**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 $\mu = \int_{\mathbb{R}^d} g(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$, of a multi-dimensional function $g$ where $\mu$ has no closed form analytic expression. Typically, $g$ 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 $\mu$ may often be expressed in the form

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

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

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

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.,

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

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 Diaconis[3], O'Hagan[12], Ritter[14], 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-

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

S. Author
second address

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 [8] 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[9,10] 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 $\boldsymbol{\theta}$:

$$\mathsf{C}^T = \mathsf{C}, \qquad \boldsymbol{a}^T \mathsf{C} \boldsymbol{a} > 0 \quad \forall \boldsymbol{a} \neq \boldsymbol{0},$$
$$\text{where } \mathsf{C} = (C_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j=1}^n,$$
$$\forall n \in \mathbb{N},\ \boldsymbol{t}, \boldsymbol{x}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in [0,1]^d. \quad (4)$$

The covariance function, $C$, and the Gram matrix, $\mathsf{C}$ 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

$$\boldsymbol{f} \sim \mathcal{N}(m\boldsymbol{1}, s^2 \mathsf{C}), \quad (5a)$$
$$\mu \sim \mathcal{N}(m, s^2 c_0), \quad (5b)$$
$$\text{where } c_0 = \int_{[0,1]^{2d}} C_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{t})\, \mathrm{d}\boldsymbol{x}\, \mathrm{d}\boldsymbol{t}, \quad (5c)$$
$$\mathrm{cov}(\boldsymbol{f}, \mu) = \left( \int_{[0,1]^d} C(\boldsymbol{t}, \boldsymbol{x}_i)\, \mathrm{d}\boldsymbol{t} \right)_{i=1}^n =: \boldsymbol{c}. \quad (5d)$$

Therefore, it follows from Lemma 1 ???? that the *conditional* distribution of the integral given the 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

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

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:

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

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

yields

$$m_{\mathrm{MLE}} = \frac{\mathbf{1}^T \mathsf{C}^{-1} \boldsymbol{y}}{\mathbf{1}^T \mathsf{C}^{-1} \mathbf{1}} \tag{10}$$

$$\begin{aligned} s^2_{\mathrm{MLE}} &= \frac{1}{n}(\boldsymbol{y} - m_{\mathrm{MLE}}\mathbf{1})^T \mathsf{C}^{-1}(\boldsymbol{y} - m_{\mathrm{MLE}}\mathbf{1}) \\ &= \frac{1}{n}\boldsymbol{y}^T \left[\mathsf{C}^{-1} - \frac{\mathsf{C}^{-1}\mathbf{1}\mathbf{1}^T\mathsf{C}^{-1}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}\right]\boldsymbol{y}, \end{aligned} \tag{11}$$

$$\begin{aligned} \boldsymbol{\theta}_{\mathrm{MLE}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \Bigg\{ & \frac{1}{n}\log(\det \mathsf{C}) \qquad (12) \\ & + \log\left(\boldsymbol{y}^T \left[\mathsf{C}^{-1} - \frac{\mathsf{C}^{-1}\mathbf{1}\mathbf{1}^T\mathsf{C}^{-1}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}\right]\boldsymbol{y}\right)\Bigg\}. \qquad (13) \end{aligned}$$

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

$$\widehat{\mu}_{\mathrm{MLE}} = \left(\frac{(1 - \mathbf{1}^T\mathsf{C}^{-1}\boldsymbol{c})}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}\mathbf{1} + \boldsymbol{c}\right)^T \mathsf{C}^{-1}\boldsymbol{y}, \tag{14}$$

$$\widehat{\sigma}^2_{\mathrm{MLE}} := \boldsymbol{y}^T \left[\mathsf{C}^{-1} - \frac{\mathsf{C}^{-1}\mathbf{1}\mathbf{1}^T\mathsf{C}^{-1}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}\right]\boldsymbol{y} \times (c_0 - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c}),$$

$$\mathbb{P}_f[|\mu - \widehat{\mu}| \le 2.58\widehat{\sigma}_{\mathrm{MLE}}] \ge 99\%. \tag{15}$$

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

### *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

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

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 $\xi$, $\lambda$, or $z$ can be discarded since we only need to preserve the proportion, however, factors that depend on $\xi$, $\lambda$, 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 compute the integral with respect to $\xi$:

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

Finally, we simplify the key term:

$$\begin{aligned} \alpha\gamma - \beta^2 &= \mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}(c_0 - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c})(z - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{y})^2 \\ &\quad - 2\mathbf{1}^T\mathsf{C}^{-1}\boldsymbol{y}(c_0 - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c})(1 - \boldsymbol{c}^T\mathsf{C}^{-1}\mathbf{1})(z - \boldsymbol{c}^T\mathsf{C}^{-1} \\ &\quad + (1 - \boldsymbol{c}^T\mathsf{C}^{-1}\mathbf{1})^2\boldsymbol{y}^T\mathsf{C}^{-1}\boldsymbol{y}(c_0 - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c}) \\ &\quad + [\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}\boldsymbol{y}^T\mathsf{C}^{-1}\boldsymbol{y} - (\mathbf{1}^T\mathsf{C}^{-1}\boldsymbol{y})^2](c_0 - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c})^2 \\ &\propto \mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}\left(z - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{y} - \frac{(1 - \boldsymbol{c}^T\mathsf{C}^{-1}\mathbf{1})\mathbf{1}^T\mathsf{C}^{-1}\boldsymbol{y}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}\right)^2 \\ &\quad - \frac{[(1 - \boldsymbol{c}^T\mathsf{C}^{-1}\mathbf{1})\mathbf{1}^T\mathsf{C}^{-1}\boldsymbol{y}]^2}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}} + (1 - \boldsymbol{c}^T\mathsf{C}^{-1}\mathbf{1})^2\boldsymbol{y}^T\mathsf{C}^{-1} \\ &\quad (c_0 - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c})[\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}\boldsymbol{y}^T\mathsf{C}^{-1}\boldsymbol{y} - (\mathbf{1}^T\mathsf{C}^{-1}\boldsymbol{y})^2] \\ &\propto \left(z - \left[\frac{(1 - \boldsymbol{c}^T\mathsf{C}^{-1}\mathbf{1})\mathbf{1}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}} + \boldsymbol{c}\right]^T \mathsf{C}^{-1}\boldsymbol{y}\right)^2 \\ &\quad + \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] \times \boldsymbol{y}^T\left[\mathsf{C}^{-1} - \frac{\mathsf{C}^{-1}}{\mathbf{1}^T} \right. \\ &\propto (z - \widehat{\mu}_{\mathrm{full}})^2 + (n-1)\sigma^2_{\mathrm{full}} \\ &\propto \left(1 + \frac{1}{n-1}\frac{(z - \mu_{\mathrm{full}})^2}{\widehat{\sigma}^2_{\mathrm{full}}}\right) \end{aligned}$$

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 cubature rule is the same as in the empirical Bayes case in (14):

$$\widehat{\mu}_{\mathrm{full}}|(\boldsymbol{f} = \boldsymbol{y}) = \left[\frac{(1 - \boldsymbol{c}^T\mathsf{C}^{-1}\mathbf{1})\mathbf{1}^T}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}} + \boldsymbol{c}^T\right]\mathsf{C}^{-1}\boldsymbol{y}, \tag{16}$$

$$\begin{aligned} \widehat{\sigma}^2_{\mathrm{full}}|(\boldsymbol{f} = \boldsymbol{y}) = \frac{1}{n-1}&\left[\frac{(1 - \boldsymbol{c}^T\mathsf{C}^{-1}\mathbf{1})^2}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}} + (c_0 - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c})\right] \\ &\times \quad \boldsymbol{y}^T\left[\mathsf{C}^{-1} - \frac{\mathsf{C}^{-1}\mathbf{1}\mathbf{1}^T\mathsf{C}^{-1}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}\right]\boldsymbol{y}. \end{aligned}$$

But the variance is different. So, the stopping criterion for the full Bayes case,

$$\begin{aligned} n = \min\Bigg\{ n_j \in \mathbb{Z}_{>0} : \frac{t^2_{n_j - 1, 0.995}}{n_j - 1}&\left[\frac{(1 - \boldsymbol{c}^T\mathsf{C}^{-1}\mathbf{1})^2}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}} + (c_0 - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c})\right] \\ &\times \boldsymbol{y}^T\left[\mathsf{C}^{-1} - \frac{\mathsf{C}^{-1}\mathbf{1}\mathbf{1}^T\mathsf{C}^{-1}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}\right]\boldsymbol{y} \le \varepsilon^2\Bigg\}. \end{aligned} \tag{17}$$

is more conservative than that in the empirical Bayes case, (**??**). 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, it typically cannot be practically considered as a hyperparameter in a full Bayesian model. The empirical Bayes approach for determining $\boldsymbol{\theta}$ in (12) is one possibility

*2.2.3 Cross-validation*

Another alternative to determining $m$, $s$, and $\boldsymbol{\theta}$ is *leave-one-out cross-validation*. Let $\widetilde{y}_i$ denote the expected value of $f(\boldsymbol{x}_i)$ conditioned on $\boldsymbol{f}_{-i} = \boldsymbol{y}_{-i}$, where the subscript $-i$ denotes vector excluding the $i^{\text{th}}$ component. The cross-validation criterion, which is to be minimized, is

$$\text{CV} = \sum_{i=1}^{n} (y_i - \widetilde{y}_i)^2. \tag{18}$$

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

$$\mathsf{C} = \begin{pmatrix} C_{ii} & \boldsymbol{C}_{-i,i}^T \\ \boldsymbol{C}_{-i,i} & \mathsf{C}_{-i,-i} \end{pmatrix}, \qquad \mathsf{A} = \begin{pmatrix} A_{ii} & \boldsymbol{A}_{-i,i}^T \\ \boldsymbol{A}_{-i,i} & \mathsf{A}_{-i,-i} \end{pmatrix}, \qquad \boldsymbol{\zeta} = \begin{pmatrix} \zeta_i \\ \boldsymbol{\zeta}_{-i} \end{pmatrix}$$

Note that in this notation, Lemma 1 implies that

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

Thus, (20) may be re-written as

$$\text{CV} = \sum_{i=1}^{n} \left( \frac{\zeta_i}{A_{ii}} \right)^2, \qquad \boldsymbol{\zeta} = \mathsf{C}^{-1}(\boldsymbol{y} - m\mathbf{1}) \tag{19}$$

Wahba recommended a generalized cross-validation criterion, which replaces the $i^{\text{th}}$ diagonal element of $\mathsf{A}$ in the denominator with the average diagonal element of $\mathsf{A}$:

$$\text{GCV} = \frac{\sum_{i=1}^{n} \zeta_i^2}{\left( \frac{1}{n} \sum_{i=1}^{n} A_{ii} \right)^2} = \frac{(\boldsymbol{y} - m\mathbf{1})^T \mathsf{C}^{-2} (\boldsymbol{y} - m\mathbf{1})}{\left( \frac{1}{n} \text{trace}(\mathsf{C}^{-1}) \right)^2}. \tag{20}$$

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

$$m_{\text{GCV}} = \frac{\mathbf{1}^T \mathsf{C}^{-2} \boldsymbol{y}}{\mathbf{1}^T \mathsf{C}^{-2} \mathbf{1}}, \tag{21}$$

$$\boldsymbol{\theta}_{\text{GCV}} = \underset{\boldsymbol{\theta}}{\text{argmin}} \log \left( \boldsymbol{y}^T \left[ \mathsf{C}^{-2} - \frac{\mathsf{C}^{-2} \mathbf{1} \mathbf{1}^T \mathsf{C}^{-2}}{\mathbf{1}^T \mathsf{C}^{-2} \mathbf{1}} \right] \boldsymbol{y} \right) - 2 \log \left( \text{trace}(\mathsf{C}^{-1}) \right) \tag{22}$$

2.3 Formulating the automatic Bayesian cubature algorithm

Let the function to integrate be $f \sim \mathcal{GP}(m, s^2 C_{\boldsymbol{\theta}})$. Our goal is to compute $\hat{\mu}$ within the error tolerance $\varepsilon$, i.e., $|\mu - \hat{\mu}| \leq \varepsilon$.

It is not possible to know true error $|\mu - \hat{\mu}|$ for real problems. So, the error-bound $\text{err}_n$ is used as the approximate error estimate. Basic principle of our algorithm is to keep adding more function values till the stopping criterion is met, i.e., $\text{err}_n \leq \varepsilon$. Once the stopping criterion is met, the approximation of $\mu$ can be computed:

$$\widehat{\mu}_n = w_0 + \boldsymbol{w}^T \boldsymbol{y}, \tag{23}$$

Where the suffix $n$ is used to imply the number of integrand-samples used. In every iteration of the *automatic cubature algorithm (1)*, we only need to numerically estimate the MLE of $\boldsymbol{\theta}$ and then use it to compute $\mathsf{C}$ and $\text{err}_n$.

The following pseudo-code briefly explains the working of the automatic cubature algorithm.

**Algorithm 1**

**procedure** AUTOCUBATURE($f, \varepsilon$) ▷ Integrate within the error tolerance
  $n_0 \leftarrow 2^8$ ▷ start with minimum number of points
  $n \leftarrow n_0$, $n' \leftarrow 0$
  **while** true **do** ▷ Iterate till error tolerance is met
    Generate $\{\boldsymbol{x}_i\}_{i=n'+1}^{n}$ and sample $\{f(x_i)\}_{i=n'+1}^{n}$
    Compute error bound $err_n$ ▷ $\text{err}_n$ data driven error bound
    **if** $err_n \leq \varepsilon$ **then break**
    **end if**
    $n' \leftarrow n, n \leftarrow 2 \times n'$
  **end while**
  Compute cubature weights $\{w_i\}_{i=1}^{n}$
  Compute approximate integral $\widehat{\mu}_n$
  **return** $\widehat{\mu}_n$ ▷ Integral estimate $\widehat{\mu}_n$
**end procedure**

As shown, the algorithm continues the iteration loop till the stopping-criterion is met, i.e., $\text{err}_n$ is smaller than error threshold $\varepsilon$. At the end of every iteration, if the computed error bound $\text{err}_n$ is higher than the required $\varepsilon$, algorithm doubles the number of points $n$ and repeats. When the error tolerance $\varepsilon$ is met, exits the loop. Finally using the $n$ and the MLE estimated parameters, computes the $\widehat{\mu}_n$.

Accurately computing the data-driven error bound $\text{err}_n$ which closely matches the true error is essential for the effectiveness of our algorithm. So, accurate and faster computation of $\text{err}_n$ and $\widehat{\mu}_n$ are the main objective of the following sections.

### 2.4 Example with Matern kernel

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:

$$C_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{t}) = \prod_{k=1}^{d} \exp(-\theta_k|\boldsymbol{x}_k - \boldsymbol{t}_k|)(1+\theta_k|\boldsymbol{x}_k - \boldsymbol{t}_k|) \tag{24}$$

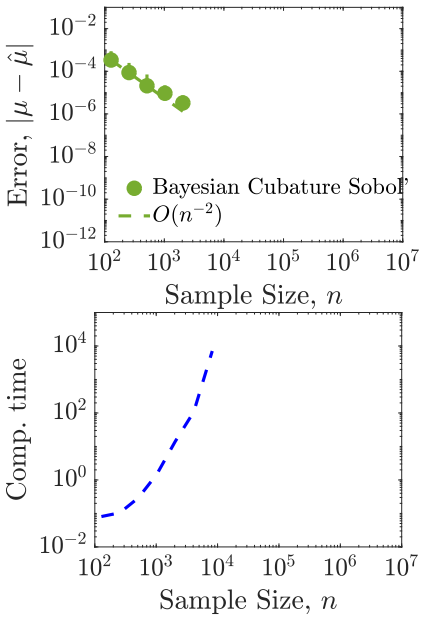


Fig. 1: MVN for d=2 with Matern kernel

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

Automatic Bayesian cubature algorithm described (Section 2.3) is slow due to its computationally intensive nature. This could be a hindrance in real world applications when a larger number of data points are needed to achieve the desired accuracy. This section and further sections explore techniques to make it faster.

Bayesian cubature algorithm uses error-bound $\mathrm{err}_n$ to decide when to stop, and uses MLE loss function (12) to find the optimal shape parameter. These computations involve covariance matrix inversions and multiplications, which are time-consuming and prone to numerical error. When the algorithm tries to adapt by increasing $n$ to get better accuracy, the covariance matrix size also grows, along with it, the matrix's condition number also grows significantly causing numerical error in matrix operations. This phenomenon is called ill-conditioning.

One approach to overcome these issues could be to use an efficient method to compute the whole equation without incurring the ill-conditioning, which involves, for example, using *stable computation* [5]. But it is going to be still slower and time-consuming.

Another approach for stable and faster computation is, using a carefully chosen kernel with some special properties which leads to faster matrix operations, like faster matrix inversion and multiplication. This approach can be designed to be faster in computation while avoiding the numerical error due to ill-conditioning. The later approach is the major objective of this research work. Our approach is especially to choose a special form of kernel, which is called *fast transform kernel* along with a suitable pointset, that will avoid expensive matrix operations.

### 3.1 Fast transform kernel

Suppose the domain is $\mathcal{X} = [0,1]^d$ and the probability measure is uniform, If the kernel $C_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{t})$ is chosen carefully along with appropriate point set $\{\boldsymbol{x}_i\}_{i=1}^n$, have the special properties, so the resulting Gram matrix has factorization of the form:

$$\mathsf{C} = \Big(C_{\boldsymbol{\theta}}(\boldsymbol{x}_i,\boldsymbol{x}_j)\Big)_{i,j=1}^{n} = (\boldsymbol{C}_1,...,\boldsymbol{C}_n)$$
$$= \frac{1}{n}\mathsf{V}\Lambda\mathsf{V}^H = \frac{1}{n}\mathsf{V}^*\Lambda\mathsf{V}^T, \qquad \mathsf{V}^H = n\mathsf{V}^{-1}, \tag{25}$$

Where $V^H$ is the Hermitian of $V$,

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

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

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

Suppose the transform $\hat{\boldsymbol{z}} = \mathsf{V}^T\boldsymbol{z}$ for any arbitrary $\boldsymbol{z}$ can be computed within the computational cost of $\mathcal{O}(n\log n)$, we call it a *Fast transform*. Consequently $C_{\boldsymbol{\theta}}$ is called a *fast transform kernel*. We use the notation $\hat{\boldsymbol{z}}$ to indicate the transform of $\boldsymbol{z}$. The properties of fast

transform can be leveraged to make the computations in Bayesian cubature algorithm faster. To begin with, $\boldsymbol{\lambda}$ can be computed faster by simply,

$$\mathsf{V}^T\boldsymbol{C}_1 = \mathsf{V}^T\left(\frac{1}{n}\mathsf{V}^*\Lambda\boldsymbol{v}_1\right) = \underbrace{\left(\frac{1}{n}\mathsf{V}^T\mathsf{V}^*\right)}\Lambda\boldsymbol{v}_1 = \Lambda\mathbf{1} = \begin{pmatrix}\lambda_1\\ \vdots\\ \lambda_n\end{pmatrix} = \boldsymbol{\lambda} \quad (26)$$

Most of the computations in the algorithm are of the form $\boldsymbol{a}^T\mathsf{C}^{-1}\boldsymbol{b}$, can be simplified easily:

$$\mathsf{C}V_1 = \lambda_1 V_1, \quad \text{by definition}$$

$$\text{So,} \quad \mathsf{C}^{-1}\mathbf{1} = \mathsf{C}^{-1}V_1 = \frac{V_1}{\lambda_1} = \frac{\mathbf{1}}{\lambda_1},$$

Similarly,

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

$$\boldsymbol{a}^T\mathsf{C}^{-2}\boldsymbol{b} = \frac{1}{n}\boldsymbol{a}^T\mathsf{V}\Lambda^{-2}\mathsf{V}^H\boldsymbol{b} = \frac{1}{n}\widehat{\boldsymbol{a}}^T\Lambda^{-2}\widehat{\boldsymbol{b}}^* = \frac{1}{n}\sum_{i=1}^n\frac{\widehat{a}_i\widehat{b}_i^*}{\lambda_i^2},$$

where $\widehat{\boldsymbol{a}} = \mathsf{V}^T\boldsymbol{a}$ and $\widehat{\boldsymbol{b}} = \mathsf{V}^T\boldsymbol{b}$. By using this, some of the recurring terms in the algorithm are simplified,

$$\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1} = \mathbf{1}^T\left(\frac{\mathbf{1}}{\lambda_1}\right) = \frac{n}{\lambda_1}$$

$$\mathbf{1}^T\mathsf{C}^{-1}\boldsymbol{y} = \mathbf{1}^T\left(\frac{\boldsymbol{y}}{\lambda_1}\right) = \frac{\sum_{i=1}^n y_i}{\lambda_1} = \frac{\widehat{y}_1}{\lambda_1}$$

$$\boldsymbol{y}^T\mathsf{C}^{-1}\boldsymbol{y} = \frac{1}{n}\sum_{i=1}^n\frac{|\widehat{y}_i|^2}{\lambda_i}$$

$$\boldsymbol{c}^T\mathsf{C}^{-1}\mathbf{1} = \boldsymbol{c}^T\left(\frac{\mathbf{1}}{\lambda_1}\right) = \frac{\widehat{c}_1}{\lambda_1}$$

$$\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c} = \frac{1}{n}\sum_{i=1}^n\frac{|\widehat{c}_i|^2}{\lambda_i}$$

$$\boldsymbol{y}^T\mathsf{C}^{-2}\boldsymbol{y} = \frac{1}{n}\sum_{i=1}^n\frac{|\widehat{y}_i|^2}{\lambda_i^2}$$

Then the MLE estimates can be simplified using these results:

$$m_{\text{MLE}} = \frac{\mathbf{1}^T\mathsf{C}^{-1}\boldsymbol{y}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}} = \frac{\widehat{y}_1}{n} = \frac{1}{n}\sum_{i=1}^n y_i,$$

$$s^2_{\text{MLE}} = \frac{1}{n}\boldsymbol{y}^T\left[\mathsf{C}^{-1} - \frac{\mathsf{C}^{-1}\mathbf{1}\mathbf{1}^T\mathsf{C}^{-1}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}\right]\boldsymbol{y} = \frac{1}{n}\left[\frac{1}{n}\sum_{i=1}^n\frac{|\widehat{y}_i|^2}{\lambda_i} - \frac{|\widehat{y}_1|^2/\lambda_1^2}{n/\lambda_1}\right] = \frac{1}{n^2}\sum_{i=2}^n\frac{|\widehat{y}_i|^2}{\lambda_i}$$

## 3.2 Application of fast transform kernel

Using the properties of the fast transform, matrix multiplications and inversions can be avoided, so the overall computation cost is $\mathcal{O}(n\log n)$. To begin with, MLE estimate of $\boldsymbol{\theta}$ can be made faster:

### *3.2.1 Empirical Bayes*

$$\boldsymbol{\theta}_{\text{MLE}} = \operatorname*{argmin}_{\boldsymbol{\theta}}\left[\frac{1}{n}\log(\det\,\mathsf{C}) + \log\left(s^2_{\text{MLE}}\right)\right] = \operatorname*{argmin}_{\boldsymbol{\theta}}\left[\frac{1}{n}\sum_{i=1}^n\log(\lambda_i) + \log\left(\frac{1}{n^2}\sum_{i=2}^n\frac{|\widehat{y}_i|^2}{\lambda_i}\right)\right] \quad (27)$$

$$\text{where} \quad \hat{\boldsymbol{y}} = (\hat{y}_i)_{i=1}^n = \mathsf{V}^T\boldsymbol{y}.$$

Similarly, the error bound $\text{err}_n$ computation can be made faster:

$$\text{err}_n = 2.58\sqrt{\frac{1}{n}\boldsymbol{y}^T\left[\mathsf{C}^{-1} - \frac{\mathsf{C}^{-1}\mathbf{1}\mathbf{1}^T\mathsf{C}^{-1}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}\right]\boldsymbol{y}(c_0 - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c})} = 2.58\sqrt{\frac{1}{n^2}\sum_{i=2}^n\frac{|\widehat{y}_i|^2}{\lambda_i}\left(c_0 - \frac{1}{n}\sum_{i=1}^n\frac{|\widehat{c}_i|^2}{\lambda_i}\right)},$$

$$\text{where} \quad \widehat{\boldsymbol{y}} = \mathsf{V}^T\boldsymbol{y}, \widehat{\boldsymbol{c}} = \mathsf{V}^T\boldsymbol{c}.$$

Finally, the computation of $\widehat{\mu}_n$ is made faster:

$$\widehat{\mu}_{\text{MLE}}(f) = m_{\text{MLE}}[1 - \mathbf{1}^T\mathsf{C}^{-1}\boldsymbol{c}] + \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{y} = \frac{\widehat{y}_1}{n}\left[1 - \frac{\widehat{c}_1}{\lambda_1}\right] + \frac{1}{n}\sum_{i=1}^n\frac{\widehat{c}_i\widehat{y}_i^*}{\lambda_i} = \frac{\widehat{y}_1}{n} + \frac{1}{n}\sum_{i=2}^n\frac{\widehat{c}_i\widehat{y}_i^*}{\lambda_i} \quad (28)$$

It is interesting to note that, in general with no assumption on $m$, $\widehat{\mu}_{\text{MLE}}$ is simply the sample mean. Moreover the choice of kernel or any MLE estimated parameters do not affect the $\widehat{\mu}_n$ but influences the accuracy of the error bound $\text{err}_n$.

### *3.2.2 Full Bayes*

Generalized cross validation criterion:

$$\boldsymbol{\theta}_{\text{GCV}} = \operatorname*{argmin}_{\boldsymbol{\theta}}\left[\log\left(\frac{1}{n}\sum_{i=1}^n\frac{|\widehat{y}_i|^2}{\lambda_i^2} - \frac{\widehat{y}_1^2/\lambda_1^4}{n/\lambda_1^2}\right) - 2\log\left(\sum_{i=1}^n\frac{1}{\lambda_i}\right)\right] = \operatorname*{argmin}_{\boldsymbol{\theta}}\left[\log\left(\frac{1}{n}\sum_{i=2}^n\frac{|\widehat{y}_i|^2}{\lambda_i^2}\right) - 2\log\left(\sum_{i=1}^n\frac{1}{\lambda_i}\right)\right] \quad (29)$$

The cubature rule:

$$\widehat{\mu}_{\text{full}}|(\boldsymbol{f} = \boldsymbol{y}) = \left[\frac{(1 - \boldsymbol{c}^T\mathsf{C}^{-1}\mathbf{1})\mathbf{1}^T}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}} + \boldsymbol{c}^T\right]\mathsf{C}^{-1}\boldsymbol{y} = \frac{\widehat{y}_1}{n} + \frac{1}{n}\sum_{i=2}^n\frac{\widehat{c}_i\widehat{y}_i^*}{\lambda_i}, \quad (30)$$

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

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

The stopping criterion for the full Bayes case,

$$n = \min\Bigg\{n_j \in \mathbb{Z}_{>0} : \frac{t^2_{n_j-1,0.995}}{n_j(n_j-1)} \times \left[\frac{\lambda_1}{n_j}\left(1-\frac{\widehat{c}_1}{\lambda_1}\right)^2 + (c_0 - \frac{1}{n_j}\sum_{i=1}^{n}\frac{|\widehat{c}_i|^2}{\lambda_i})\right] \times \sum_{i=2}^{n}\frac{|\widehat{y}_i|^2}{\lambda_i} \le \varepsilon^2\Bigg\}. \tag{31}$$

These simplified computations involve no matrix inversion or multiplications. It just uses scalar divisions and multiplications so the computational cost is $\mathcal{O}(n)$. But computation of fast transform $\widehat{\boldsymbol{y}}$ is $\mathcal{O}(n\log n)$. Consequently the overall computation cost is $\mathcal{O}(n\log n)$.

## 4 Implementation 1: Using a shift invariant kernel for Bayesian cubature

We have established the concept of fast transform and showed how it can make the computations faster. But we assumed there exist a kernel that meets the requirements. Now we are going to show a example. The following shift invariant kernel satisfies the requirements of the fast transform kernel when combined with Rank-1 lattice points:

$$C_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{t}) := \sum_{\boldsymbol{k}\in\mathbb{Z}^d} \alpha_{\boldsymbol{k},\boldsymbol{\theta}} e^{2\pi\sqrt{-1}\boldsymbol{k}^T\boldsymbol{x}} e^{-2\pi\sqrt{-1}\boldsymbol{k}^T\boldsymbol{t}}, \tag{32}$$

where $\alpha_{\boldsymbol{k},\boldsymbol{\theta}} := \prod_{l=1}^{d}\frac{1}{\max(\frac{|k_l|}{\theta_l},1)^r_{\theta_l\le 1}}$, with $\alpha_{\mathbf{0},\boldsymbol{\theta}}=1$,

where $d$ is number of dimensions and $\alpha_{\boldsymbol{k}}$ is a scalar. The Gram matrix formed by this kernel is a Hermitian matrix. With some more simplifications:

$$C_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{t}) = \prod_{l=1}^{d}\left[1 + \sum_{k_l=1}^{\infty}\left|\frac{\theta_l}{k_l}\right|^r 2\cos(2\pi\sqrt{-1}k_l(x_l-t_l))\right].$$

The *shape parameter* $\boldsymbol{\theta}$ is used to make the kernel customizable. To be specific, the shape parameter tweaks the kernel, so that the function space spanned by the kernel closely resembles the space where the integrand function belongs.

This form of the kernel is very convenient to use in any analytical derivations or proofs, but not suitable for use with finite precision computers as this kernel involves infinite sum. It is preferred to have a simpler expression of the kernel without infinite sum for practical computations.

### 4.1 Using Lattice points

Along with the kernel (32), Rank-1 lattice points $\boldsymbol{x}_i \in \mathsf{L}_{n,\boldsymbol{h}}$ are used to get the *fast transform kernel*. In this work, we use the lattice points as defined in [15]:

$$\mathsf{L}_{n,\boldsymbol{h}} := \{\boldsymbol{x}_i := \boldsymbol{h}\frac{i-1}{n} \mod 1;\ \ i=1,\dots,n\},$$

Where $\boldsymbol{h}$ is the generating vector. Its dual lattice is defined as:

$$\mathsf{L}^{\perp}_{n,\boldsymbol{h}} := \{\boldsymbol{k}\in\mathbb{Z}^d : \boldsymbol{h}^T\boldsymbol{k} \equiv 0 \ \ (\mod n)\},$$

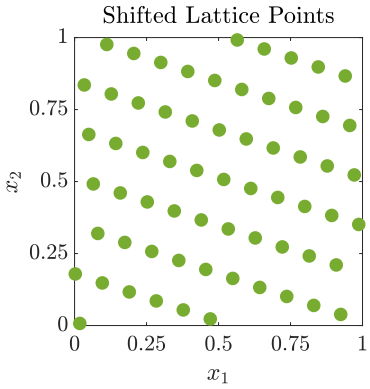


Fig. 2: Randomly shifted Lattice points in d=2

With lattice points, the kernel is written as:

$$C(\boldsymbol{x}_i,\boldsymbol{x}_j) = \sum_{\boldsymbol{k}\in\mathbb{Z}^d}\alpha_{\boldsymbol{k}} e^{2\pi\sqrt{-1}\boldsymbol{k}^T\frac{\boldsymbol{h}i}{n}} e^{-2\pi\sqrt{-1}\boldsymbol{k}^T\frac{\boldsymbol{h}j}{n}}$$

$$= \sum_{\boldsymbol{k}\in\mathbb{Z}^d}\alpha_{\boldsymbol{k}} e^{2\pi\sqrt{-1}\boldsymbol{k}^T\boldsymbol{h}\frac{(i-j)}{n}}$$

Please note that ' mod 1' need not be used explicitly in $C(\boldsymbol{x}_i,\boldsymbol{x}_j)$ since $e^{\pm 2\pi\sqrt{-1}}=1$. The kernel (32), when used with Rank-1 Lattice points, leads to symmetric and circulant Gram matrix $\mathsf{C}$. We can demonstrate the resulting Gram matrix satisfies the conditions of a fast transform.

### 4.2 Computing the kernel using Bernoulli polynomials

The shift-invariant-kernel (32) 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 [4] 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 (32) decay. The following very useful expansion is

referenced from [4] under eqn (24.8.3):

$$B_r(x) = \frac{-r!}{(2\pi\sqrt{-1})^r} \sum_{\substack{k\neq 0,\\ k=-\infty}}^{\infty} \frac{e^{2\pi\sqrt{-1}kx}}{k^r}$$

$$\begin{cases} \text{for } r = 1, & 0 < x < 1 \\ \text{for } r = 2, 3, \ldots & 0 \le x \le 1 \end{cases} \quad (33)$$

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 (33):

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

$$\text{for } -1 \le x \le 1$$

Comparing this expansion with (32), one can deduce, for $d = 1$:

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

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:

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

$$\text{where } 0 \le |x_l - t_l| \le 1, \quad (34)$$

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

We call (34) a Fourier kernel as the resulting Fast transform is Fast Fourier transform. Plots of this kernel in $d = 2$ is shown in Figure **??**.

### 4.3 Eigenvalues and Eigenvectors of $\mathsf{C}$

The kernel's Gram matrix $\mathsf{C}$ is circulant. It is generated by the vector:

$$(C_{\boldsymbol{\theta}}(\boldsymbol{x}_1, \boldsymbol{x}_1), C_{\boldsymbol{\theta}}(\boldsymbol{x}_2, \boldsymbol{x}_1), ..., C_{\boldsymbol{\theta}}(\boldsymbol{x}_n, \boldsymbol{x}_1))^T =: \boldsymbol{C}_1, \quad (35)$$

Which is the first row or column of $\mathsf{C}$ (since it is a symmetric matrix). By the properties of circulant matrix,

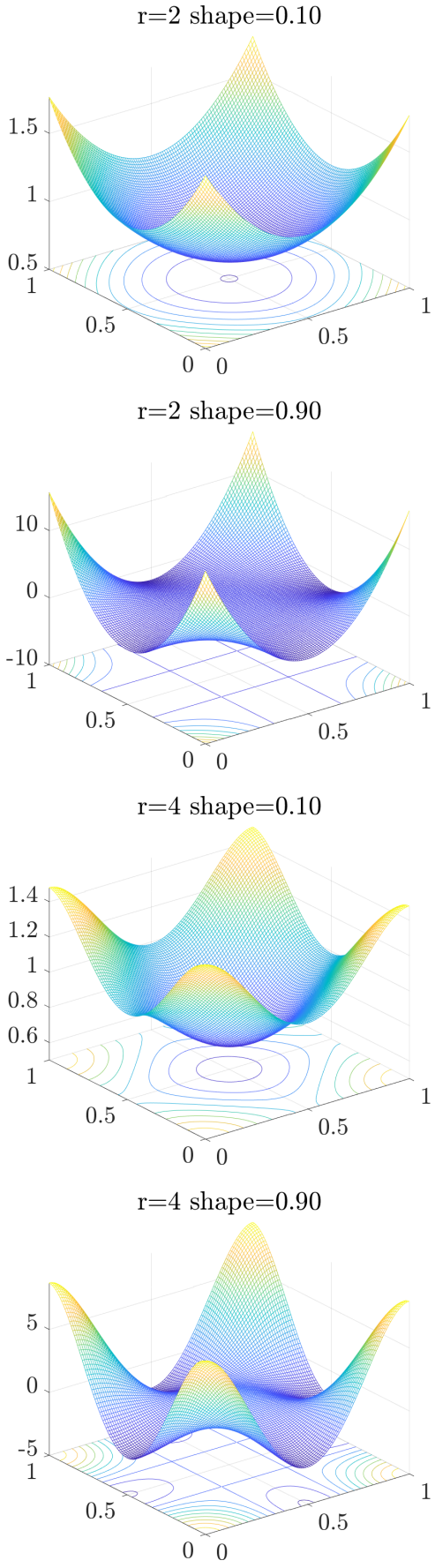


Fig. 3: Fourier kernel (34) with $r = 2, 4$ and $\theta = 0.1, 0.9$

the normalized eigenvectors are:

$$\left(1, e^{-2\pi\sqrt{-1}\frac{j}{n}}, e^{-2\pi\sqrt{-1}\frac{2j}{n}}, ..., e^{-2\pi\sqrt{-1}\frac{j(n-1)}{n}}\right)^T, \quad \text{for } j = 0, 1, ..., n-1.$$

The matrix constructed with these eigenvectors is:

$$\mathsf{V} := \left(e^{-2\pi\sqrt{-1}\frac{ij}{n}}\right)_{i,j=0}^{n-1},$$

Whereas, $\mathsf{V}^H := \frac{1}{n}\left(e^{2\pi\sqrt{-1}\frac{ij}{n}}\right)_{i,j=0}^{n-1}$,

Where the first column of $\mathsf{V}$, $\boldsymbol{V}_1 = \mathbf{1}$. The columns of $\mathsf{V}$ are complex exponential vectors independent of the kernel values. But their corresponding eigenvalues are:

$\Lambda = (\lambda_j)_{j=1}^n = \mathsf{V}^T\boldsymbol{C}_1 := \mathcal{DFT}\left\{\boldsymbol{C}_1\right\}$,

i.e., eigenvalues of the matrix $\mathsf{C}$ are computed by applying DFT over $\boldsymbol{C}_1$. for this kernel, the corresponding fast transform is *Fast Fourier Transform* (FFT). The kernel's Gram matrix $\mathsf{C}$ can be written in the factorized form:

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

This is the exact factorization used for the construction of fast transform. Another requirement $\boldsymbol{V}_1 = \mathbf{1}$ is also met. Additionally, computational cost of FFT is $\mathcal{O}(n \log n)$. Thus we can conclude the shift invariant kernel in eqn (32), obeys all the requirements to be considered as a fast transform kernel. So the operation $\mathsf{V}^T z$ is a fast transform. Numerical experiments using this kernel will be shown in the section (4.9).

## 4.4 Additional assumptions

The shift invariant kernel (32) has some more desirable properties, leading to $c_0 = 1$ and $\boldsymbol{c} = \mathbf{1}$, which will help to make the computations even faster:

$$c_0 = \int_{[0,1]^d\times[0,1]^d} C_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{t})\mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{t} = 1,$$

$$\boldsymbol{c} = \left(\int_{[0,1]^d} C_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{t})\mathrm{d}\boldsymbol{t}\right)_{i=1}^n = \mathbf{1}.$$

This gives simplified error bound:

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

$$= 2.58\sqrt{\frac{1}{n^2}\sum_{i=2}^n \frac{|\widehat{y}_i|^2}{\lambda_i}\left(1 - \frac{n}{\lambda_1}\right)},$$

where $\widehat{\boldsymbol{c}} = (n, 0, ..., 0)^T$, $\widehat{\boldsymbol{y}} = \mathsf{V}^T\boldsymbol{y}, \widehat{\boldsymbol{c}} = \mathsf{V}^T\boldsymbol{c}$.

And the cubature:

$$\widehat{\mu}_n(f) = w_0 + \boldsymbol{w}^T\boldsymbol{y} = \frac{\widehat{y}_1}{n} + \frac{1}{n}\sum_{i=2}^n \frac{\widehat{c}_i\widehat{y}_i^*}{\lambda_i}$$

$$= \frac{\widehat{y}_1}{n}, \quad \text{since} \quad \widehat{\boldsymbol{c}}_i = 0 \;\forall\; i \neq 0$$

### *4.4.1 Full Bayes*

Final version of full Bayes equations optimized using the properties of the shift invariant kernel are:

$$\widehat{\mu}_{\text{full}}|(\boldsymbol{f} = \boldsymbol{y}) = \frac{\widehat{y}_1}{n}, \tag{36}$$

$$\boldsymbol{\theta}_{\text{GCV}} = \underset{\boldsymbol{\theta}}{\text{argmin}}\left[\log\left(\frac{1}{n}\sum_{i=2}^n \frac{|\widehat{y}_i|^2}{\lambda_i^2}\right) - 2\log\left(\sum_{i=1}^n \frac{1}{\lambda_i}\right)\right] \tag{37}$$

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

$$= \frac{1}{n-1}\frac{1}{n}\sum_{i=2}^n \frac{|\widehat{y}_i|^2}{\lambda_i} \times \left(1 - \frac{n}{\lambda_1}\right)\left[\frac{\lambda_1}{n}\left(1 - \frac{n}{\lambda_1}\right) + 1\right]$$

$$= \frac{1}{n(n-1)}\sum_{i=2}^n \frac{|\widehat{y}_i|^2}{\lambda_i} \times \left(\frac{\lambda_1}{n} - 1\right)$$

The stopping criterion for the full Bayes case,

$$n = \min\Bigg\{n_j \in \mathbb{Z}_{>0} : \frac{t^2_{n_j-1,0.995}}{n_j(n_j-1)}\left(\frac{\lambda_1}{n_j} - 1\right) \times \sum_{i=2}^{n_j} \frac{|\widehat{y}_i|^2}{\lambda_i} \leq \varepsilon^2\Bigg\}. \tag{38}$$

## 4.5 Variable transforms

Fast transform kernel technique discussed with shift invariant kernel eqn(32) 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.

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

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

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

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

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

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.

### 4.6 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.

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

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:

$$\text{modified kernel} \quad \widetilde{C}(\boldsymbol{x}_i, \boldsymbol{x}_j) = C(\boldsymbol{x}_i, \boldsymbol{x}_j) - 1,$$

$$\text{Gram matrix} \quad \widetilde{\mathsf{C}} = \mathsf{C} - \mathbf{1}\mathbf{1}^T,$$

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

$$\lambda_1 \mathbf{1} = \mathsf{C}\mathbf{1} = (\widetilde{\mathsf{C}} + \mathbf{1}\mathbf{1}^T)\mathbf{1} = \tilde{\lambda}_1 \mathbf{1} + n\mathbf{1},$$

This shows $\tilde{\lambda}_1 = \lambda - n$. In fact for the rest of the values are:

$$\tilde{\lambda}_j = \lambda_j,\ \forall j = 1, ..., n-1. \tag{39}$$

Let $\boldsymbol{v}_j, \forall j = 0, ..., n-1$ be the eigenvectors of $\mathsf{C}$, similarly $\tilde{\boldsymbol{v}}_j, \forall j = 0, ..., n-1$ be the eigenvectors of $\widetilde{\mathsf{C}}$,
$\boldsymbol{v}_j^T \mathsf{C}\mathbf{1} = \lambda_1 \boldsymbol{v}_j^T \mathbf{1} = \mathbf{1}^T \mathsf{C} \boldsymbol{v}_j = \lambda_j \mathbf{1}^T \boldsymbol{v}_j$
Since $\lambda_1 \neq \lambda_j$, the above statement implies $\boldsymbol{v}_j^T \mathbf{1} = 0$. This interesting property provides the proof of the eqn. (44).
$\lambda_j \boldsymbol{v}_j = \mathsf{C}\boldsymbol{v}_j = 1_{n\times n}\boldsymbol{v}_j + \widetilde{\mathsf{C}}\boldsymbol{v}_j = \widetilde{\mathsf{C}}\boldsymbol{v}_j = \tilde{\lambda}_j \boldsymbol{v}_j$,
Thus proven. Using this result, cancellation error in the computation of $\mathrm{err}_n$ 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:

$$\begin{aligned}
d = 1: \quad & C_1 = 1 + \theta B(\{x_{i_1} - x_{j_1}\}) = 1 + \tilde{C}_1 \\
d = 2: \quad & C_2 = [1 + \theta B(\{x_{i_2} - x_{j_2}\})][1 + \tilde{C}_1] \\
& = 1 + \theta B(\{x_{i_2} - x_{j_2}\})[1 + \tilde{C}_1] + \tilde{C}_1 \\
& = 1 + \underbrace{\theta B(\{x_{i_2} - x_{j_2}\})C_1 + \tilde{C}_1}_{\tilde{C}_2} \\
d = 3 \quad & C_3 = [1 + \theta B(\{x_{i_3} - x_{j_3}\})][1 + \tilde{C}_2] \\
& = 1 + \theta B(\{x_{i_3} - x_{j_3}\})[1 + \tilde{C}_2] + \tilde{C}_2 \\
& = 1 + \theta B(\{x_{i_3} - x_{j_3}\})C_2 + \tilde{C}_2 \\
& \vdots \\
\forall d > 2 \quad & C_d = [1 + \theta B(\{x_{i_d} - x_{j_d}\})][1 + \tilde{C}_{d-1}] \\
& = 1 + \theta B(\{x_{i_d} - x_{j_d}\})[1 + \tilde{C}_{d-1}] + \tilde{C}_{d-1} \\
& = 1 + \theta B(\{x_{i_d} - x_{j_d}\})C_{d-1} + \tilde{C}_{d-1}
\end{aligned}$$

## 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 $\mathsf{C}$ 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,

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

And,

$$\begin{aligned}
\operatorname{cov}\left[\frac{1}{\sqrt{n}}\Lambda^{-\frac{1}{2}}\mathsf{V}^H\boldsymbol{f}\right] &= \frac{1}{n}\mathbb{E}\left[\Lambda^{-\frac{1}{2}}\mathsf{V}^H(\boldsymbol{f} - m\mathbf{1})(\boldsymbol{f} - m\mathbf{1})^T\mathsf{V}\Lambda^{-\frac{1}{2}}\right] \\
&= \frac{1}{n}\Lambda^{-\frac{1}{2}}\mathsf{V}^H\mathbb{E}\left[(\boldsymbol{f} - m\mathbf{1})(\boldsymbol{f} - m\mathbf{1})^T\right]\mathsf{V}\Lambda^{-\frac{1}{2}} \\
&= \frac{1}{n}\Lambda^{-\frac{1}{2}}\mathsf{V}^H\frac{1}{n}\mathsf{V}\Lambda\mathsf{V}^H\mathsf{V}\Lambda^{-\frac{1}{2}} \\
&= \mathsf{I}
\end{aligned}$$

Let $\boldsymbol{f}' = \frac{1}{\sqrt{n}}\Lambda^{-\frac{1}{2}}\mathsf{V}^H\boldsymbol{f}$,
$\boldsymbol{f}' \sim \mathcal{N}(m'\boldsymbol{e}_1, 1)$,
Where $m' = m\sqrt{\frac{n}{\lambda_1}}$. If we can verify the sample distribution of $\boldsymbol{f}'$ is approximately $\mathcal{N}(m'\boldsymbol{e}_1, 1)$ by using Normal plots , could validate our assumption.

## 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.

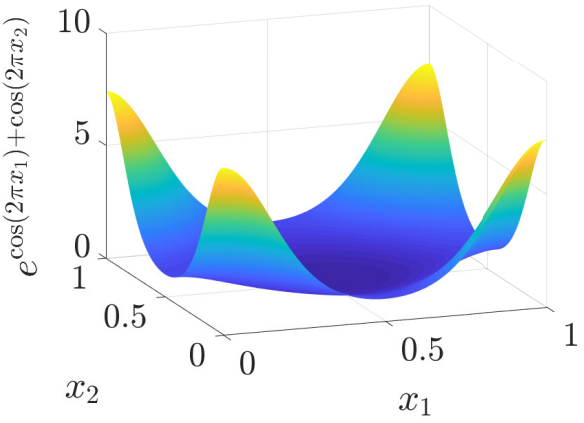


Fig. 4: Exp(Cos) in d=2

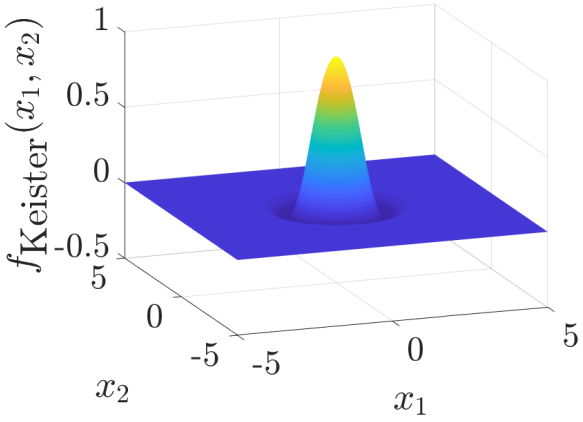


Fig. 5: Keister function in d=2

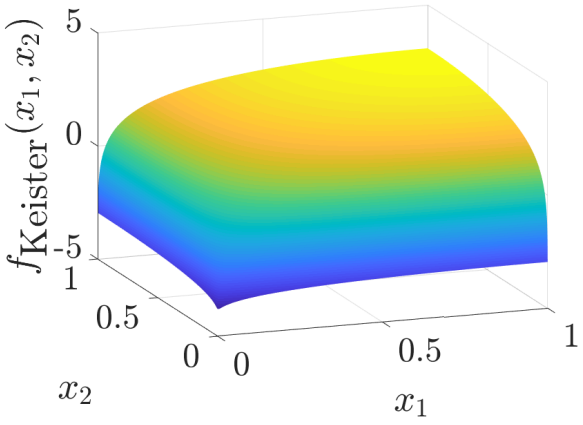


Fig. 6: Keister function transformed to $[0,1]^2$

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

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

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

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

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

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

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

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

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.

$$\begin{aligned}\mu &= \int_{[\boldsymbol{a},\boldsymbol{b}]\in\mathbb{R}^d} \frac{\exp\left(-\frac{1}{2}\boldsymbol{t}^T\Sigma^{-1}\boldsymbol{t}\right)}{\sqrt{(2\pi)^d\det(\Sigma)}}\,\mathrm{d}\boldsymbol{t} \\ &\overset{[6]}{=} \int_{[0,1]^{d-1}} f_{\text{Genz}}(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}\end{aligned}$$

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

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

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

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

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

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.

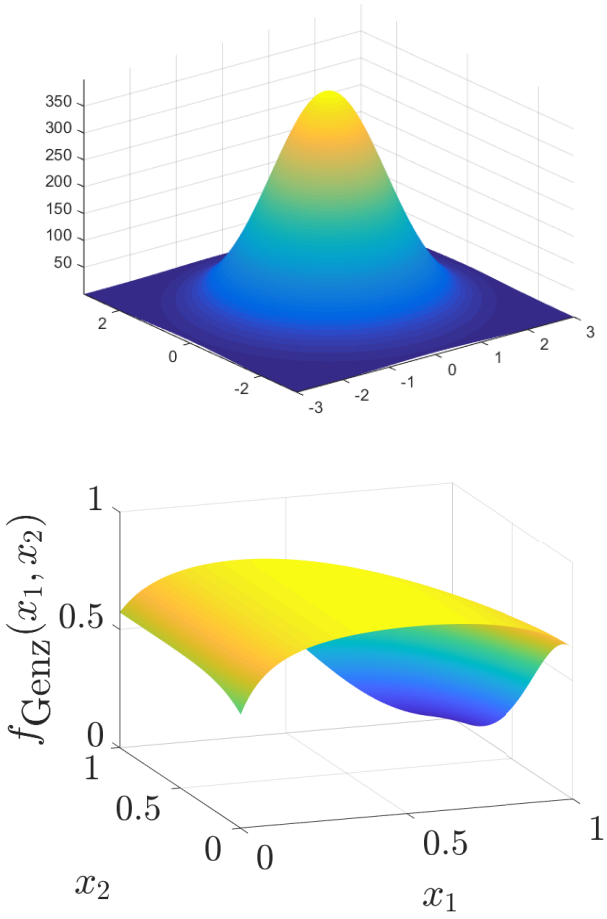


Fig. 7: Mutivariate Normal and Genz function

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

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

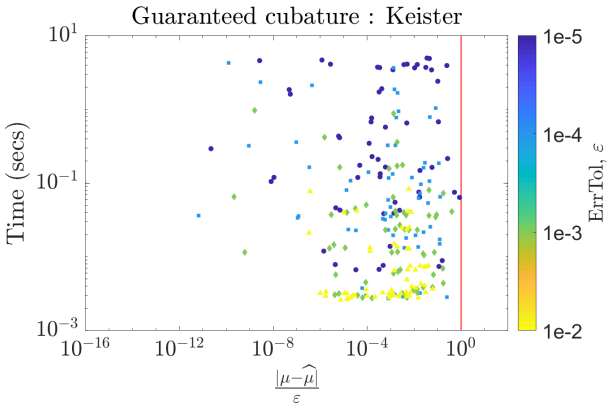


Fig. 8: Keister function integrated within the given threshold $\varepsilon$ and finite time

where the function payoff(.) describes 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 $\mathsf{L}$ 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 $T, S, S_0, K, r, \sigma$ 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 $\varepsilon$.

## 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 $\mathcal{O}(n \log n)$

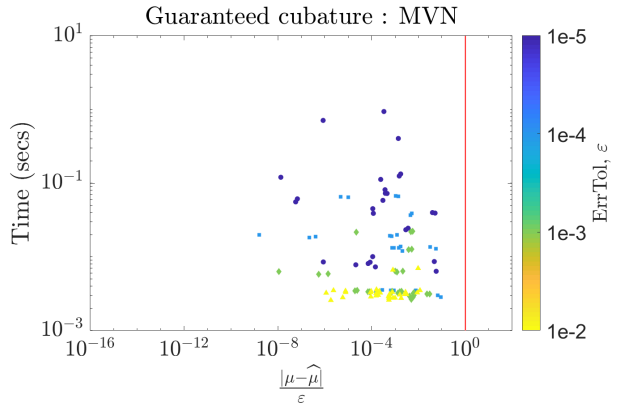


Fig. 9: MVN integrated within the given threshold $\varepsilon$ and finite time

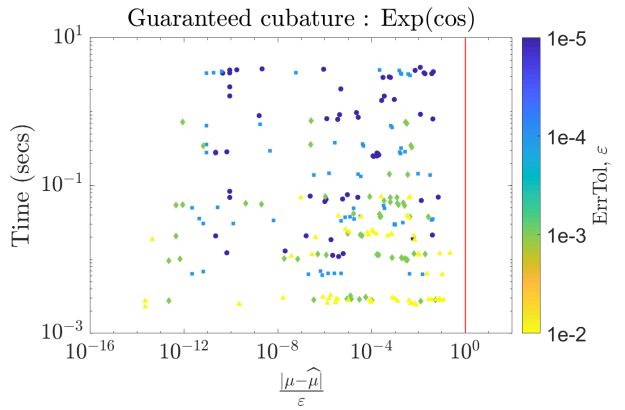


Fig. 10: Exp(Cos) integrated within the given threshold $\varepsilon$ and finite time

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 $\mathrm{err}_n$ still depends on the kernel order. So when we use higher order $r$, the error bound $\mathrm{err}_n$ 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 $\mathrm{err}_n$ 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

## 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*
$\boldsymbol{m} = \begin{pmatrix}\boldsymbol{m}_1 \\ \boldsymbol{m}_2\end{pmatrix} = \begin{pmatrix}\mathbb{E}(\boldsymbol{Y}_1) \\ \mathbb{E}(\boldsymbol{Y}_2)\end{pmatrix}$, $\mathsf{C} = \begin{pmatrix}\mathsf{C}_{11} & \mathsf{C}_{21}^T \\ \mathsf{C}_{21} & \mathsf{C}_{22}\end{pmatrix} = \begin{pmatrix}\mathrm{var}(\boldsymbol{Y}_1) & \mathrm{cov}( \\ \mathrm{cov}(\boldsymbol{Y}_2, \boldsymbol{Y}_1) & \mathrm{va}\end{pmatrix}$
*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* : Note that

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

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

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

Let's denote the first partition $\boldsymbol{Y}_1$ and the second $\boldsymbol{Y}_2$. 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

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

$$\operatorname{var}(\boldsymbol{Y}_1|\boldsymbol{Y}_2) = \mathsf{C}_{11} - \mathsf{C}_{21}^T\mathsf{C}_{22}^{-1}\mathsf{C}_{21}.$$

Now define $z = \boldsymbol{Y}_1 + A\boldsymbol{Y}_2$ where $A = -\mathsf{C}_{21}^T\mathsf{C}^{-1}{}_{22}$. We can write

$$\begin{aligned}\operatorname{cov}(z, \boldsymbol{Y}_2) &= \mathsf{C}_{21}^T + \operatorname{cov}(A\boldsymbol{Y}_2, \boldsymbol{Y}_2) \\ &= \mathsf{C}_{21}^T + A\operatorname{var}(\boldsymbol{Y}_2) \\ &= \mathsf{C}_{21}^T - \mathsf{C}_{21}^T\mathsf{C}^{-1}{}_{22}\mathsf{C}_{22} = 0\end{aligned}$$

Therefore $z$ and $\boldsymbol{Y}_2$ are uncorrelated and, since they are jointly normal, they are independent. Now, clearly $\mathbb{E}(z) = m_1 + Am_2$, therefore it follows that

$$\begin{aligned}\mathbb{E}(\boldsymbol{Y}_1|\boldsymbol{Y}_2) &= \mathbb{E}(z - A\boldsymbol{Y}_2|\boldsymbol{Y}_2) \\ &= \mathbb{E}(z|\boldsymbol{Y}_2) - \mathbb{E}(A\boldsymbol{Y}_2|\boldsymbol{Y}_2) = \mathbb{E}(z) - A\boldsymbol{Y}_2 \\ &= m_1 + A(m_2 - \boldsymbol{Y}_2) = m_1 + \mathsf{C}_{21}^T\mathsf{C}^{-1}{}_{22}(\boldsymbol{Y}_2 - m_2)\end{aligned}$$

which proves the mean $\mathbb{E}(\boldsymbol{Y}_1|\boldsymbol{Y}_2)$. For the covariance matrix, note that

$$\begin{aligned}\operatorname{var}(\boldsymbol{Y}_1|\boldsymbol{Y}_2) &= \operatorname{var}(z - A\boldsymbol{Y}_2|\boldsymbol{Y}_2) \\ &= \operatorname{var}(z|\boldsymbol{Y}_2) + \operatorname{var}(A\boldsymbol{Y}_2|\boldsymbol{Y}_2) - A\operatorname{cov}(z, \boldsymbol{Y}_2) - \operatorname{cov}(z, \boldsymbol{Y}_2)A^T \\ &= \operatorname{var}(z|\boldsymbol{Y}_2) \\ &= \operatorname{var}(z)\end{aligned}$$

So, it is enough to show:

$$\begin{aligned}\operatorname{var}(\boldsymbol{Y}_1|\boldsymbol{Y}_2) = \operatorname{var}(z) &= \operatorname{var}(\boldsymbol{Y}_1 + A\boldsymbol{Y}_2) \\ &= \operatorname{var}(\boldsymbol{Y}_1) + A\operatorname{var}(\boldsymbol{Y}_2)A^T + A\operatorname{cov}(\boldsymbol{Y}_1, \boldsymbol{Y}_2) + \operatorname{cov}(\boldsymbol{Y}_2, \boldsymbol{Y}_1)A^T \\ &= \mathsf{C}_{11} + \mathsf{C}_{21}^T\mathsf{C}_{22}^{-1}\mathsf{C}_{22}\mathsf{C}_{22}^{-1}\mathsf{C}_{21} - 2\mathsf{C}_{21}^T\mathsf{C}_{22}^{-1}\mathsf{C}_{21} \\ &= \mathsf{C}_{11} + \mathsf{C}_{21}^T\mathsf{C}_{22}^{-1}\mathsf{C}_{21} - 2\mathsf{C}_{21}^T\mathsf{C}_{22}^{-1}\mathsf{C}_{21} \\ &= \mathsf{C}_{11} - \mathsf{C}_{21}^T\mathsf{C}_{22}^{-1}\mathsf{C}_{21}\end{aligned}$$

which proves the variance.

## 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. DLMF: Nist digital library of mathematical functions. http://dlmf.nist.gov/ **Part 2**, Release 1.0.5 of 2012–10–01 (2012)
5. Fasshauer, G.E.: Meshfree Approximation Methods with Matlab. World Scientific Publishing Co (2007)
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.: Error analysis for quasi-monte carlo methods. arXiv:1702.01487 [math.NA] (2017)
8. 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
9. 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]
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. Ritter, K.: Average-case analysis of numerical problems. Lecture Notes in Mathematics, Springer-Verlag, Berlin **vol. 1733**, 163–175 (2000)
15. Sabrina Dammertz, A.K.: Image synthesis by rank-1 lattices. Monte Carlo and Quasi-Monte Carlo Methods **Part 2**, 217–236 (2006)