Fast Automatic Bayesian Cubature Using Lattice Sampling

R. Jagadeeswaran · Fred. J. Hickernell

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.

1 Introduction

Cubature is the problem of inferring a numerical value for the integral $\mu = \int_{\mathbb{R}^d} g(\boldsymbol{x}) \, d\boldsymbol{x}$, of a multi-dimensional function g where μ has no closed form analytic expression. Typically, g is only accessible in the form of a

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

S. Author
second address

black-box function routine. Cubature is a key component of many problems in scientific computing, finance, statistical modeling, and machine learning.

The integral μ may often be expressed in the form

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

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

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

where the weights, w_0, and $\boldsymbol{w} = (w_i)_{i=1}^n \in \mathbb{R}^n$, and the nodes, $\{x_i\}_{i=1}^n \subset [0,1]^d$, are chosen to make the error, $|\mu - \widehat{\mu}|$, small. In this article the nodes is assumed to be deterministic.

Our concern is constructing 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.,

$$|\mu - \widehat{\mu}| \le \varepsilon \qquad (3)$$

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 Diaconis [3], O'Hagan [12], Ritter [15], Rasmussen and Ghahramani [13], Briol et al. [1], and others. Thus, our algorithm is an example of *probabilistic numerics*.

Our primary contribution here is to demonstrate how the choice of a family of kernels that match the low discrepancy sampling nodes facilitates fast computation of the data-driven stopping criterion. If n function values, at a cost of $\$(f)$ each, are obtained for cubature purposes, then $\mathcal{O}(n \log(n))$ additional oper-

ations are required to check whether the error tolerance is satisfied. The total cost of our algorithm is then $\mathcal{O}(\$(f)n + n \log(n))$. This is significantly fewer operations than the $\mathcal{O}(n^3)$ typically required for Bayesian cubature.

Hickernell [7] compares and contrasts 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 $\mathsf{C} = \left(C_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \right)_{i,j=1}^n$,

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

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

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

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

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

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

Therefore, it follows from [14] Appendix A.2 Gaussian Identities that the *conditional* distribution of the integral given observed function values, $\boldsymbol{f} = \boldsymbol{y}$ is also Gaussian:

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

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

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

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

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

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 maximimum likelihood estimation (MLE). The log-likelihood function of the parameters given the function data $\boldsymbol{y}$ is:

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

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

yields

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

$$s_{\text{MLE}}^2 = \frac{1}{n} (\boldsymbol{y} - m_{\text{MLE}} \mathbf{1})^T \mathsf{C}^{-1} (\boldsymbol{y} - m_{\text{MLE}} \mathbf{1})$$

$$= \frac{1}{n} \boldsymbol{y}^T \left[\mathsf{C}^{-1} - \frac{\mathsf{C}^{-1} \mathbf{1} \mathbf{1}^T \mathsf{C}^{-1}}{\mathbf{1}^T \mathsf{C}^{-1} \mathbf{1}} \right] \boldsymbol{y}, \qquad (11)$$

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

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

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

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

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

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

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

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 $\rho_{m,s^2}(\xi, \lambda) \propto 1/\lambda$. Then the posterior density for the integral given the data using Bayes theorem is

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

where
$$\alpha = (1 - \boldsymbol{c}^T \mathsf{C}^{-1} \mathbf{1})^2 + \mathbf{1}^T \mathsf{C}^{-1} \mathbf{1}(c_0 - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{c}),$$
$$\beta = (1 - \boldsymbol{c}^T \mathsf{C}^{-1} \mathbf{1})(z - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{y})$$

$$+ \mathbf{1}^T \mathsf{C}^{-1} \boldsymbol{y}(c_0 - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{c}),$$
$$\gamma = (z - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{y})^2 + \boldsymbol{y}^T \mathsf{C}^{-1} \boldsymbol{y}(c_0 - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{c}).$$

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:

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

Finally, we simplify the key term via straightforward calculations to the following:
$$\alpha \gamma - \beta^2 \propto 1 + \frac{(z - \widehat{\mu}_{\text{MLE}})^2}{(n-1)\widehat{\sigma}_{\text{full}}^2},$$
where
$$\widehat{\sigma}_{\text{full}}^2 := \frac{1}{n-1} \boldsymbol{y}^T \left[\mathsf{C}^{-1} - \frac{\mathsf{C}^{-1} \mathbf{1} \mathbf{1}^T \mathsf{C}^{-1}}{\mathbf{1}^T \mathsf{C}^{-1} \mathbf{1}} \right] \boldsymbol{y}$$
$$\times \left[\frac{(1 - \boldsymbol{c}^T \mathsf{C}^{-1} \mathbf{1})^2}{\mathbf{1}^T \mathsf{C}^{-1} \mathbf{1}} + (c_0 - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{c}) \right]. \quad (17)$$

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:
$$\mathbb{P}_f \left[|\mu - \widehat{\mu}_{\text{MLE}}| \le \text{err}_{\text{full}} \right] = 99\%, \qquad (18)$$
where
$$\text{err}_{\text{full}} := t_{n_j - 1, 0.995} \widehat{\sigma}_{\text{full}} > \text{err}_{\text{MLE}}. \qquad (19)$$

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 ith 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:

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

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

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

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

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

Thus, (20) may be re-written as

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

Wahba **??** recommended a generalized cross-validation criterion, which replaces the ith diagonal element of A in the denominator by the average diagonal element of A:

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

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

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

$$\boldsymbol{\theta}_{\text{GCV}} = \underset{\boldsymbol{\theta}}{\text{argmin}}\left\{ \log\left(\boldsymbol{y}^T\left[\mathsf{C}^{-2} - \frac{\mathsf{C}^{-2}\mathbf{1}\mathbf{1}^T\mathsf{C}^{-2}}{\mathbf{1}^T\mathsf{C}^{-2}\mathbf{1}}\right]\boldsymbol{y}\right) - 2\log\left(\text{trace}(\mathsf{C}^{-1})\right)\right\}. \tag{23}$$

Plugging this value of m into (7) yields

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

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

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

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

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

where

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

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

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

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

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.

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

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

so far. For this example, we use the Matern kernel:

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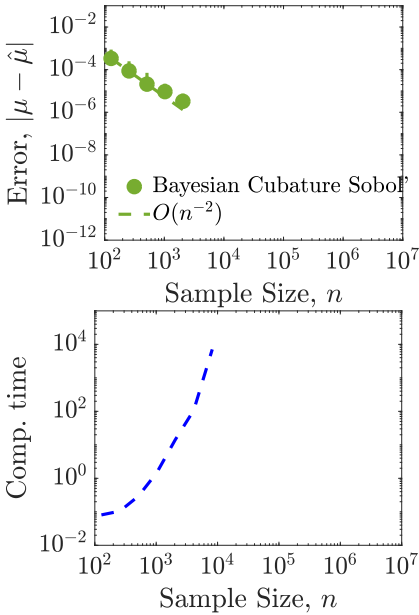
(28)

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

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 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

$$\mathsf{C} = \Big(C_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j)\Big)_{i,j=1}^n = (\boldsymbol{C}_1, \ldots, \boldsymbol{C}_n)$$

$$= \frac{1}{n}\mathsf{V}\Lambda\mathsf{V}^H = \frac{1}{n}\mathsf{V}^*\Lambda\mathsf{V}^T, \qquad \mathsf{V}^H = n\mathsf{V}^{-1}, \qquad (29)$$

$$\mathsf{C}^{-1} = \frac{1}{n}\mathsf{V}\Lambda^{-1}\mathsf{V}^H = \frac{1}{n}\mathsf{V}^*\Lambda^{-1}\mathsf{V}^T,$$

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.

$\mathsf{V} = (\boldsymbol{V}_1, \ldots, \boldsymbol{V}_n) = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n)^T$, also $\boldsymbol{v}_1 = \boldsymbol{V}_1 = \mathbf{1}$

$\Lambda = \mathrm{diag}(\boldsymbol{\lambda}), \quad \boldsymbol{\lambda} = (\lambda_1, \ldots \lambda_n),$

$\mathsf{V}^{-1} = \frac{1}{n}\mathsf{V}^H$, $\mathsf{C}^{-1} = \frac{1}{n}\mathsf{V}\Lambda^{-1}\mathsf{V}^H = \frac{1}{n}\mathsf{V}^*\Lambda^{-1}\mathsf{V}^T$.

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)

$\boldsymbol{v}_1 = \boldsymbol{V}_1 = \mathbf{1}$,   (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:

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

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

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

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

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,

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

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

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

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 $\boldsymbol{c} = \mathbf{1}$. Under (33), the expressions above may be further simplified:

$$\boldsymbol{c}^T \mathsf{C}^{-1} \mathbf{1} = \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{c} = \frac{n}{\lambda_1}.$$

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:

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

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

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

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

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

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 $\widehat{\sigma}_{\mathrm{full}}^2$ can also be computed swiftly under assumptions (30):

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

Under assumption (33) further simplification can be made:

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

3.4 Generalized Cross-Validation

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

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

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

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

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

$$\mathrm{err}_{\mathrm{GCV}} = \frac{2.58}{\sqrt{n}} \left\{ \sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i^2} \left[\sum_{i=1}^{n} \frac{1}{\lambda_i} \right]^{-1} \right.$$
$$\left. \times \left(c_0 - \frac{1}{n} \sum_{i=1}^{n} \frac{|\widetilde{c}_i|^2}{\lambda_i} \right) \right\}^{1/2}.$$

Moreover, under further assumption (33) it follows that

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

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

$$\left. \times \left(1 - \frac{n}{\lambda_1} \right) \right\}^{1/2}. \qquad (38)$$

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).

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

$$\boldsymbol{x}_i = \boldsymbol{h}\phi(i-1) + \boldsymbol{\Delta} \mod \mathbf{1}, \qquad i \in \mathbb{N}. \qquad (39)$$

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	$\cdots$

To facilitate the fast transform to exist, n must be a power of 2.

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

4.2 Shift Invariant Kernels

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

$$C(\boldsymbol{t}, \boldsymbol{x}) = K(\boldsymbol{t} - \boldsymbol{x} \mod \mathbf{1}). \qquad (40)$$

This is called a *shift invariant kernel* because shifting both arguments of the covariance function by the same
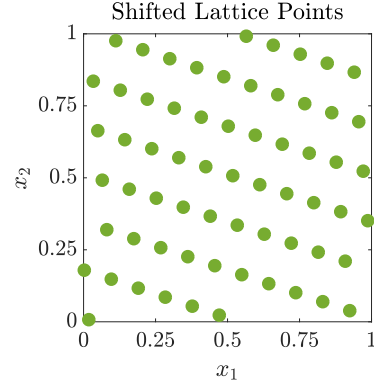

Fig. 2: Example of a shifted integration lattice node set in $d = 2$

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). For covariance functions of the form (40) the Gram matrix takes the form

$$\mathsf{C} = \left(C(\boldsymbol{x}_i, \boldsymbol{x}_j) \right)_{i,j=1}^{n} \qquad (41)$$

$$= \left(K(\boldsymbol{h}(\phi(i-1) - \phi(j-1)) \mod \mathbf{1}) \right)_{i,j=1}^{n}. \qquad (42)$$

We now demonstrate that the eigenvector matrix for C is

$$\mathsf{V} = \left(\mathrm{e}^{2\pi n\sqrt{-1}\phi(i-1)\phi(j-1)} \right)_{i=1}^{n}. \qquad (43)$$

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 $0,\ldots,1-1/n$ 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

$$\omega_{k,\ell} = \sum_{i,j=1}^{n} K(\boldsymbol{h}(\phi(i-1) - \phi(j-1)) \mod \mathbf{1})$$

$$\times \mathrm{e}^{-2\pi n\sqrt{-1}\phi(k-1)\phi(i-1)} \mathrm{e}^{2\pi n\sqrt{-1}\phi(j-1)\phi(m-1)}$$

Noting that the sequence $\{\phi(i-1)\}_{i=1}^{n}$ is a re-ordering of $0,\ldots,1-1/n$ 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 $(j-1)/n$:

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

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

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

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

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

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).

4.3 Iterative Computation of the Fast Transform

4.4 Computing the kernel using Bernoulli polynomials

The shift-invariant-kernel (**??**) cannot be computed directly due to it's infinite sum. If the coefficients $\alpha_{\boldsymbol{k}}$ were chosen appropriately, then there exist a direct expression for the kernel in terms of Bernoulli polynomial. We can use the Fourier series expansion properties [5] of Bernoulli polynomial B_r to find the appropriate $\alpha_{\boldsymbol{k}}$. This provides a closed form expression for the kernel without the infinite sum. It also allows to choose the smoothness of kernel, i.e, how fast the coefficients $\alpha_{\boldsymbol{k}}$ in (**??**) decay. The following very useful expansion is referenced from [5] under eqn (24.8.3):

$$B_r(x) = \frac{-r!}{(2\pi\sqrt{-1})^r} \sum_{\substack{k\neq 0, \\ k=-\infty}}^{\infty} \frac{e^{2\pi\sqrt{-1}kx}}{k^r}$$
$$\begin{cases} \text{for } r=1, & 0 < x < 1 \\ \text{for } r=2,3,\dots & 0 \le x \le 1 \end{cases} \quad (44)$$

Our goal is to replace the infinite sum of the kernel using Bernoulli polynomials. This would allows the computations to be carried out in any software like Matlab. For 1-D ($d=1$) rewriting (44):

$$\sum_{k\neq 0, k=-\infty}^{\infty} \frac{e^{2\pi\sqrt{-1}k|x|}}{(\frac{k}{\theta})^r}$$

$$= -\theta^r B_r(|x|)\frac{(2\pi\sqrt{-1})^r}{r!} \text{ for } -1 \le x \le 1$$

Comparing this expansion with (**??**), one can deduce, for $d = 1$:

$$C(x,t) = \alpha_0 + \sum_{k\neq 0, k=-\infty}^{\infty} \alpha_k e^{2\pi\sqrt{-1}k|x-t|}, \quad -1 \le x, t \le 1,$$

Where the kernel coefficients $\alpha_{\boldsymbol{k}}$ can be expressed explicitly:

$$\alpha_k = \begin{cases} 1, & k = 0 \\ \frac{1}{k^r}, & \text{otherwise} \end{cases},$$

In general for $d > 1$, the coefficients $\alpha_{\boldsymbol{k}}$ for $C_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{t})$ is computed using:

$$\alpha_{\boldsymbol{k},\boldsymbol{\theta}} = \frac{1}{\prod_{l=1}^{d} \max(\frac{|k_l|}{\theta_l}, 1)^r},$$

Where $\boldsymbol{\theta}$ is the shape parameter and 'r' is the Bernoulli polynomial. The value of 'r' will be chosen specific to the integrand when using in the cubature algorithm. We keep the value of 'r' to be even positive integer in this work. Using the above-found results, we get the simplified form of the kernel:

$$C_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{t}) = \prod_{l=1}^{d} 1 - \theta_l^r \frac{(2\pi\sqrt{-1})^r}{r!} B_r(|x_l - t_l|),$$
$$\text{where } 0 \le |x_l - t_l| \le 1, \quad (45)$$

where $\boldsymbol{\theta} \in (0,1]^d$ is the shape parameter, 'r' is the order of the Bernoulli polynomial B_r.

We call (45) a Fourier kernel as the resulting Fast transform is the *Fast Fourier transform* (FFT). Plots of this kernel in $d = 2$ is shown in Figure 3.
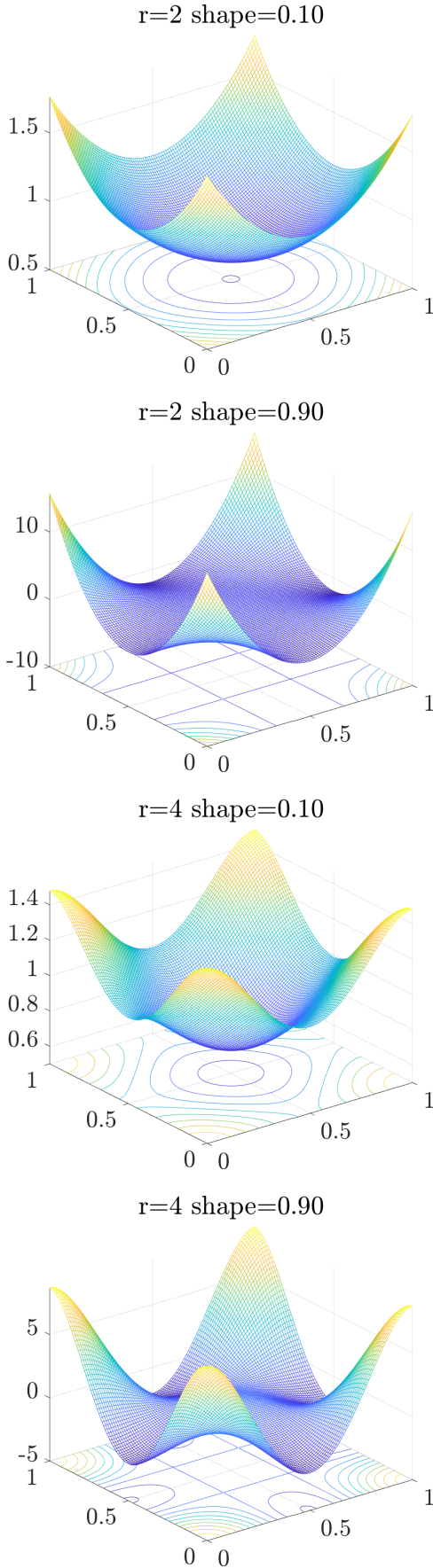
4.5 Variable transforms

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.

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

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

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

r=2 shape=0.10

r=2 shape=0.90

r=4 shape=0.10

r=4 shape=0.90

Fig. 3: Fourier kernel (45) with $r = 2, 4$ and $\theta = 0.1, 0.9$

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

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

where

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

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

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

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

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

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.

4.6 Iterative Discrete Fourier transform for function values

Automatic cubature algorithms needs to compute Discrete Fourier transform of function values $\boldsymbol{y} = (f(\boldsymbol{x}_i))_{i=1}^{n}$ in every iteration with newly added points. Recomputing the whole Fourier transform in every iteration can be avoided when using Rank-1 Lattice points by using structural properties of the Lattice points. Discrete Fourier transform is defined as :

$$\mathcal{DFT}\{y\} := \hat{y} = \left(\sum_{j=1}^{n} y_j e^{-\frac{2\pi\sqrt{-1}}{n}(j-1)(i-1)} \right)_{i=1}^{n}$$

In essence:

$$\hat{y}_i = \sum_{j=1}^{n} y_j e^{-\frac{2\pi\sqrt{-1}}{n}(j-1)(i-1)}$$

We could rearrange the sum into even indexed $j = 2l$ and odd indexed $j = 2l + 1$:

$$\hat{y}_i = \underbrace{\sum_{l=1}^{n/2} y_{2l} e^{-\frac{2\pi\sqrt{-1}}{n/2}(l-1)(i-1)}}_{\text{DFT of even-indexed part of } y_i}$$

$$+ e^{-\frac{2\pi\sqrt{-1}}{n}(i-1)} \underbrace{\sum_{l=1}^{n/2} y_{2l+1} e^{-\frac{2\pi\sqrt{-1}}{n/2}(l-1)(i-1)}}_{\text{DFT of odd-indexed part of } y_i},$$

Which shows two separately computed DFTs can be combined to produce single output. For example, the odd indexed were the existing DFT and the even indexed are from the new halve of samples, algorithm wants to add to improve accuracy. We use this concept to avoid recomputing the full length DFT of $\boldsymbol{y}$ in every iteration. In other words, DFT is computed only for the newly added samples in every iteration.

4.7 Overcoming the cancellation error

During the numerical experiments, we noticed numerical cancellation error in the computation of the term, especially for the bigger values $n > 2^{15}$. Cancellation error happens because two almost equal values are subtracted, when they differ only in very high decimal values.

$$\left(c_{0,\boldsymbol{\theta}} - \boldsymbol{c}_{\boldsymbol{\theta}}^T C^{-1} \boldsymbol{c}_{\boldsymbol{\theta}}\right) = \left(1 - \frac{n}{\lambda_1}\right)$$

In this expression $\frac{n}{\lambda_1}$ almost close to 1. We would like to explore techniques to avoid the subtraction in this computation. Let's recollect the definition of the shift invariant kernel:

$$C(\boldsymbol{x}_i, \boldsymbol{x}_j) = \prod_{k=1}^{d} \left[1 + \theta B(\{x_{i_k} - x_{j_k}\})\right].$$

Let's define the modified kernel:
$$\widetilde{C}(\boldsymbol{x}_i, \boldsymbol{x}_j) := C(\boldsymbol{x}_i, \boldsymbol{x}_j) - 1,$$

Then Its Gram matrix: $\widetilde{C} = C - \boldsymbol{1}\boldsymbol{1}^T$,

Let $(\lambda_1, ..., \lambda_n)$ be the eigenvalues of C. Similarly let $(\tilde{\lambda}_1, ..., \tilde{\lambda}_n)$ be the eigenvalues of $\widetilde{C}$. As per the definition of the Gram matrix C, the eigenvector corresponding to λ_1 is a vector one $\boldsymbol{1}$. Then:
$$\lambda_1 \boldsymbol{1} = C\boldsymbol{1} = (\widetilde{C} + \boldsymbol{1}\boldsymbol{1}^T)\boldsymbol{1} = \tilde{\lambda}_1 \boldsymbol{1} + n\boldsymbol{1},$$

This shows $\tilde{\lambda}_1 = \lambda - n$. In fact for the rest of the values are:
$$\tilde{\lambda}_j = \lambda_j, \ \forall j = 1, ..., n-1. \tag{46}$$

Let $\boldsymbol{v}_j, \forall j = 0, ..., n-1$ be the eigenvectors of C, similarly $\tilde{\boldsymbol{v}}_j, \forall j = 0, ..., n-1$ be the eigenvectors of $\widetilde{C}$,
$$\boldsymbol{v}_j^T C\boldsymbol{1} = \lambda_1 \boldsymbol{v}_j^T \boldsymbol{1} = \boldsymbol{1}^T C\boldsymbol{v}_j = \lambda_j \boldsymbol{1}^T \boldsymbol{v}_j$$

Since $\lambda_1 \neq \lambda_j$, the above statement implies $\boldsymbol{v}_j^T \boldsymbol{1} = 0$. This interesting property provides the proof of the eqn. (46).
$$\lambda_j \boldsymbol{v}_j = C\boldsymbol{v}_j = \boldsymbol{1}_{n \times n} \boldsymbol{v}_j + \widetilde{C}\boldsymbol{v}_j = \widetilde{C}\boldsymbol{v}_j = \tilde{\lambda}_j \boldsymbol{v}_j,$$

Thus proven. Using this result, cancellation error in the computation of err_n can be avoided:
$$\left(1 - \frac{n}{\lambda_1}\right) = \left(1 - \frac{n}{\tilde{\lambda}_1 + n}\right)$$
$$= \left(\frac{\tilde{\lambda}_1 + n - n}{\tilde{\lambda}_1 + n}\right) = \left(\frac{\tilde{\lambda}_1}{\tilde{\lambda}_1 + n}\right).$$

The following technique shows an iterative approach to compute $\tilde{C}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for $d > 1$. This iterative technique is developed using induction:

$$d = 1: \quad C_1 = 1 + \theta B(\{x_{i_1} - x_{j_1}\}) = 1 + \tilde{C}_1$$

$$d = 2: \quad C_2 = [1 + \theta B(\{x_{i_2} - x_{j_2}\})][1 + \tilde{C}_1]$$
$$= 1 + \theta B(\{x_{i_2} - x_{j_2}\})[1 + \tilde{C}_1] + \tilde{C}_1$$
$$= 1 + \underbrace{\theta B(\{x_{i_2} - x_{j_2}\})C_1 + \tilde{C}_1}_{\tilde{C}_2}$$

$$d = 3 \quad C_3 = [1 + \theta B(\{x_{i_3} - x_{j_3}\})][1 + \tilde{C}_2]$$
$$= 1 + \theta B(\{x_{i_3} - x_{j_3}\})[1 + \tilde{C}_2] + \tilde{C}_2$$
$$= 1 + \theta B(\{x_{i_3} - x_{j_3}\})C_2 + \tilde{C}_2$$

$$\vdots$$

$$\forall d > 2 \quad C_d = [1 + \theta B(\{x_{i_d} - x_{j_d}\})][1 + \tilde{C}_{d-1}]$$
$$= 1 + \theta B(\{x_{i_d} - x_{j_d}\})[1 + \tilde{C}_{d-1}] + \tilde{C}_{d-1}$$
$$= 1 + \theta B(\{x_{i_d} - x_{j_d}\})C_{d-1} + \tilde{C}_{d-1}$$

4.8 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,

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

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

Then,

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

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

$$\text{cov}\left[\boldsymbol{f}'\right] = \frac{1}{n} \mathbb{E}\left[\Lambda^{-\frac{1}{2}} V^H (\boldsymbol{f} - m\boldsymbol{1})(\boldsymbol{f} - m\boldsymbol{1})^T V \Lambda^{-\frac{1}{2}}\right]$$
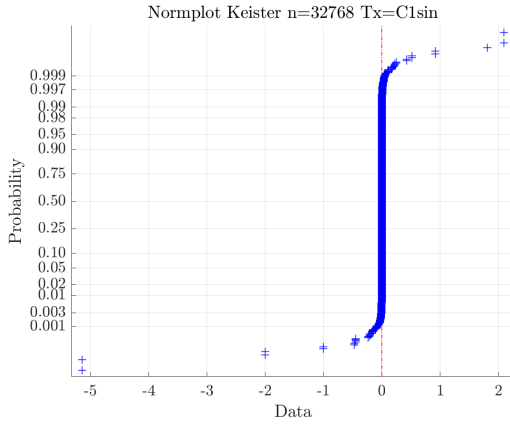
Fig. 4: Normal plot : Keister function with arbMean assumption

Fig. 6: Exp(Cos) in d=2
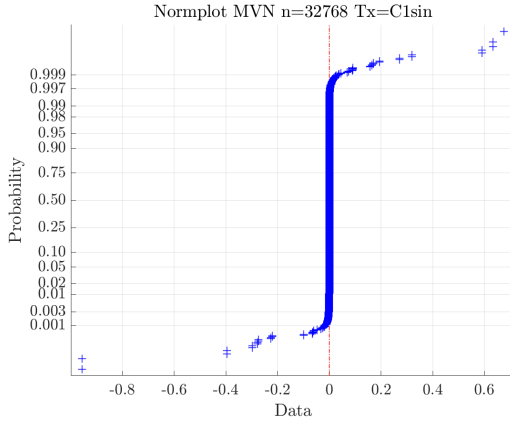
Fig. 5: Normal plot : MVN with arbMean assumption

$$= \frac{1}{n}\Lambda^{-\frac{1}{2}}\mathsf{V}^H\,\mathbb{E}\left[(\boldsymbol{f}-m\boldsymbol{1})(\boldsymbol{f}-m\boldsymbol{1})^T\right]\mathsf{V}\Lambda^{-\frac{1}{2}}$$
$$= \frac{1}{n}\Lambda^{-\frac{1}{2}}\mathsf{V}^H\frac{1}{n}\mathsf{V}\Lambda\mathsf{V}^H\mathsf{V}\Lambda^{-\frac{1}{2}}$$
$$= \mathsf{I}$$

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

Where $m' = m\sqrt{\frac{n}{\lambda_1}}$. If we can verify the sample distribution of $\boldsymbol{f}'$ is approximately $\mathcal{N}\left(m'\boldsymbol{e}_1, \mathsf{I}\right)$ 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.
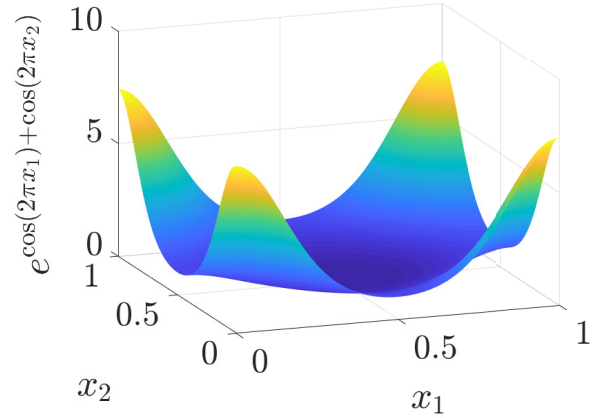
4.9 Numerical Experiments

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

4.9.1 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

1. Exponential of Cosine:
   This is a very simple function, serves as an example to check the working of the cubature, where the function is defined as
   $$f(\boldsymbol{x}) = e^{\sum_{i=1}^{d}\cos(2\pi x_d)}, \quad \int_{[0,1]^d} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = \mathrm{BesselI}(0,1)^d$$
   where 'BesselI' is the *modified Bessel* function. Exp(Cos) function is periodic in $[0,1]$, so we do not need to use any *transform* to make it periodic.

2. Keister function
   The following multidimensional integral example is based on the paper [11], inspired by a physics application.

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

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

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

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

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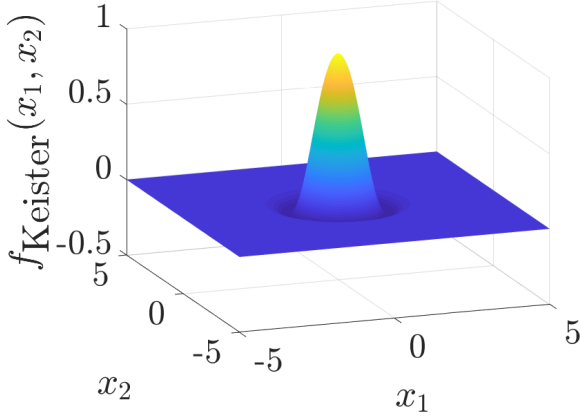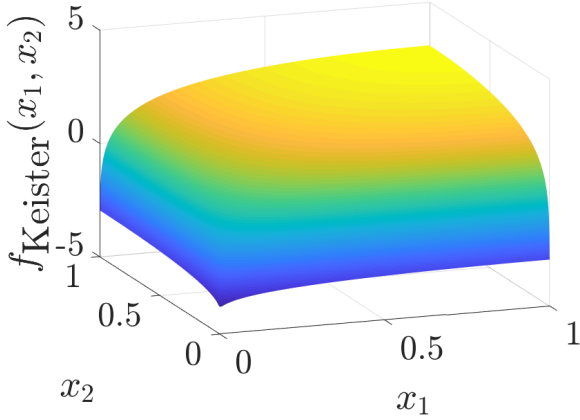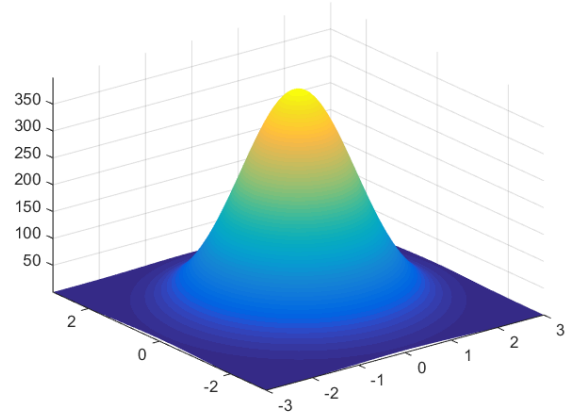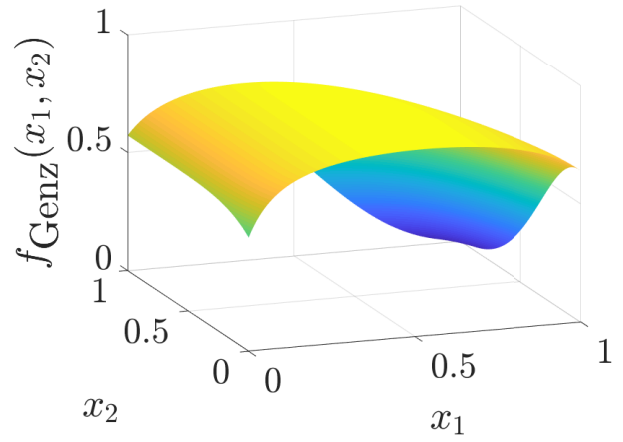 $$= 0.4244363835020225,$$

Fig. 7: Keister function in d=2

Fig. 8: Keister function transformed to $[0,1]^2$

Fig. 9: Mutivariate Normal and Genz function

$$\text{CI}(2) = \frac{1 - \text{SI}(1)}{2}, \ \text{SI}(2) = \frac{\text{CI}(1)}{2}$$

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

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

3. 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.

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

$$\overset{[6]}{=} \int_{[0,1]^{d-1}} f_{\text{Genz}}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$$

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

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

$$\boldsymbol{\alpha}_j(x_1, ..., x_{j-1}) =$$

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

$$j = 2, ..., d,$$

$$\boldsymbol{\beta}_j(x_1, ..., x_{j-1}) =$$

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

$$j = 2, ..., d,$$

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

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

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

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

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.

4.10 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 10, Figure 11 and Figure 12 show the integration done in a finite time within the error tolerance ε. Similarly, the figures Figure 13, Figure 14 and Figure 15 show the integration done using a finite number of samples within the error tolerance ε.
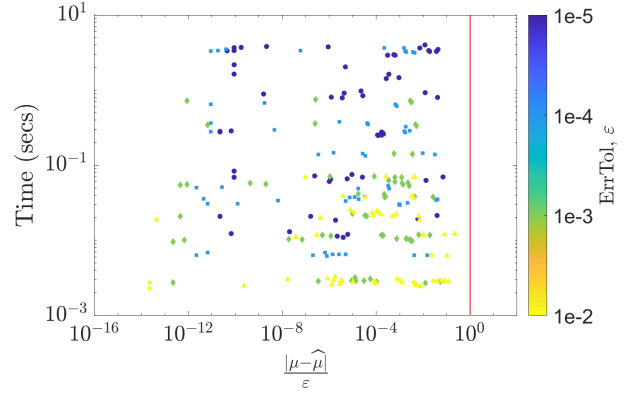

Fig. 10: Exp(Cos) integrated within ε and finite time
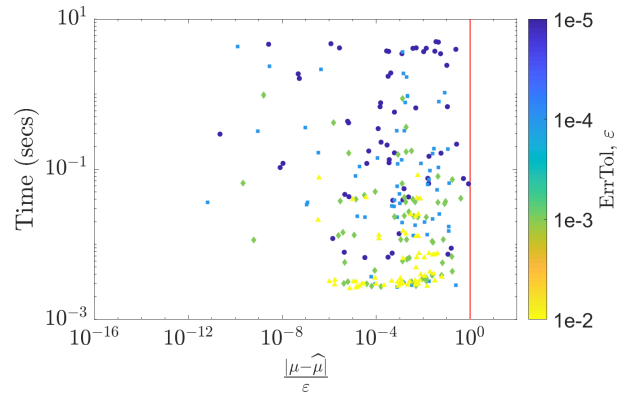

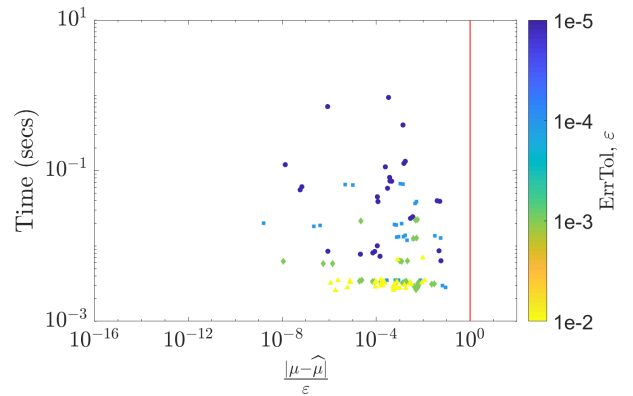Fig. 11: Keister function integrated within ε and finite time

Fig. 12: MVN integrated within ε and finite time

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
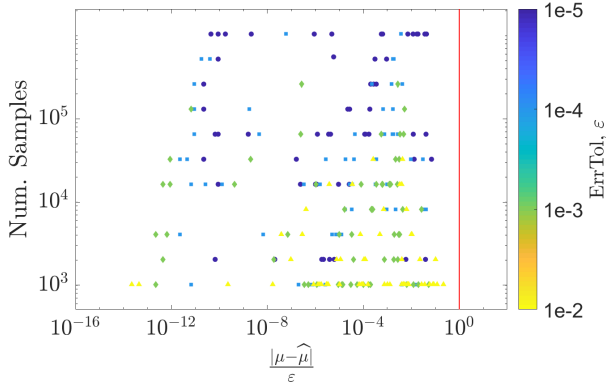
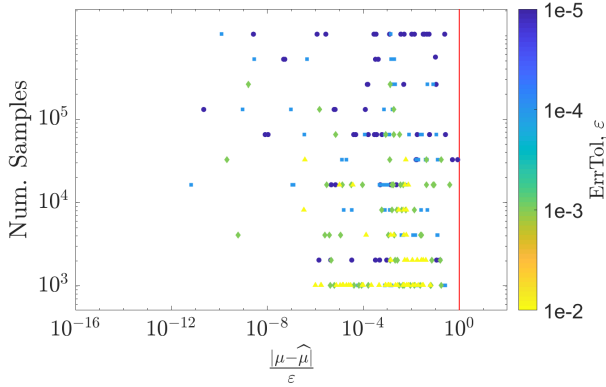Fig. 13: Exp(Cos) integrated within ε using finite number of samples

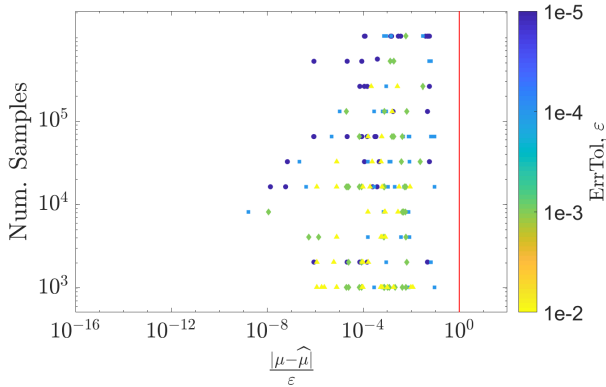Fig. 14: Keister integrated within ε using finite number of samples

Fig. 15: MVN integrated within ε using finite number of samples

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

$$\tilde{f}(\boldsymbol{x}) = \tilde{g}(\boldsymbol{\psi}(\boldsymbol{x}))$$
$$= \sum w_i C(.,.)$$

4. Deterministic interpretation of Bayesian cubature

7 Appendix

7.1 Properties of Multivariate Normal Distributions

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

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

Proof : The inverse of the matrix C may be partitioned as [?]

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

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

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

Since $\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$ have a Gaussian distribution, the conditional distribution of $\boldsymbol{Y}_1 | \boldsymbol{Y}_2$ is also Gaussian. So, it is sufficient to prove that $\mathbb{E}(\boldsymbol{Y}_1 | \boldsymbol{Y}_2)$ and $\mathrm{cov}(\boldsymbol{Y}_1 | \boldsymbol{Y}_2)$ are as hypothesized.

Define $\boldsymbol{Z} = \boldsymbol{Y}_1 - \mathsf{B}\boldsymbol{Y}_2$ where $\mathsf{B} = \mathsf{C}_{21}^T\mathsf{C}^{-1}{}_{22}$. Note that $\mathbb{E}(\boldsymbol{Z}) = \boldsymbol{m}_1 - \mathsf{B}\boldsymbol{m}_2$. Elementary calculations establish that $\boldsymbol{Z}$ and $\boldsymbol{Y}_2$ are uncorrelated and therefore independent. Therefore it follows that the conditional mean is

$$\mathbb{E}(\boldsymbol{Y}_1 | \boldsymbol{Y}_2) = \mathbb{E}(\boldsymbol{Z} + \mathsf{B}\boldsymbol{Y}_2 | \boldsymbol{Y}_2)$$
$$= \mathbb{E}(\boldsymbol{Z} | \boldsymbol{Y}_2) + \mathbb{E}(\mathsf{B}\boldsymbol{Y}_2 | \boldsymbol{Y}_2) = \mathbb{E}(\boldsymbol{Z}) + \mathsf{B}\boldsymbol{Y}_2$$
$$= \boldsymbol{m}_1 + \mathsf{B}(\boldsymbol{Y}_2 - \boldsymbol{m}_2)$$
$$= \boldsymbol{m}_1 + \mathsf{C}_{21}^T\mathsf{C}^{-1}{}_{22}(\boldsymbol{Y}_2 - \boldsymbol{m}_2)$$

Likewise the conditional covariance is

$$\mathrm{cov}(\boldsymbol{Y}_1 | \boldsymbol{Y}_2) = \mathrm{cov}(\boldsymbol{Z} + \mathsf{B}\boldsymbol{Y}_2 | \boldsymbol{Y}_2)$$
$$= \mathrm{cov}(\boldsymbol{Z}) = \mathrm{var}(\boldsymbol{Y}_1 - \mathsf{B}\boldsymbol{Y}_2)$$
$$= \mathrm{cov}(\boldsymbol{Y}_1) + \mathsf{B}\mathrm{cov}(\boldsymbol{Y}_2)\mathsf{B}^T$$
$$\qquad - 2\mathrm{cov}(\boldsymbol{Y}_1, \boldsymbol{Y}_2)\mathsf{B}^T$$
$$= \mathsf{C}_{11} + \mathsf{C}_{21}^T\mathsf{C}_{22}^{-1}\mathsf{C}_{22}\mathsf{C}_{22}^{-1}\mathsf{C}_{21} - 2\mathsf{C}_{21}^T\mathsf{C}_{22}^{-1}\mathsf{C}_{21}$$
$$= \mathsf{C}_{11} - \mathsf{C}_{21}^T\mathsf{C}_{22}^{-1}\mathsf{C}_{21}.$$

This completes the proof. $\square$

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. DLMF: Nist digital library of mathematical functions. http://dlmf.nist.gov/ **Part 2**, Release 1.0.5 of 2012–10–01 (2012)
6. Genz, A.: Comparison of methods for the computation of multivariate normal probabilities. Computing Science and Statistics **25**, 400 – 405 (1993)
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. Rasmussen, C.E., Ghahramani., Z.: Bayesian monte carlo. Advances in Neural Information Processing Systems pp. 489–496 (2003)
14. Rasmussen, C.E., Williams, C.: Gaussian Processes for Machine Learning. MIT Press, Cambridge, Massachusetts (2006). (online version at http://www.gaussianprocess.org/gpml/)
15. Ritter, K.: Average-case analysis of numerical problems. Lecture Notes in Mathematics, Springer-Verlag, Berlin **vol. 1733**, 163–175 (2000)