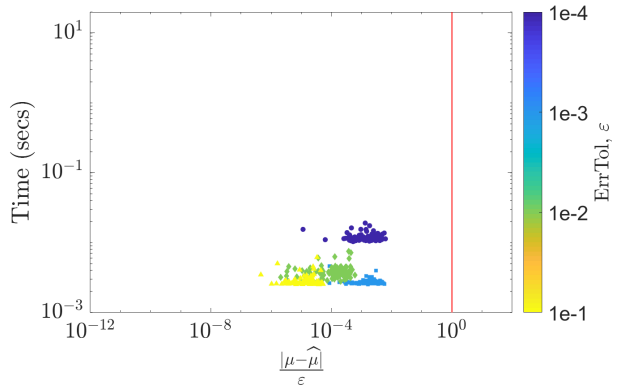

Fig. 14: MVN integrated within ε using Full Bayes stopping criterion

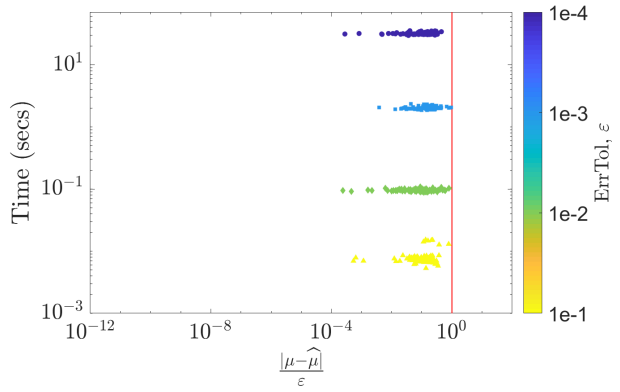

Fig. 15: Option pricing estimated within ε using Full Bayes stopping criterion and GCV

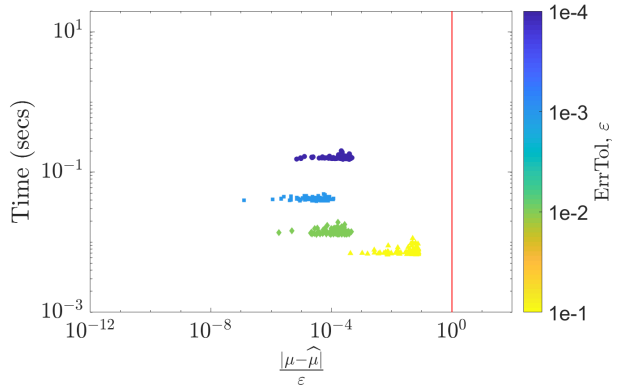

Fig. 16: Keister integrated within ε using Full Bayes stopping criterion and GCV

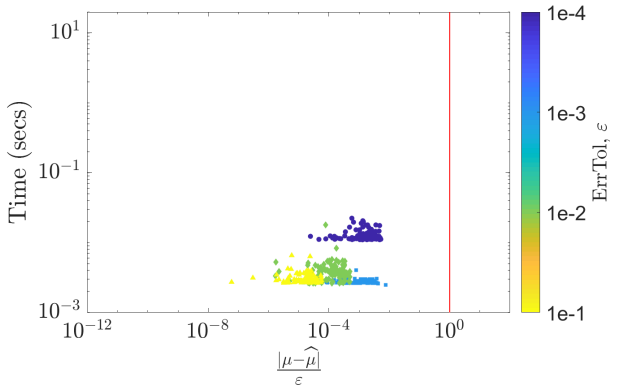

Fig. 17: MVN integrated within ε using Full Bayes stopping criterion and GCV

5 Conclusion

We developed a generalized technique of *Fast transform kernel*. Using this technique, further developed fast automatic Bayesian cubature algorithm that takes very low computational cost in the order of $\mathcal{O}(n \log n)$ comparing to direct Bayesian cubature of $\mathcal{O}(n^3)$, so it can be used in practical applications. By adjusting the Bernoulli order r of the kernel and using the appropriately smoother variable transformation, we could get higher order of error convergence when the integrand is assumed to have zero mean. In general case without any assumption on the integrand mean , the optimal $\widehat{\mu}$ is just the sample mean and so the order of convergence is defined by how smoother the underlying function being integrated and the variable transformation being used. Though the optimal $\widehat{\mu}$ is just the sample mean, the error bound err_n still depends on the kernel order. So when we use higher order r, the error bound err_n closely matches the actual error. We could use the algorithm to integrate upto 2^{23} data points on a 16GB of RAM memory and i7-3630QM computer within 5 minutes. Numerical results show that the theoretical error err_n closely matches the actual error but it could still be improved with more tighter error bound especially when the r is smaller.

6 Future work

As an extension to the ideas and techniques established in this work, the following are considered potential future work

1. Sobol points and Fast Walsh Transform (FWT)
   We have shown one example of a special form of kernel with defined requirements to build a *Fast transform kernel*. Using the established generalized

requirements for a fast-transform-kernel, we could use the same approach with other kernels with suitable point-sets to achieve similar or better performance and accuracy. One such point-sets that to consider in future is, *Sobol points* and with appropriate choice of kernel, which should lead to *Fast Walsh Transform*.

2. Control variates
   We would like to approximate a function of the form $(f - \beta_1 g_1 -, ..., -\beta_p g_p)$
   than
   $f = \mathcal{N}\left(\beta_0 + \beta_1 g_1 +, ..., +\beta_p g_p, s^2 \mathsf{C}\right)$
3. Function approximation
   Let us consider approximating a function of the form
   $$\int_{[0,1]^d} \underbrace{f(\boldsymbol{\phi}(\boldsymbol{t})).\left|\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{t}}\right|}_{g(\boldsymbol{t})} \mathrm{d}\boldsymbol{t}$$
   where $\left|\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{t}}\right|$ is Jacobian and then
   $$g(\boldsymbol{\psi}(\boldsymbol{x})) = f(\underbrace{\boldsymbol{\phi}(\boldsymbol{\psi}(\boldsymbol{x}))}_{\boldsymbol{x}}).\left|\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{t}}\right|(\boldsymbol{\psi}(\boldsymbol{x}))$$
   $$f(\boldsymbol{x}) = g(\boldsymbol{\psi}(\boldsymbol{x})).\frac{1}{\left|\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{t}}\right|(\boldsymbol{\psi}(\boldsymbol{x}))}$$
   Finally, the function approximation is
   $$\begin{aligned}\tilde{f}(\boldsymbol{x}) &= \tilde{g}(\boldsymbol{\psi}(\boldsymbol{x})) \\ &= \sum w_i C(.,.)\end{aligned}$$
4. Deterministic interpretation of Bayesian cubature

References

1. Briol, F.X., Oates, C.J., Griolami, M., Osborne, M.A., Sejdinovic, D.: Probabilistic integration: A role in statistical computation? Statist. Sci. (2018+). To appear
2. Cools, R., Nuyens, D. (eds.): Monte Carlo and Quasi-Monte Carlo Methods: MCQMC, Leuven, Belgium, April 2014, *Springer Proceedings in Mathematics and Statistics*, vol. 163. Springer-Verlag, Berlin (2016)
3. Diaconis, P.: Bayesian numerical analysis. Statistical decision theory and related topics IV, Papers from the 4th Purdue symp., West Lafayette, Indiana 1986, p. 163–175 (1988)
4. Dick, J., Kuo, F., Sloan, I.H.: High dimensional integration — the Quasi-Monte Carlo way. Acta Numer. **22**, 133–288 (2013). DOI 10.1017/S0962492913000044
5. Genz, A.: Comparison of methods for the computation of multivariate normal probabilities. Computing Science and Statistics **25**, 400 – 405 (1993)
6. Hickernell, F.J.: Quadrature error bounds with applications to lattice rules. SIAM J. Numer. Anal. **33**, 1995–2016 (1996). Corrected printing of Sections 3-6 in ibid., **34** (1997), 853–866
7. Hickernell, F.J.: The trio identity for quasi-Monte Carlo error analysis. In: P. Glynn, A. Owen (eds.) Monte Carlo and Quasi-Monte Carlo Methods: MCQMC, Stanford, USA, August 2016, Springer Proceedings in Mathematics and Statistics, pp. 13–37. Springer-Verlag, Berlin (2018). ArXiv:1702.01487
8. Hickernell, F.J., Jiménez Rugama, Ll.A.: Reliable adaptive cubature using digital sequences. In: Cools and Nuyens [2], pp. 367–383. ArXiv:1410.8615 [math.NA]
9. Hickernell, F.J., Niederreiter, H.: The existence of good extensible rank-1 lattices. J. Complexity **19**, 286–300 (2003)
10. Jiménez Rugama, Ll.A., Hickernell, F.J.: Adaptive multidimensional integration based on rank-1 lattices. In: Cools and Nuyens [2], pp. 407–422. ArXiv:1411.1966
11. Keister, B.D.: Multidimensional quadrature algorithms. Computers in Physics **10**, 119–122 (1996)
12. O'Hagan, A.: Bayes-hermite quadrature. J. Statist. Plann. Inference **29**, 245–260 (1991)
13. Olver, F.W.J., Lozier, D.W., Boisvert, R.F., Clark, C.W., Dalhuis, A.B.O.: Digital library of mathematical functions (2018). URL http://dlmf.nist.gov/
14. Rasmussen, C.E., Ghahramani., Z.: Bayesian monte carlo. Advances in Neural Information Processing Systems pp. 489–496 (2003)
15. Rasmussen, C.E., Williams, C.: Gaussian Processes for Machine Learning. MIT Press, Cambridge, Massachusetts (2006). (online version at `http://www.gaussianprocess.org/gpml/`)
16. Ritter, K.: Average-case analysis of numerical problems. Lecture Notes in Mathematics, Springer-Verlag, Berlin **vol. 1733**, 163–175 (2000)
17. Wahba, G.: A comparison of gcv and gml for choosing the smoothing parameter in the generalized spline smoothing problem. The Annals of Statistics **13**, 1378–1402 (1985)