FAST AUTOMATIC BAYESIAN CUBATURE USING MATCHING KERNELS AND DESIGN

BY

R. JAGADEESWARAN

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Applied Mathematics
in the Graduate College of the
Illinois Institute of Technology

Approved _____
Advisor

Chicago, Illinois
Received: date / Accepted: date

ACKNOWLEDGMENT

I want to thank my advisor Prof. Fred Hickernell for his support and guidance in my completion of this thesis and throughout my studies here at IIT. His support and motivation have given me the confidence to endure through the research.

I would like to also thank the GAIL project collaborators with whom I have worked to add my new algorithms to the GAIL Matlab toolbox: Professor Sou-Cheng Choi, Yuhan Ding, Lan Jiang, Xin Tong, and Kan Zhang. Especially, Professor Sou-Cheng Chois support and guidance as the project leader helped me to focus of my cubature algorithms.

My special gratitude also goes to my thesis committee members, Professor Jinqiao Duan, Professor Fred J. Hickernell, Professor Shuwang Li, and Professor Geoffrey Williamson. Above all, I want to thank them because they were flexible and willing dedicate time to review my work and attend my comprehensive and defense examinations.

I would like to thank Dirk Nuyens for suggestions, valuable tips and notes when we were researching higher order nets and kernels.

I would like to thank the organizers of the SAMSI-Lloyds-Turing Workshop on Probabilistic Numerical Methods, where a part of preliminary version of this work was discussed. I also thank Chris Oates and Sou-Cheng Choi for valuable comments.

I would like to specifically thank my friend Samuel Davidson for reviewing and suggesting the improvements on the text.

Last but not least, I would not be able to make it without the support of my family. I would like to thank my wife for her continuous support and sacrifice. I also would like to thank my parents for their endless support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Automatic cubatures approximate multidimensional integrals to user-specified error tolerances. In many real world integration problems, the analytical solution is either unavailable or difficult to compute. To overcome this, one can use numerical algorithms that approximately estimates the value of the integral.

For high dimensional integrals, usage of quasi-Monte Carlo (QMC) methods are very popular. QMC methods JR: add ref are equal-weight quadrature rules where the quadrature points are chosen deterministically unlike Monte Carlo (MC) methods where the points are chosen randomly. The families of "lattice rules" and "digital nets" are the most popular quadrature points used.

These methods consider the integrand a deterministic function. There is an alternate approach called Bayesian methods. Bayesian cubature methods postulates that the integrand as an instance of a stochastic process. For high dimensional problems, it makes sense to fix the sampling density but determine the sample size, n, automatically. Here we assume a Gaussian process parameterized by a constant mean and a covariance function defined by a scale parameter and a function specifying how the integrand values at two different points in the domain are related. These parameters are estimated from integrand values or are given non-informative priors. This leads to credible interval for the integral. The sample size, n, is chosen to make the credible interval for the Bayesian posterior error no greater than the error tolerance.

However, the process just outlined typically requires vector-matrix operations with a computational cost of $O(n^3)$. Our innovation is to pair low discrepancy nodes with matching kernels that lower the computational cost to $O(n \log n)$. This approach is demonstrated using two methods: 1) rank-1 lattice sequences and shift-invariant kernels, 2) Sobol sequences and walsh kernel. It is also implemented in the Guaranteed Automatic Integration Library (GAIL).

CHAPTER 1

INTRODUCTION

1.1 Cubature

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

The integral may often be expressed as

$$\mu := \mu(f) := \mathbb{E}[f(\boldsymbol{X})] = \int_{[0,1]^d} f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}, \tag{1.1}$$

where $f : [0,1]^d \to \mathbb{R}$ is the integrand, and $\boldsymbol{X} \sim \mathcal{U}[0,1]^d$. The process of transforming the original integral into the form of (1.1) is not addressed here. The cubature may be an affine function of integrand values:

$$\widehat{\mu} := \widehat{\mu}(f) := w_0 + \sum_{i=1}^{n} f(\boldsymbol{x}_i) w_i, \tag{1.2}$$

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

1.2 Stopping Criterion

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

$$|\mu - \widehat{\mu}| \le \varepsilon. \tag{1.3}$$

Rather than relying on strong assumptions about the integrand, such as an upper bound on its variance or total variation, we construct a stopping criterion that is based on a credible interval arising from a Bayesian approach to the problem. We build upon the work of Briol et al. [2], Diaconis [6], O'Hagan [26], Ritter [30], Rasmussen and Ghahramani [28], and others. Our algorithm is an example of *probabilistic numerics*.

JR: Briefly explain Probabilistic numeric ..

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

1.3 Low discrepancy points

Low discrepancy points are characterized by how uniformly and randomly the points are distributed which is measured by the *discrepancy* score. Especially when the points are projected onto low-dimensions. The discrepancy measure is defined as below. Let $\mathcal{M}$ be the set of all intervals of the form $\prod_{j=1}^{d}[a_j, b_j) = \{\boldsymbol{x} \in \mathbb{R}^d : a_j \leq x_j \leq b_j, 0 \leq a_j \leq b_j \leq 1$, $|\mathcal{P}|$ the cardinality of the set $\mathcal{P}$, and λ the Lebesgue measure. Then, the discrepancy of a point set $\mathcal{P}$ is,

$$D(\mathcal{P}) := \sup_{M\in\mathcal{M}} \left| \frac{|M \cap \mathcal{P}|}{|\mathcal{P}|} - \lambda(M) \right|$$

The *low discrepancy points* satisfy $D(\mathcal{P}) = \mathcal{O}(\log(n)^d/n)$. In this work we experiment with two most popular low discrepancy point sets, 1) Lattice points, 2) Sobol points.

1.4 Prior Work

Hickernell [13] compares different approaches to cubature error analysis depending on whether the rule is deterministic or random and whether the integrand is assumed to be deterministic or random. Error analyis that assumes a deterministic integrand lying in a Banach space leads to an error bound that is typically impractical for deciding how large n must be to satisfy (1.3). 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 [14, 19] have developed stopping criteria for cubature rules based on low discrepancy nodes by tracking the decay of the discrete Fourier coefficients of the integrand. The algorithm proposed here also relies on discrete Fourier coefficients, but in a different way. Although we only explore automatic Bayesian cubature for absolute error tolerances, the recent work by Hickernell, Jiménez-Rugama, and Li [15] suggest how one might accommodate more general error criteria, such as relative error tolerances.

Chapter 2 explains the Bayesian approach to estimate the posterior cubature error and defines our automatic Bayesian cubature. Although much of this material is known, it is included for completeness. We end Chapter 2 by demonstrating why Bayesian cubature is typically computationally expensive. Chapter 3 introduces the concept of covariance kernels that match the nodes and expedite the computations required by our automatic Bayesian cubature. Chapter 4 implements this concept for shift invariant kernels and rank-1 lattice nodes. It also describes how to avoid cancellation error for kernels of product form. Chapter 5 demonstrates another implementation of matching nodes and kernel using sobol points and walsh kernel. Numerical examples are provided in Chapter 6 to demonstrate our new algorithm. We conclude with a brief discussion.

CHAPTER 2

BAYESIAN CUBATURE

JR: Briefly explain Bayesian approach using the words of diaconis, poincare work ..

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 constant mean m and covariance function $s^2 C_{\boldsymbol{\theta}}$, where s is a non-negative scale factor, and $C_{\boldsymbol{\theta}} : [0,1]^d \times [0,1]^d \to \mathbb{R}$ is a symmetric, positive-definite function and parameterized by $\boldsymbol{\theta}$:

$$\mathsf{C}^T = \mathsf{C}, \quad \boldsymbol{a}^T \mathsf{C} \boldsymbol{a} > 0, \quad \text{where } \mathsf{C} = (C_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j))_{i,j=1}^n, \quad \text{for all } \boldsymbol{a} \neq 0,\ n \in \mathbb{N},\ \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in [0,1]^d. \tag{2.1}$$

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

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

$$\boldsymbol{f} \sim \mathcal{N}(m\mathbf{1}, s^2 \mathsf{C}), \tag{2.2a}$$

$$\mu \sim \mathcal{N}(m, s^2 c_0), \tag{2.2b}$$

$$\text{where } c_0 = \int_{[0,1]^d \times [0,1]^d} C_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{t}) \, \mathrm{d}\boldsymbol{x} \, \mathrm{d}\boldsymbol{t}, \tag{2.2c}$$

$$\operatorname{cov}(\boldsymbol{f}, \mu) = \left(\int_{[0,1]^d} C(\boldsymbol{t}, \boldsymbol{x}_i) \, \mathrm{d}\boldsymbol{t} \right)_{i=1}^n =: \boldsymbol{c}. \tag{2.2d}$$

We need the following lemma to derive the posterior error of our cubature.

Lemma 2.1.1 *[29, (A.6), (A.11–13)] If* $\boldsymbol{Y} = (\boldsymbol{Y}_1, \boldsymbol{Y}_2)^T \sim \mathcal{N}(\boldsymbol{m}, \mathsf{C})$, *where* $\boldsymbol{Y}_1$ *and*

$\boldsymbol{Y}_2$ are random vectors of arbitrary length, and

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

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

then

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

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

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

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

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

It follows from Lemma 2.1.1 that the *conditional* distribution of the integral given observed function values, $\boldsymbol{f} = \boldsymbol{y}$ is also Gaussian:

$$\mu | (\boldsymbol{f} = \boldsymbol{y}) \sim \mathcal{N}\big(m(1 - \boldsymbol{c}^T \mathsf{C}^{-1}\mathbf{1}) + \boldsymbol{c}^T \mathsf{C}^{-1}\boldsymbol{y}, \ \ s^2(c_0 - \boldsymbol{c}^T \mathsf{C}^{-1}\boldsymbol{c})\big). \tag{2.3}$$

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

$$\widehat{\mu} | (\boldsymbol{f} = \boldsymbol{y}) = m(1 - \mathbf{1}^T \mathsf{C}^{-1}\boldsymbol{c}) + \boldsymbol{c}^T \mathsf{C}^{-1}\boldsymbol{y}, \tag{2.4}$$

which takes the form of (1.2). Under this definition, the cubature error has zero mean and a variance depending on the choice of nodes:

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

A credible interval for the integral is given by

$$\mathbb{P}_f\left[|\mu - \widehat{\mu}| \;\le\; 2.58 s\sqrt{c_0 - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c}}\right] = 99\%. \tag{2.5}$$

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

2.2 Parameter estimation The credible interval in (2.5) suggests how our automatic Bayesian cubature proceeds. Integrand data is accumulated until the width of the credible interval 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. The first and a very straight forward approach is to estimate the parameters is via maximum likelihood estimation (MLE). The log-likelihood function of the parameters given the function data $\boldsymbol{y}$ is:

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

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

$$m_{\text{MLE}} = \frac{\mathbf{1}^T\mathsf{C}^{-1}\boldsymbol{y}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}, \tag{2.6}$$

$$\begin{aligned} s^2_{\text{MLE}} &= \frac{1}{n}(\boldsymbol{y} - m_{\text{MLE}}\mathbf{1})^T\mathsf{C}^{-1}(\boldsymbol{y} - m_{\text{MLE}}\mathbf{1}) \\ &= \frac{1}{n}\boldsymbol{y}^T\left[\mathsf{C}^{-1} - \frac{\mathsf{C}^{-1}\mathbf{1}\mathbf{1}^T\mathsf{C}^{-1}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}\right]\boldsymbol{y}, \end{aligned} \tag{2.7}$$

$$\boldsymbol{\theta}_{\text{MLE}} = \underset{\boldsymbol{\theta}}{\text{argmin}}\left\{\log\left(\boldsymbol{y}^T\left[\mathsf{C}^{-1} - \frac{\mathsf{C}^{-1}\mathbf{1}\mathbf{1}^T\mathsf{C}^{-1}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}\right]\boldsymbol{y}\right) + \frac{1}{n}\log(\det(\mathsf{C}))\right\}. \tag{2.8}$$

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

Under these estimates of the parameters, the cubature (2.4) and the credible

interval (2.5) simplify to

$$\widehat{\mu}_{\text{MLE}} = \left(\frac{(1 - \mathbf{1}^T \mathsf{C}^{-1} \boldsymbol{c})\mathbf{1}}{\mathbf{1}^T \mathsf{C}^{-1} \mathbf{1}} + \boldsymbol{c} \right)^T \mathsf{C}^{-1} \boldsymbol{y}, \tag{2.9}$$

$$\text{err}^2_{\text{MLE}} := \frac{2.58^2}{n} \boldsymbol{y}^T \left[\mathsf{C}^{-1} - \frac{\mathsf{C}^{-1} \mathbf{1} \mathbf{1}^T \mathsf{C}^{-1}}{\mathbf{1}^T \mathsf{C}^{-1} \mathbf{1}} \right] \boldsymbol{y} (c_0 - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{c}), \tag{2.10}$$

$$\mathbb{P}_f \left[|\mu - \widehat{\mu}_{\text{MLE}}| \leq \text{err}_{\text{MLE}} \right] = 99\%. \tag{2.11}$$

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

2.2.1.1 Gradient descent to find optimal shape parameter. Searching for $\boldsymbol{\theta}_{\text{MLE}}$ as defined in (2.8) is not very efficient. Using the derivative of $l(s, m, \boldsymbol{\theta}|\boldsymbol{y})$, We can apply optimization techniques such as gradient descent to find the optimal value faster. Let us define the objective function for the same purpose by excluding the negative sign, which modifies the problem as minimization

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

Taking derivative with respect to $\boldsymbol{\theta}_i$

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\theta}_i} \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y}) &= \frac{1}{n} \frac{\partial}{\partial \boldsymbol{\theta}_i} \log(\det\, \mathsf{C}) + \frac{\partial}{\partial \boldsymbol{\theta}_i} \log \left((\boldsymbol{y} - m_{\text{MLE}} \mathbf{1})^T \mathsf{C}^{-1} (\boldsymbol{y} - m_{\text{MLE}} \mathbf{1}) \right) \\ &= \frac{1}{n} \text{trace} \left(\mathsf{C}^{-1} \frac{\partial \mathsf{C}}{\partial \boldsymbol{\theta}_i} \right) - \frac{\left((\boldsymbol{y} - m_{\text{MLE}} \mathbf{1})^T \mathsf{C}^{-1} \right)^T \left(\frac{\partial \mathsf{C}}{\partial \boldsymbol{\theta}_i} \right) \left((\boldsymbol{y} - m_{\text{MLE}} \mathbf{1})^T \mathsf{C}^{-1} \right)}{(\boldsymbol{y} - m_{\text{MLE}} \mathbf{1})^T \mathsf{C}^{-1} (\boldsymbol{y} - m_{\text{MLE}} \mathbf{1})} \end{aligned}$$

where we used some of the results from [9]. This can be used further for steepest gradient descent as follows,

$$\boldsymbol{\theta}_i^{(j+1)} = \boldsymbol{\theta}_i^{(j+1)} - \nu \frac{\partial}{\partial \boldsymbol{\theta}_i} \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y}) \tag{2.12}$$

where ν is the stepping constant.

2.2.2 Full Bayes. Rather than use maximum likelihood to determine m and s one can treat them as hyper-parameters with 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

$$
\begin{aligned}
&\rho_\mu(z|\boldsymbol{f}=\boldsymbol{y}) \\
&\quad\propto \int_0^\infty \int_{-\infty}^\infty \rho_\mu(z|\boldsymbol{f}=\boldsymbol{y}, m=\xi, s^2=\lambda)\rho_{\boldsymbol{f}}(\boldsymbol{y}|\xi,\lambda)\rho_{m,s^2}(\xi,\lambda)\,\mathrm{d}\xi\mathrm{d}\lambda \\
&\qquad\qquad \text{by the properties of conditional probability} \\
&\quad\propto \int_0^\infty \int_{-\infty}^\infty \rho_\mu(z|\boldsymbol{f}=\boldsymbol{y}, m=\xi, s^2=\lambda)\rho_{\boldsymbol{f}}(\boldsymbol{y}|\xi,\lambda)\rho_{m,s^2}(\xi,\lambda)\,\mathrm{d}\xi\mathrm{d}\lambda \\
&\qquad\qquad \text{by Bayes' Theorem} \\
&\quad\propto \int_0^\infty \frac{1}{\lambda^{(n+3)/2}} \int_{-\infty}^\infty \exp\Bigg(-\frac{1}{2\lambda}\Bigg\{\frac{[z-\xi(1-\boldsymbol{c}^T\mathsf{C}^{-1}\mathbf{1})-\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{y}]^2}{c_0-\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c}} \\
&\qquad\qquad + (\boldsymbol{y}-\xi\mathbf{1})^T\mathsf{C}^{-1}(\boldsymbol{y}-\xi\mathbf{1})\Bigg\}\Bigg)\,\mathrm{d}\xi\mathrm{d}\lambda \\
&\qquad\qquad \text{by (2.2), (2.3) and } \rho_{m,s^2}(\xi,\lambda)\propto 1/\lambda \\
&\quad\propto \int_0^\infty \frac{1}{\lambda^{(n+3)/2}} \int_{-\infty}^\infty \exp\left(-\frac{\alpha\xi^2-2\beta\xi+\gamma}{2\lambda(c_0-\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c})}\right)\,\mathrm{d}\xi\mathrm{d}\lambda,
\end{aligned}
$$

where

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

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

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

$$\propto \int_0^\infty \frac{1}{\lambda^{(n+2)/2}} \exp\left(-\frac{\gamma-\beta^2/\alpha}{2\lambda(c_0-\boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c})}\right) \mathrm{d}\lambda$$
$$\propto \left(\gamma-\frac{\beta^2}{\alpha}\right)^{-n/2} \propto \left(\alpha\gamma-\beta^2\right)^{-n/2}.$$

Finally, we simplify the key term:

$$\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 \\
& - 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}\boldsymbol{y}) \\
& + (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}) \\
& + [\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 \\
& - \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}\boldsymbol{y} \\
& (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 \\
& + \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} \\
\propto {} & (z-\widehat{\mu}_{\text{full}})^2+(n-1)\sigma^2_{\text{full}} \\
\propto {} & \left(1+\frac{1}{n-1}\frac{(z-\mu_{\text{full}})^2}{\widehat{\sigma}^2_{\text{full}}}\right)
\end{aligned}$$

i.e.,

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

where $\widehat{\mu}_{\text{full}}=\widehat{\mu}_{\text{MLE}}$ and

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

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, $\widehat{\mu}_{\text{full}} = \widehat{\mu}_{\text{MLE}}$, but the confidence interval is wider:

$$\mathbb{P}_f \left[|\mu - \widehat{\mu}_{\text{MLE}}| \leq \text{err}_{\text{full}} \right] = 99\%, \tag{2.13}$$

where

$$\text{err}_{\text{full}} := t_{n_j - 1, 0.995} \widehat{\sigma}_{\text{full}} > \text{err}_{\text{MLE}}. \tag{2.14}$$

Here $t_{n-1,0.995}$ denotes the 99.5 percentile of a standard Student's t-distribution with $n-1$ degrees of freedom. The stopping criterion for the full Bayes case is more conservative than that in the empirical Bayes case, (2.11).

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 (2.8) or the cross-validation approach described next.

2.2.3 Full Bayes with generic prior. Rather than use non-informative, conjugate prior one can use generic prior, namely $\boldsymbol{\rho}_{m,s^2}(\xi, \lambda) \propto g(\lambda)$, than can generalize to any generic function. Then the posterior density for the integral given the data using Bayes theorem is

$$\begin{aligned}
&\rho_\mu(z|\boldsymbol{f} = \boldsymbol{y}) \\
&\quad \propto \int_0^\infty \int_{-\infty}^\infty \rho_\mu(z|\boldsymbol{f} = \boldsymbol{y}, m = \xi, s^2 = \lambda) \rho_{\boldsymbol{f}}(\boldsymbol{y}|\xi, \lambda) \rho_{m,s^2}(\xi, \lambda) \, \mathrm{d}\xi \mathrm{d}\lambda \\
&\qquad\qquad \text{by the properties of conditional probability} \\
&\quad \propto \int_0^\infty \int_{-\infty}^\infty \rho_\mu(z|\boldsymbol{f} = \boldsymbol{y}, m = \xi, s^2 = \lambda) \rho_{\boldsymbol{f}}(\boldsymbol{y}|\xi, \lambda) \rho_{m,s^2}(\xi, \lambda) \, \mathrm{d}\xi \mathrm{d}\lambda \\
&\qquad\qquad \text{by Bayes' Theorem} \\
&\quad \propto \int_0^\infty \frac{g(\lambda)}{\lambda^{(n+1)/2}} \int_{-\infty}^\infty \exp\Bigg(-\frac{1}{2\lambda} \Bigg\{ \frac{[z - \xi(1 - \boldsymbol{c}^T \mathsf{C}^{-1} \mathbf{1}) - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{y}]^2}{c_0 - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{c}} \\
&\qquad\qquad + (\boldsymbol{y} - \xi \mathbf{1})^T \mathsf{C}^{-1} (\boldsymbol{y} - \xi \mathbf{1}) \Bigg\} \Bigg) \, \mathrm{d}\xi \mathrm{d}\lambda
\end{aligned}$$

$$\text{by (2.2), (2.3) and } \rho_{m,s^2}(\xi,\lambda) \propto g(\lambda)$$
$$\propto \int_0^\infty \frac{g(\lambda)}{\lambda^{(n+1)/2}} \int_{-\infty}^\infty \exp\left(-\frac{\alpha\xi^2 - 2\beta\xi + \gamma}{2\lambda(c_0 - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c})}\right) \mathrm{d}\xi\mathrm{d}\lambda,$$

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. But, factors that depend on ξ, λ, or z must be kept. Completing the square $\alpha\xi^2 - 2\beta\xi + \gamma = \alpha(\xi - \beta/\alpha)^2 - (\beta^2/\alpha) + \gamma$, allows us to evaluate the integrals with respect to ξ and λ:

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

This can be interpreted as Laplace transform of $g(\lambda)$.

$$\begin{aligned}
\rho_\mu(z|\boldsymbol{f} = \boldsymbol{y}) &\propto \int_0^\infty \frac{g(\lambda)}{\lambda^{n/2}} \exp\left(-\frac{\gamma - \beta^2/\alpha}{2\lambda(c_0 - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c})}\right) \mathrm{d}\lambda \\
&\propto \int_0^\infty \frac{g(\lambda)}{\lambda^{n/2}} \exp\left(-\frac{1}{\lambda}\eta\right) \mathrm{d}\lambda, \quad \text{where} \quad \eta = \frac{\gamma - \beta^2/\alpha}{2(c_0 - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c})}
\end{aligned}$$

Let $\lambda = \dfrac{1}{w}$, $\quad \mathrm{d}\lambda = -w^{-2}\mathrm{d}w$ then,

$$\begin{aligned}
\rho_\mu(z|\boldsymbol{f} = \boldsymbol{y}) &\propto \int_0^\infty \frac{g(\lambda)}{\lambda^{n/2}} \exp\left(-\frac{1}{\lambda}\eta\right) \mathrm{d}\lambda \\
&= \int_0^\infty \frac{g(1/w)}{w^{-n/2}} \exp\left(-w\eta\right) (-w^{-2})\mathrm{d}w \\
&= \int_\infty^0 -g(1/w) w^{\frac{n}{2}-2} \exp\left(-w\eta\right) \mathrm{d}w
\end{aligned}$$

$$= \int_0^\infty g(1/w) w^{\frac{n-4}{2}} \exp(-w\eta)\, \mathrm{d}w$$
$$= \mathcal{LT}\{g(1/\eta)\}^{(\frac{n-4}{2})'}$$

where $\mathcal{LT}(.)$ denotes the Laplace transform and $(\frac{n-4}{2})'$ indicates the derivative taken after the transform. Here we used frequency domain derivative property of the Laplace transform. Thus, $\rho_\mu(z|\boldsymbol{f} = \boldsymbol{y})$ is proportional to $(\frac{n-4}{2})$th derivative of the Laplace transform of $g(1/\eta)$.

If $g(\lambda) = \dfrac{1}{\lambda}$, then

$$\begin{aligned}
\rho_\mu(z|\boldsymbol{f} = \boldsymbol{y}) &= \int_0^\infty g(1/w) w^{\frac{n}{2}-2} \exp(-w\eta)\, \mathrm{d}w \\
&= \left(\mathcal{LT}(g(1/t))\right)^{(\frac{n}{2}-2)'} |_{t=\eta} \\
&= \left(1/t^2\right)^{(\frac{n}{2}-2)'} |_{t=\eta} \\
&\propto \eta^{-n/2} = \left(\frac{\gamma - \beta^2/\alpha}{2(c_0 - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{c})}\right)^{-n/2} \\
&\propto \left(\gamma - \frac{\beta^2}{\alpha}\right)^{-n/2} \\
&\propto \left(\alpha\gamma - \beta^2\right)^{-n/2}.
\end{aligned}$$

Laplace transform of $g(1/t)$ is $1/t^2$, taking $\frac{n}{2} - 2$th derivative gives us

Finally, we simplify the key term η:

$$\begin{aligned}
\alpha\gamma - \beta^2 = {} & \mathbf{1}^T \mathsf{C}^{-1} \mathbf{1} (c_0 - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{c})(z - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{y})^2 \\
& - 2\mathbf{1}^T \mathsf{C}^{-1} \boldsymbol{y} (c_0 - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{c})(1 - \boldsymbol{c}^T \mathsf{C}^{-1} \mathbf{1})(z - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{y}) \\
& + (1 - \boldsymbol{c}^T \mathsf{C}^{-1} \mathbf{1})^2 \boldsymbol{y}^T \mathsf{C}^{-1} \boldsymbol{y} (c_0 - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{c}) \\
& + [\mathbf{1}^T \mathsf{C}^{-1} \mathbf{1} \boldsymbol{y}^T \mathsf{C}^{-1} \boldsymbol{y} - (\mathbf{1}^T \mathsf{C}^{-1} \boldsymbol{y})^2](c_0 - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{c})^2 \\
\propto {} & \mathbf{1}^T \mathsf{C}^{-1} \mathbf{1} \left(z - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{y} - \frac{(1 - \boldsymbol{c}^T \mathsf{C}^{-1} \mathbf{1}) \mathbf{1}^T \mathsf{C}^{-1} \boldsymbol{y}}{\mathbf{1}^T \mathsf{C}^{-1} \mathbf{1}}\right)^2 \\
& - \frac{[(1 - \boldsymbol{c}^T \mathsf{C}^{-1} \mathbf{1}) \mathbf{1}^T \mathsf{C}^{-1} \boldsymbol{y}]^2}{\mathbf{1}^T \mathsf{C}^{-1} \mathbf{1}} + (1 - \boldsymbol{c}^T \mathsf{C}^{-1} \mathbf{1})^2 \boldsymbol{y}^T \mathsf{C}^{-1} \boldsymbol{y}
\end{aligned}$$

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

i.e.,

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

where $\widehat{\mu}_{\text{full}} = \widehat{\mu}_{\text{MLE}}$ and

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

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, $\widehat{\mu}_{\text{full}} = \widehat{\mu}_{\text{MLE}}$, but the confidence interval is wider:

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

where

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

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

$$
\text{CV} = \sum_{i=1}^{n}(y_i - \widetilde{y}_i)^2. \tag{2.15}
$$

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

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

where the subscript i denotes the i^{th} row or column, and the subscript $-i$ denotes all rows or columns except the i^{th}. Following this notation, Lemma 2.1.1 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, (2.15) may be re-written as

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

The *generalized cross-validation* criterion (GCV) replaces the i^{th} diagonal element of A in the denominator by the average diagonal element of A [5, 11, 34]:

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

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

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

$$\boldsymbol{\theta}_{\text{GCV}} \;=\; \underset{\boldsymbol{\theta}}{\text{argmin}} \left\{ \log\left(\boldsymbol{y}^T \left[\mathsf{C}^{-2} - \frac{\mathsf{C}^{-2}\mathbf{1}\mathbf{1}^T\mathsf{C}^{-2}}{\mathbf{1}^T\mathsf{C}^{-2}\mathbf{1}} \right] \boldsymbol{y} \right) \;-\; 2\log\left(\text{trace}(\mathsf{C}^{-1})\right) \right\}. \quad (2.16)$$

Plugging this value of m into (2.4) yields

$$\widehat{\mu}_{\text{GCV}} = \left(\frac{(1 - \mathbf{1}^T\mathsf{C}^{-1}\boldsymbol{c})\mathsf{C}^{-1}\mathbf{1}}{\mathbf{1}^T\mathsf{C}^{-2}\mathbf{1}} + \boldsymbol{c} \right)^T \mathsf{C}^{-1}\boldsymbol{y}. \quad (2.17)$$

An estimate for s may be obtained by noting that by Lemma 2.1.1,

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

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

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

where

$$s^2_{\mathrm{GCV}} := \boldsymbol{y}^T\left[\mathsf{C}^{-2} - \frac{\mathsf{C}^{-2}\mathbf{1}\mathbf{1}^T\mathsf{C}^{-2}}{\mathbf{1}^T\mathsf{C}^{-2}\mathbf{1}}\right]\boldsymbol{y}\left[\mathrm{trace}(\mathsf{C}^{-1})\right]^{-1}.$$

The confidence interval based on generalized cross-validation corresponds to (2.5) with the GCV estimates for m, s, and $\boldsymbol{\theta}$:

$$\mathrm{err}_{\mathrm{GCV}} = 2.58 s_{\mathrm{GCV}}\sqrt{c_0 - \boldsymbol{c}^T\mathsf{C}^{-1}\boldsymbol{c}}, \tag{2.18}$$

$$\mathbb{P}_f\left[|\mu - \widehat{\mu}_{\mathrm{GCV}}| \le \mathrm{err}_{\mathrm{GCV}}\right] = 99\%. \tag{2.19}$$

Looking back over the results of Sections 2.2.1–2.2.4, it is noted that if the original covariance function, C, is replaced by bC 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 Cone of Functions and the credible interval

JR: Revise f_{nice} using kernel approximation

We assume the integrand belongs to a cone of functions to make the computations tractable. The concept of cone in general for cubature error analysis can be stated using the error bound definition. Given the data driven error bound $\mathrm{err}_{f,n}$

$$|\mu(f) - \widehat{\mu}_n(f)| \leq \mathrm{err}_{f,n}(f(x_1), \dots f(x_n)), \quad \forall f \in \mathcal{C}$$

where the cone of functions is defined as $\mathcal{C} = \{f : f \in \mathcal{C} \Rightarrow af \in \mathcal{C}\}$. More specifically, if a function f belongs to $\mathcal{C}$, then the obtained by constant multiplication af also belongs to the cone. This property is very useful, since

$$\begin{aligned} |\mu(af) - \widehat{\mu}_n(af)| &\leq |a|\,|\mu(f) - \widehat{\mu}_n(f)| \\ &\leq |a|\,\mathrm{err}_{f,n}(f(x_1), \dots f(x_n)) \\ &= \mathrm{err}_{f,n}(af(x_1), \dots af(x_n)) \end{aligned}$$

In the context of Bayesian cubature, we can explain the cone concept beginning with the definition of credible interval (2.5). Let $f \sim \mathcal{GP}$, is drawn from Gaussian process

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

This can be interpreted as $|\mu(f) - \widehat{\mu}_n(f)| \leq \mathrm{err}_n(f)$ with 99% confidence. If f is in the 99% of the functions such that $f(x_i) = y_i$ then af is also in the center of 99%.

We explain the credible interval using the following example. For this purpose, chosen a smooth and periodic integrand $f_{nice}(\boldsymbol{x}) = \exp(\sum_{j=1}^{d} \cos(2\pi x_j)) + a_{nice} f_{noise}$ and another integrand $f_{peaky}(\boldsymbol{x}) = f_{nice} + a_{peaky} f_{noise}$ where $a_{peaky} >> a_{nice}$. Where the function $f_{noise}(\boldsymbol{x}) = (1 - \exp(2\pi i\sqrt{-1}\boldsymbol{x}^T\boldsymbol{\zeta}))$ chosen to be zero at the sampling nodes $\{\boldsymbol{x}_i\}_{i=1}^n$, where $a \in \mathbb{R}$ is some constant, $\boldsymbol{\zeta} \in \mathbb{R}^d$ is some d-dimensional vector belonging to the dual space of the lattice nodes $\{\boldsymbol{x}_i\}_{i=1}^n$ chosen to sample the integrand. This help with to satisfy $f_{nice}(\boldsymbol{x}_i) = f_{real}(\boldsymbol{x}_i)$ at the sampling nodes $\{\boldsymbol{x}_i\}_{i=1}^n$

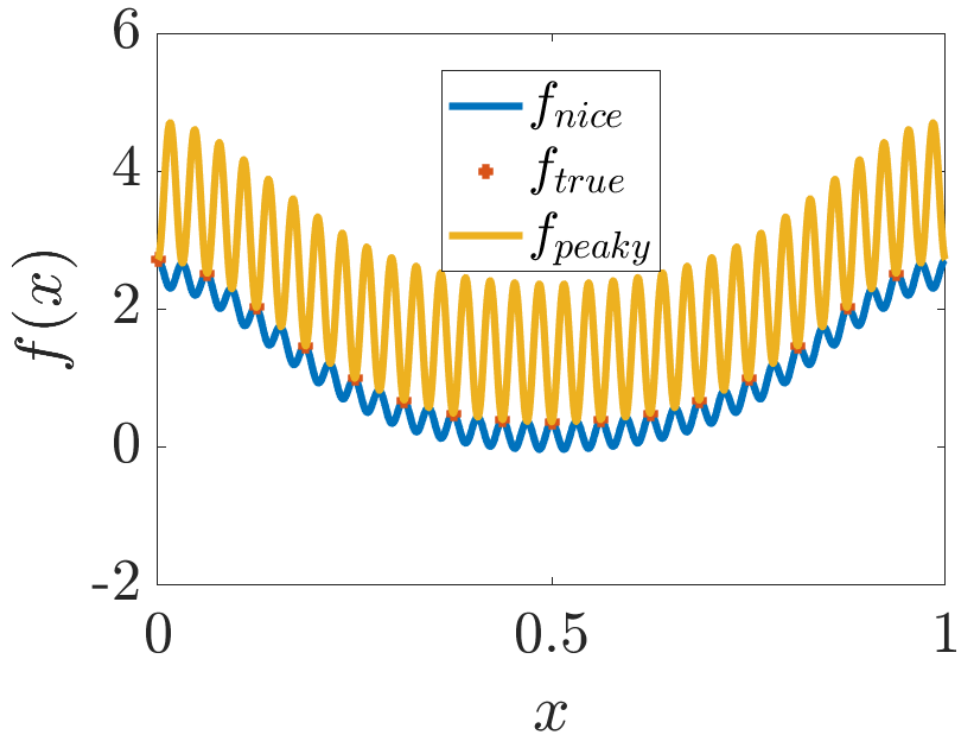

Figure 2.1. Example integrands 1) f_{true} true integrand, 2) f_{nice} a smooth function, and 3) f_{peaky} a peaky function, all have the same values at $\{\boldsymbol{x}_i\}_{i=1}^n$.

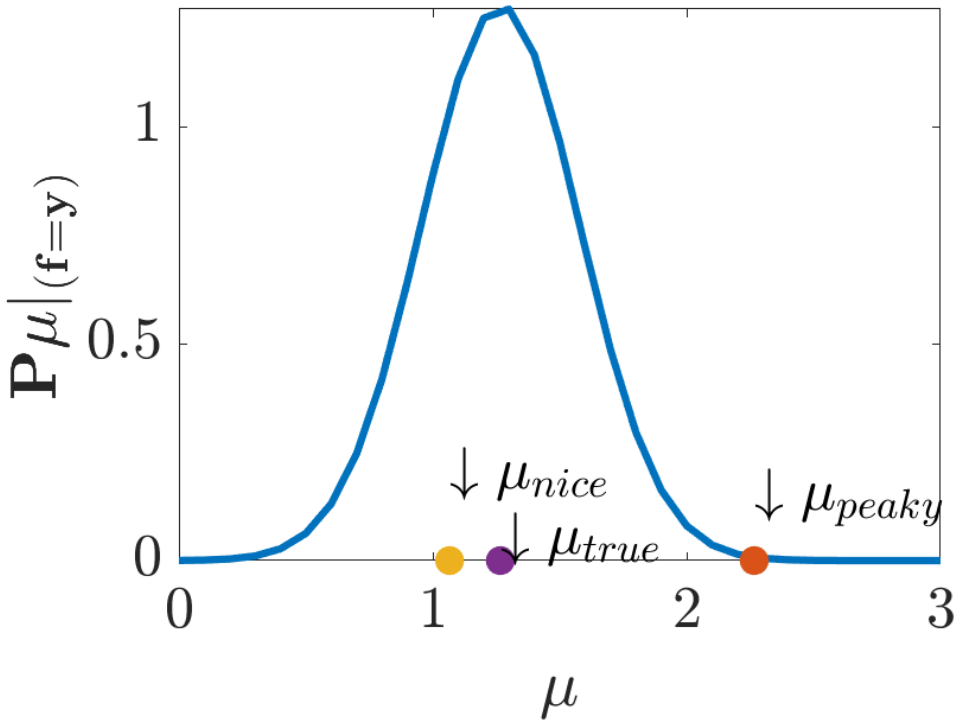

Figure 2.2. Probablity distributions showing the relative position integral of a smooth and a peaky function. f_{nice} lies within the center 99% of the confidence interval, and f_{peaky} lies on the outside of 99% of the confidence interval.JR: f_{nice} using kernel approximation

As shown in Figure 2.1, sampled function values $\{f(\boldsymbol{x}_i)\}_{i=1}^n$ from a smooth integrand f_{true} are shows as dots. One can imagine these samples $\{f(\boldsymbol{x}_i)\}_{i=1}^n$ were obtained from f_{nice}, a moderately smoother function or from f_{peaky} a highly oscillating function.

When using $n = 16$ rank-1 lattice points and $r = 1$ shift-invariant kernel, we get the posterior distribution of μ as shown in Figure 2.2. The true integral value is shown as μ_{true} which is at the center of the plot. The integral of the peaky function f_{peaky} almost lies outside of the 99% of the credible interval given by (2.11), whereas the μ_{nice} falls within.

2.4 The automatic Bayesian cubature algorithm

The previous section presents three credible intervals, (2.11), (2.13), and (2.19), 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 successively doubling the sample size. The steps are detailed in Algorithm 1.

2.5 Example with the Matérn kernel

To demonstrate automatic Bayesian cubature consider a Matérn covariance kernel:

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

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; the maximum sample size $n_{\max}$

1: $n \leftarrow n_0,\ n' \leftarrow 0,\ \text{err} \leftarrow \infty$

2: **while** err $> \varepsilon$ and $n \le n_{\max}$ **do**

3: Generate $\{\boldsymbol{x}_i\}_{i=n'+1}^{n}$ and sample $\{f(\boldsymbol{x}_i)\}_{i=n'+1}^{n}$

4: Compute $\boldsymbol{\theta}$ by (2.8) or (2.16)

5: Compute err according to (2.10), (2.14), or (2.18)

6: $n' \leftarrow n,\ n \leftarrow 2n'$

7: **end while**

8: Sample size to compute $\widehat{\mu}$, $n \leftarrow n'$

9: Compute $\widehat{\mu}$, the approximate integral, according to (2.9) or (2.17)

10: **return** $\widehat{\mu}$, n and err

Also, consider the integration problem of evaluating *multivariate normal probabilities*:

$$\mu = \int_{(\boldsymbol{a},\boldsymbol{b})} \frac{\exp\left(-\frac{1}{2}\boldsymbol{t}^T\Sigma^{-1}\boldsymbol{t}\right)}{\sqrt{(2\pi)^d \det(\Sigma)}}\,\mathrm{d}\boldsymbol{t}, \tag{2.20}$$

where $(\boldsymbol{a}, \boldsymbol{b})$ is a finite, semi-infinite or infinite box in $\mathbb{R}^d$. This integral does not have an analytic expression for general Σ, so cubatures are required.

Genz [10] introduced a variable transformation to transform (2.20) into an integral on the unit cube. Let $\Sigma = \mathsf{L}\mathsf{L}^T$ be the Cholesky decomposition where $\mathsf{L} = (l_{jk})_{j,k=1}^{d}$ is a lower triangular matrix. Iteratively define

$$\begin{aligned}
\alpha_1 &= \Phi(a_1), \qquad \beta_1 = \Phi(b_1),\\
\alpha_j(x_1, ..., x_{j-1}) &= \Phi\left(\frac{1}{l_{jj}}\left(a_j - \sum_{k=1}^{j-1} l_{jk}\Phi^{-1}(\alpha_k + x_k(\beta_k - \alpha_k))\right)\right), \quad j = 2, ..., d,\\
\beta_j(x_1, ..., x_{j-1}) &= \Phi\left(\frac{1}{l_{jj}}\left(b_j - \sum_{k=1}^{j-1} l_{jk}\Phi^{-1}(\alpha_k + x_k(\beta_k - \alpha_k))\right)\right), \quad j = 2, ..., d,
\end{aligned}$$

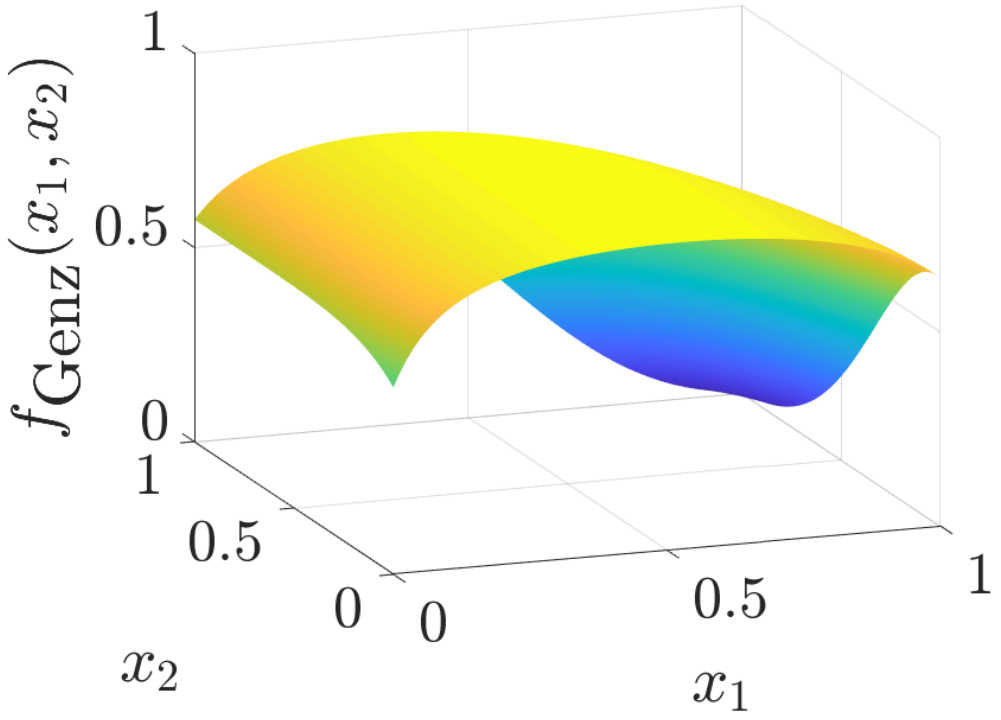

Figure 2.3. The $d = 3$ multivarivate normal probability transformed to an integral of f_{Genz} with $d = 2$.

$$f_{\text{Genz}}(\boldsymbol{x}) = \prod_{j=1}^{d} [\beta_j(\boldsymbol{x}) - \alpha_j(\boldsymbol{x})]. \tag{2.21}$$

where Φ is the cumulative standard normal distribution function. Then, $\mu = \int_{[0,1]^{d-1}} f_{\text{Genz}}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}$.

We use the following parameter values in the simulation:

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

On our test computer, it took more than an hour to compute $\widehat{\mu}_n$ with $n = 2^{14}$. As shown in Figure 2.4, computation time increases rapidly with n. Especially, maximum likelihood estimation of $\boldsymbol{\theta}$, which needs the objective function, is the most time

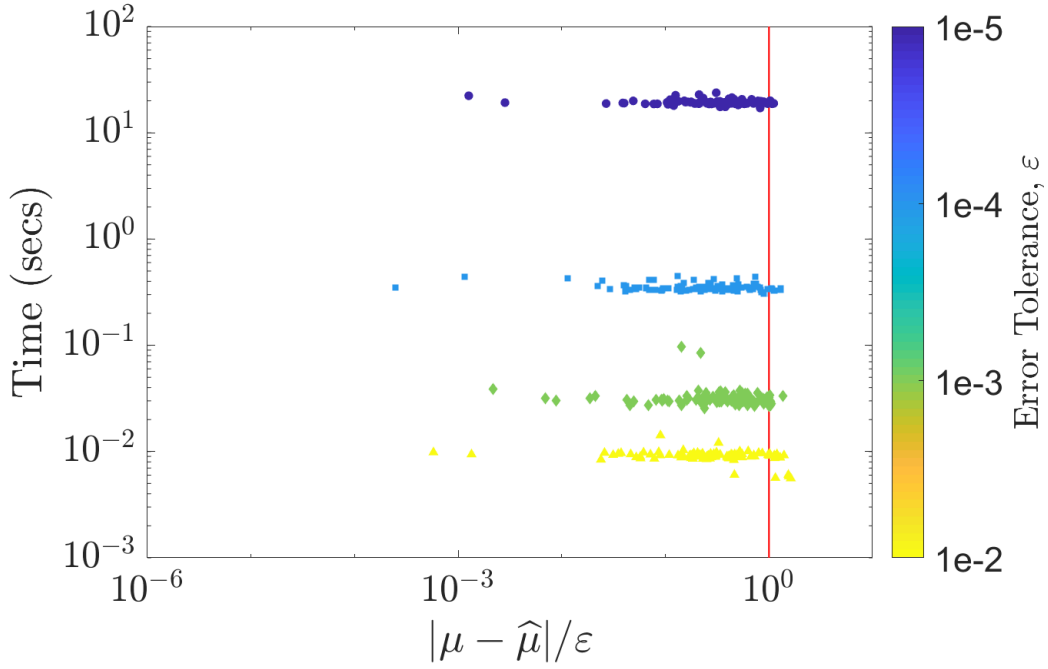

Figure 2.4. Multivariate Normal probability: Guaranteed integration using Matérn kernel in $d = 2$ using empirical stopping criterion within error tolerance ε.

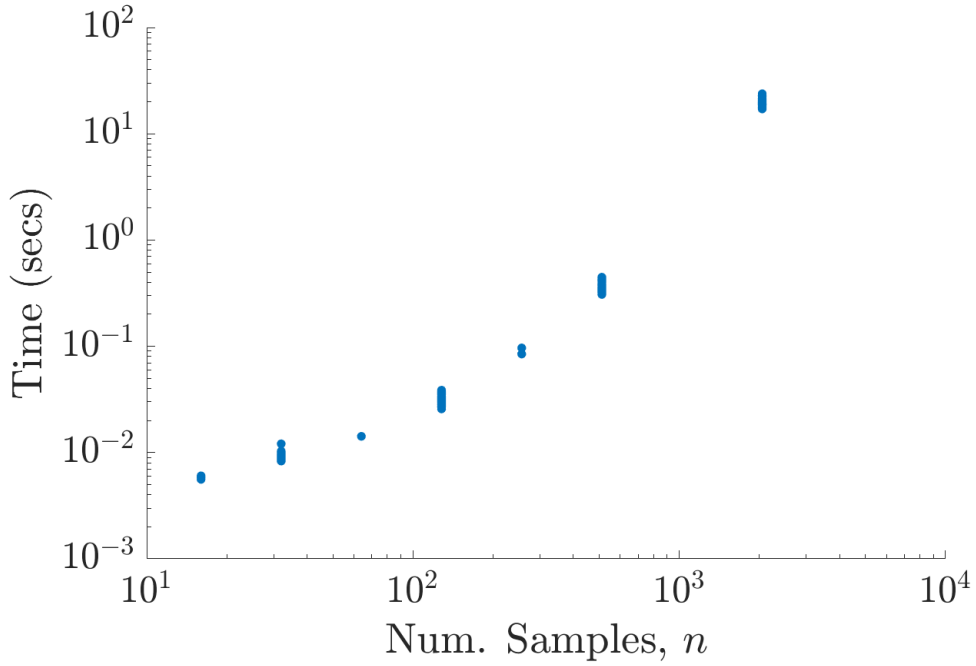

Figure 2.5. Multivariate Normal probability estimated using Matérn kernel in $d = 2$ using empirical stopping criterion. Computation time rapidly increases with increase of n

consuming of all. Because the objective function needs to be computed multiple times in every iteration to find its minimum. Not only the computational cost increases, also the C becomes highly ill-conditioned with increasing n. So, our algorithm in the current form is not straightaway usable for any practical applications.

CHAPTER 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. This section explains how to speed up the calculations. 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 introduce the notation

$$\begin{aligned} \mathsf{C} &= \Big(C_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j)\Big)_{i,j=1}^{n} = (\boldsymbol{C}_1, ..., \boldsymbol{C}_n) \\ &= \frac{1}{n}\mathsf{V}\Lambda\mathsf{V}^H, \qquad \mathsf{V}^H = n\mathsf{V}^{-1}, \\ \mathsf{V} &= (\boldsymbol{v}_1, ..., \boldsymbol{v}_n)^T = (\boldsymbol{V}_1, ..., \boldsymbol{V}_n) \\ \mathsf{C}^p &= \frac{1}{n}\mathsf{V}\Lambda^p\mathsf{V}^H, \qquad \forall p \in \mathbb{Z}, \end{aligned} \tag{3.1}$$

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. For any $n \times n$ vector $\boldsymbol{b}$, define the notation $\widetilde{\boldsymbol{b}} := \mathsf{V}^H\boldsymbol{b}$.

We make three assumptions that allow the fast computation:

$$\mathsf{V} \text{ may be identified analytically,} \tag{3.2a}$$

$$\boldsymbol{v}_1 = \boldsymbol{V}_1 = \boldsymbol{1}, \tag{3.2b}$$

$$\mathsf{V}^H\boldsymbol{b} \text{ requires only } \mathcal{O}(n \log(n)) \text{ operations } \forall \boldsymbol{b}. \tag{3.2c}$$

We call the transformation $\boldsymbol{b} \mapsto \mathsf{V}^H \boldsymbol{b}$ a *fast transform* and $C_{\boldsymbol{\theta}}$ a *fast transform kernel*.

Under assumptions (3.2) the eigenvalues may be identified as the fast transform of the first column of C:

$$\begin{aligned}\boldsymbol{\lambda} &= \begin{pmatrix}\lambda_1, & \ldots, & \lambda_n\end{pmatrix}^T = \Lambda \mathbf{1} = \Lambda \boldsymbol{v}_1^* = \underbrace{\left(\frac{1}{n}\mathsf{V}^H\mathsf{V}\right)}_{\mathsf{I}} \Lambda \boldsymbol{v}_1^* \\ &= \mathsf{V}^H \left(\frac{1}{n}\mathsf{V}\Lambda \boldsymbol{v}_1^*\right) = \mathsf{V}^H \boldsymbol{C}_1 = \widetilde{\boldsymbol{C}}_1. \end{aligned} \tag{3.3}$$

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

$$\widetilde{\mathbf{1}} = \mathsf{V}^H \mathbf{1} = \mathsf{V}^H \boldsymbol{V}_1 = \begin{pmatrix} n, & 0, & \ldots, & 0\end{pmatrix}^T.$$

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

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

In particular,

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

where $\widetilde{\boldsymbol{y}} = \mathsf{V}^H \boldsymbol{y}$ and $\widetilde{\boldsymbol{c}} = \mathsf{V}^H \boldsymbol{c}$. For any real $\boldsymbol{b}$, with $\widetilde{\boldsymbol{b}} = \mathsf{V}^H \boldsymbol{b}$, it follows that $\widetilde{b}_1$ is real since the first row of V^H is $\mathbf{1}$.

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

$$\int_{[0,1]^d} C(\boldsymbol{t}, \boldsymbol{x}) \, \mathrm{d}\boldsymbol{t} = 1 \qquad \forall \boldsymbol{x} \in [0,1]^d, \tag{3.4}$$

which implies that $c_0 = 1$ and $\boldsymbol{c} = \mathbf{1}$. Under (3.4), 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 (3.2), the empirical Bayes parameters in (2.6), (2.7), (2.8) (2.9), and (2.10) 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:

$$\begin{aligned}
m_{\text{MLE}} &= \frac{\widetilde{y}_1}{n} = \frac{1}{n}\sum_{i=1}^{n} y_i, \\
s^2_{\text{MLE}} &= \frac{1}{n^2}\sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i}, \\
\boldsymbol{\theta}_{\text{MLE}} &= \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\log\left(\sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i} \right) + \frac{1}{n}\sum_{i=1}^{n} \log(\lambda_i) \right], \\
\widehat{\mu}_{\text{MLE}} &= \frac{\widetilde{y}_1}{n} + \frac{1}{n}\sum_{i=2}^{n} \frac{\widetilde{c}_i^* \widetilde{y}_i}{\lambda_i}, \\
\text{err}_{\text{MLE}} &= \frac{2.58}{n} \sqrt{\sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i} \left(c_0 - \frac{1}{n}\sum_{i=1}^{n} \frac{|\widetilde{c}_i|^2}{\lambda_i} \right)},
\end{aligned} \tag{3.5}$$

Since all the quantities on the right hand sides can be obtained 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 (3.4) it follows that

$$\begin{aligned}
\widehat{\mu}_{\text{MLE}} &= \frac{\widetilde{y}_1}{n} = \frac{1}{n}\sum_{i=1}^{n} y_i, \\
\text{err}_{\text{MLE}} &= \frac{2.58}{n} \sqrt{\sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i} \left(1 - \frac{n}{\lambda_1} \right)}.
\end{aligned} \tag{3.6}$$

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

3.2.1 Gradient of the objective function using fast transform. If V does

not depend on $\boldsymbol{\theta}$ then one can fast compute the derivative of matrix C. Starting from the definition (3.1) and taking derivative w.r.t. θ_j,

$$\frac{\partial \mathsf{C}}{\partial \theta_j} = \frac{1}{n}\mathsf{V}\frac{\partial \mathsf{\Lambda}}{\partial \theta_j}\mathsf{V}^H = \frac{1}{n}\mathsf{V}\bar{\mathsf{\Lambda}}_{(j)}\mathsf{V}^H,$$

$$\text{where} \quad \bar{\mathsf{\Lambda}}_{(j)} = \operatorname{diag}(\bar{\boldsymbol{\lambda}}_{(j)}), \quad \text{and}$$

$$\bar{\boldsymbol{\lambda}}_{(j)} = \frac{\partial \boldsymbol{\lambda}}{\partial \theta_j} = \left(\frac{\partial \lambda_i}{\partial \theta_j}\right)_{i=1}^{n} = \left(\frac{\partial}{\partial \theta_j}\mathsf{V}^H \boldsymbol{C}_1\right) = \mathsf{V}^H \left(\frac{\partial}{\partial \theta_j} C_{\boldsymbol{\theta}}(\boldsymbol{x}_1, \boldsymbol{x}_i)\right)_{i=1}^{n}.$$

where we used the fast transform property (3.3). We use the notation $\bar{\boldsymbol{\lambda}}_{(j)} = \mathsf{V}^H \bar{\boldsymbol{C}}_{1(j)}$, where $\bar{\boldsymbol{C}}_{1(j)}$ denotes the first row of the gram matrix after taking derivative, i.e.

$$\bar{\boldsymbol{C}}_{1(j)} = \left(\frac{\partial}{\partial \theta_j} C_{\boldsymbol{\theta}}(\boldsymbol{x}_1, \boldsymbol{x}_i)\right)_{i=1}^{n}.$$

The goal is to find derivative of the objective function. First, let's rewrite the objective function from (3.5),

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y}) = \underbrace{\frac{1}{n}\log(\det\, \mathsf{C})}_{\mathcal{L}_{|\mathsf{C}|}} + \underbrace{\log\left((\boldsymbol{y} - m_{\text{MLE}}\mathbf{1})^T \mathsf{C}^{-1}(\boldsymbol{y} - m_{\text{MLE}}\mathbf{1})\right)}_{\mathcal{L}_{\boldsymbol{y}}}$$

$$=: \mathcal{L}_{|\mathsf{C}|} + \mathcal{L}_{\boldsymbol{y}}$$

Now, take the derivative

$$\frac{\partial}{\partial \theta_j}\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{\partial}{\partial \theta_j}\mathcal{L}_{|\mathsf{C}|} + \frac{\partial}{\partial \theta_j}\mathcal{L}_{\boldsymbol{y}}$$

Let's tackle the individual terms

$$\begin{aligned}
\frac{\partial}{\partial \theta_j}\mathcal{L}_{|\mathsf{C}|} &= \frac{\partial}{\partial \theta_j}\frac{1}{n}\log(\det\, \mathsf{C}) \\
&= \frac{1}{n}\operatorname{trace}\left(\mathsf{C}^{-1}\frac{\partial \mathsf{C}}{\partial \boldsymbol{\theta}_j}\right) = \frac{1}{n}\operatorname{trace}\left(\mathsf{V}\mathsf{\Lambda}^{-1}\mathsf{V}^H\frac{1}{n}\mathsf{V}\overline{\mathsf{\Lambda}}_{(j)}\mathsf{V}^H\right) \\
&= \frac{1}{n}\operatorname{trace}\left(\mathsf{V}\mathsf{\Lambda}^{-1}\overline{\mathsf{\Lambda}}_{(j)}\mathsf{V}^H\right), \quad \text{where we used } \mathsf{V}^H\mathsf{V} = n \\
&= \frac{1}{n}\operatorname{trace}\left(\mathsf{V}\operatorname{diag}\left(\frac{\overline{\lambda}_{i(j)}}{\lambda_i}\right)_{i=1}^{n}\mathsf{V}^H\right) = \frac{1}{n}\sum_{i=1}^{n}\frac{\overline{\lambda}_{i(j)}}{\lambda_i}
\end{aligned}$$

where we used the fact from [17]

$$\log(\det\, \mathsf{C}) = \operatorname{trace}(\log(\mathsf{C})).$$

Part of the $\mathcal{L}_{\boldsymbol{y}}$ was already simplified using the fast transform,

$$(\boldsymbol{y} - m_{\text{MLE}}\mathbf{1})^T \mathsf{C}^{-1} (\boldsymbol{y} - m_{\text{MLE}}\mathbf{1}) = \frac{1}{n} \sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i}.$$

Using the above result,

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} \mathcal{L}_{\boldsymbol{y}} &= \frac{\partial}{\partial \boldsymbol{\theta}_j} \log \left(\frac{1}{n} \sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i} \right) \\
&= \left(\frac{1}{n} \sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i} \right)^{-1} \frac{\partial}{\partial \boldsymbol{\theta}_j} \left(\frac{1}{n} \sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i} \right) \\
&= \left(\frac{1}{n} \sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i} \right)^{-1} \frac{1}{n} \sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i^2} \left(-\frac{\partial \lambda_i}{\partial \boldsymbol{\theta}_j} \right) \\
&= - \left(\sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i} \right)^{-1} \left(\sum_{i=2}^{n} |\widetilde{y}_i|^2 \frac{\bar{\lambda}_{i(j)}}{\lambda_i^2} \right).
\end{aligned}$$

Finally, using the above results

$$\frac{\partial}{\partial \theta_j} \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{1}{n} \sum_{i=1}^{n} \frac{\bar{\lambda}_{i(j)}}{\lambda_i} - \left(\sum_{i=2}^{n} \frac{|\tilde{\boldsymbol{y}}_i|^2 \bar{\lambda}_{i(j)}}{\lambda_i^2} \right) \left(\sum_{i=2}^{n} \frac{|\tilde{\boldsymbol{y}}_i|^2}{\lambda_j} \right)^{-1}.$$

If $m = 0$ assumption can be made

$$\frac{\partial}{\partial \theta_j} \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y}) = \frac{1}{n} \sum_{i=1}^{n} \frac{\bar{\lambda}_{i(j)}}{\lambda_i} - \left(\sum_{i=1}^{n} \frac{|\tilde{\boldsymbol{y}}_i|^2 \bar{\lambda}_{i(j)}}{\lambda_i^2} \right) \left(\sum_{i=1}^{n} \frac{|\tilde{\boldsymbol{y}}_i|^2}{\lambda_j} \right)^{-1}.$$

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 $\text{err}_{\text{full}} := t_{n_j - 1, 0.995} \widehat{\sigma}_{\text{full}}$, where $\widehat{\sigma}^2_{\text{full}}$ can also be computed swiftly under assumptions (3.2):

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

Under assumption (3.4) further simplification can be made:

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

It follows that

$$\mathrm{err}_{\mathrm{full}} = t_{n_j-1,0.995}\sqrt{\frac{1}{n(n-1)}\sum_{i=2}^{n}\frac{|\widetilde{y}_i|^2}{\lambda_i}\left(\frac{\lambda_1}{n}-1\right)}. \tag{3.7}$$

3.4 Generalized Cross-Validation

GCV yields a different cubature, which nevertheless can also be computed quickly using the fast transform. Under assumptions (3.2):

$$\begin{aligned}
m_{\mathrm{GCV}} &= m_{\mathrm{MLE}} = \frac{\widetilde{y}_1}{n} = \frac{1}{n}\sum_{i=1}^{n} y_i,\\
s^2_{\mathrm{GCV}} &:= \frac{1}{n}\sum_{i=2}^{n}\frac{|\widetilde{y}_i|^2}{\lambda_i^2}\left[\sum_{i=1}^{n}\frac{1}{\lambda_i}\right]^{-1},\\
\boldsymbol{\theta}_{\mathrm{GCV}} &= \operatorname*{argmin}_{\boldsymbol{\theta}}\left[\log\left(\sum_{i=2}^{n}\frac{|\widetilde{y}_i|^2}{\lambda_i^2}\right) - 2\log\left(\sum_{i=1}^{n}\frac{1}{\lambda_i}\right)\right],\\
\widehat{\mu}_{\mathrm{GCV}} &= \widehat{\mu}_{\mathrm{MLE}} = \frac{\widetilde{y}_1}{n} + \frac{1}{n}\sum_{i=2}^{n}\frac{\widetilde{c}_i^*\widetilde{y}_i}{\lambda_i},\\
\mathrm{err}_{\mathrm{GCV}} &= \frac{2.58}{n}\left\{\sum_{i=2}^{n}\frac{|\widetilde{y}_i|^2}{\lambda_i^2}\left[\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\lambda_i}\right]^{-1}\times\left(c_0 - \frac{1}{n}\sum_{i=1}^{n}\frac{|\widetilde{c}_i|^2}{\lambda_i}\right)\right\}^{1/2}.
\end{aligned} \tag{3.8}$$

Moreover, under further assumption (3.4) it follows that

$$\begin{aligned}
\widehat{\mu}_{\mathrm{GCV}} &= \widehat{\mu}_{\mathrm{MLE}} = \widehat{\mu}_{\mathrm{full}} = \frac{\widetilde{y}_1}{n} = \frac{1}{n}\sum_{i=1}^{n} y_i,\\
\mathrm{err}_{\mathrm{GCV}} &= \frac{2.58}{n}\left\{\sum_{i=2}^{n}\frac{|\widetilde{y}_i|^2}{\lambda_i^2}\left[\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\lambda_i}\right]^{-1}\times\left(1-\frac{n}{\lambda_1}\right)\right\}^{1/2}.
\end{aligned} \tag{3.9}$$

In this case too, $\widehat{\mu}$ is simply the sample mean.

3.4.1 Derivative of the loss function. Using the results obtained from the previous section, we can reduce the computational cost of the derivative of the loss function,

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y})_{\mathrm{GCV}} = \log\left(\sum_{i=2}^{n}\frac{|\widetilde{y}_i|^2}{\lambda_i^2}\right) - 2\log\left(\sum_{i=1}^{n}\frac{1}{\lambda_i}\right)$$

Using the similar techniques from Section 3.2.1

$$
\begin{aligned}
&\frac{\partial}{\partial\theta_j}\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y})_{\mathrm{GCV}} \\
&= \left(\sum_{i=2}^{n}\frac{|\widetilde{y}_i|^2}{\lambda_i^2}\right)^{-1}\frac{\partial}{\partial\theta_j}\left(\sum_{i=2}^{n}\frac{|\widetilde{y}_i|^2}{\lambda_i^2}\right) - 2\left(\sum_{i=1}^{n}\frac{1}{\lambda_i}\right)^{-1}\frac{\partial}{\partial\theta_j}\left(\sum_{i=1}^{n}\frac{1}{\lambda_i}\right) \\
&= \left(\sum_{i=2}^{n}\frac{|\widetilde{y}_i|^2}{\lambda_i^2}\right)^{-1}\left(\sum_{i=2}^{n}\frac{|\widetilde{y}_i|^2}{\lambda_i^3}(-2)\frac{\partial\lambda_i}{\partial\theta_j}\right) \\
&\qquad\qquad - 2\left(\sum_{i=1}^{n}\frac{1}{\lambda_i}\right)^{-1}\left(\sum_{i=1}^{n}\frac{1}{\lambda_i^2}(-1)\frac{\partial\lambda_i}{\partial\theta_j}\right) \\
&= -2\left(\sum_{i=2}^{n}\frac{|\widetilde{y}_i|^2}{\lambda_i^2}\right)^{-1}\left(\sum_{i=2}^{n}\frac{|\widetilde{y}_i|^2\,\bar{\lambda}_{i(j)}}{\lambda_i^3}\right) + 2\left(\sum_{i=1}^{n}\frac{1}{\lambda_i}\right)^{-1}\left(\sum_{i=1}^{n}\frac{\bar{\lambda}_{i(j)}}{\lambda_i^2}\right)
\end{aligned}
$$

3.5 Product kernel

In this research, We use product kernels in the demonstrations and numerical implementations. They got nice properties which are helpful to obtain analytical results easily. Product kernels are of the form in d dimensions

$$C_{\boldsymbol{\theta}}(\boldsymbol{t},\boldsymbol{x}) = \prod_{l=1}^{d}\Big[1-\eta\ \mathfrak{C}(x_l,t_l)\Big] \tag{3.10}$$

where η is called shape parameter and $\mathfrak{C}$ is some positive definite function. The derivative of the product kernel can be obtained easily.

3.5.1 Derivative of the product kernel. We discussed using gradient descent to find optimal shape parameter in Section 2.2.1.1 which requires the derivative of kernel w.r.t. η

$$
\begin{aligned}
\frac{\partial}{\partial\eta}C_{\boldsymbol{\theta}}(\boldsymbol{t},\boldsymbol{x}) &= \frac{\partial}{\partial\eta}\prod_{l=1}^{d}\Big[1-\eta\mathfrak{C}(x_l,t_l)\Big], \\
&= \sum_{j=1}^{d}\prod_{l=1,l\neq j}^{d}\Big[1-\eta\mathfrak{C}(x_l,t_l)\Big]\Big(-\mathfrak{C}(x_j,t_j)\Big) \\
&= \prod_{l=1}^{d}\Big[1-\eta\mathfrak{C}(x_l,t_l)\Big]\sum_{j=1}^{d}\frac{\Big(-\mathfrak{C}(x_j,t_j)\Big)}{1-\eta\mathfrak{C}(x_j,t_j)}
\end{aligned}
$$

$$
\begin{aligned}
&= C_{\boldsymbol{\theta}}(\boldsymbol{t},\boldsymbol{x})\frac{1}{\eta}\sum_{j=1}^{d}\frac{\Big(1-\eta\mathfrak{C}(x_j,t_j)-1\Big)}{1-\eta\mathfrak{C}(x_j,t_j)}\\
&= C_{\boldsymbol{\theta}}(\boldsymbol{t},\boldsymbol{x})\frac{1}{\eta}\sum_{j=1}^{d}\Big(1-\frac{1}{1-\eta\mathfrak{C}(x_j,t_j)}\Big)\\
&= \frac{d}{\eta}\underbrace{\Bigg(\prod_{l=1}^{d}\Big[1-\eta\mathfrak{C}(x_l,t_l)\Big]\Bigg)}_{C_{\boldsymbol{\theta}}(\boldsymbol{t},\boldsymbol{x})}\Bigg(1-\frac{1}{d}\sum_{j=1}^{d}\frac{1}{1-\eta\mathfrak{C}(x_j,t_j)}\Bigg)
\end{aligned}
$$

3.5.1.1 When η_j fixed per dimension j. In this case, we will have a vector of length d shape parameters. For the dimension j,

$$
\begin{aligned}
\frac{\partial}{\partial\eta_j}C_{\boldsymbol{\theta}}(\boldsymbol{t},\boldsymbol{x}) &= \frac{\partial}{\partial\eta_j}\prod_{l=1}^{d}\Big[1-\eta_l\mathfrak{C}(x_l,t_l)\Big]\\
&= \prod_{l=1,l\neq j}^{d}\Big[1-\eta_l\mathfrak{C}(x_l,t_l)\Big]\Big(-\mathfrak{C}(x_j,t_j)\Big)\\
&= \prod_{l=1}^{d}\Big[1-\eta_l\mathfrak{C}(x_l,t_l)\Big]\frac{\Big(-\mathfrak{C}(x_j,t_j)\Big)}{1-\eta_j\mathfrak{C}(x_j,t_j)}\\
&= C_{\boldsymbol{\theta}}(\boldsymbol{t},\boldsymbol{x})\frac{1}{\eta_j}\frac{\Big(1-\eta_j\mathfrak{C}(x_j,t_j)-1\Big)}{1-\eta_j\mathfrak{C}(x_j,t_j)}\\
&= C_{\boldsymbol{\theta}}(\boldsymbol{t},\boldsymbol{x})\frac{1}{\eta_j}\Big(1-\frac{1}{1-\eta_j\mathfrak{C}(x_j,t_j)}\Big)\\
&= \frac{1}{\eta_j}\underbrace{\Bigg(\prod_{l=1}^{d}\Big[1-\eta\mathfrak{C}(x_l,t_l)\Big]\Bigg)}_{C_{\boldsymbol{\theta}}(\boldsymbol{t},\boldsymbol{x})}\Big(1-\frac{1}{1-\eta_j\mathfrak{C}(x_j,t_j)}\Big)
\end{aligned}
$$

Please note the above derivatives do not depend on $\mathfrak{C}(x,t)$ and most importantly these computations are applicable to any product kernel of the form (3.10).

3.5.2 Shape parameter search using steepest descent. Having obtained the derivative, we can easily implement the steepest descent search introduced in Section 2.2.1.1

$$\eta^{(j+1)} = \eta^{(j)} - \nu\frac{\partial}{\partial\eta}\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{y})$$

CHAPTER 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, $C_{\boldsymbol{\theta}}$ and designs, $\{\boldsymbol{x}_i\}_{i=1}^n$ that satisfy assumptions (3.2). In this chapter, we demonstrate such a covariance kernel and matching design. When periodic shift-invariant kernels are combined with rank-1 Lattice nodes as design, we get circulant Gram matrix. We will also satisfy assumption (3.4). To make the computations simpler and facilitate the fast transform, it is assumed in this section and the next that n is power of 2.

4.1 Extensible Integration Lattice Node Sets

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

$$\boldsymbol{x}_i = \boldsymbol{h}\phi(i-1) + \boldsymbol{\Delta} \mod \mathbf{1}, \qquad i \in \mathbb{N}. \tag{4.1}$$

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	$\cdots$

(4.2)

A random shift $\boldsymbol{\Delta}$ is added to $\boldsymbol{x}_i$ to avoid the origin zero in our point set. However, this shift will preserve the discrepancy properties of $\boldsymbol{x}_i$. These are called

rank-1 lattices. rank-1 lattices with the module one addition have a very desirable group structure that helps to satisfy fast transform kernel assumptions.

An example of 64 nodes is given in Figure 4.1. 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 [8] and [16] for more on extensible integration lattices.

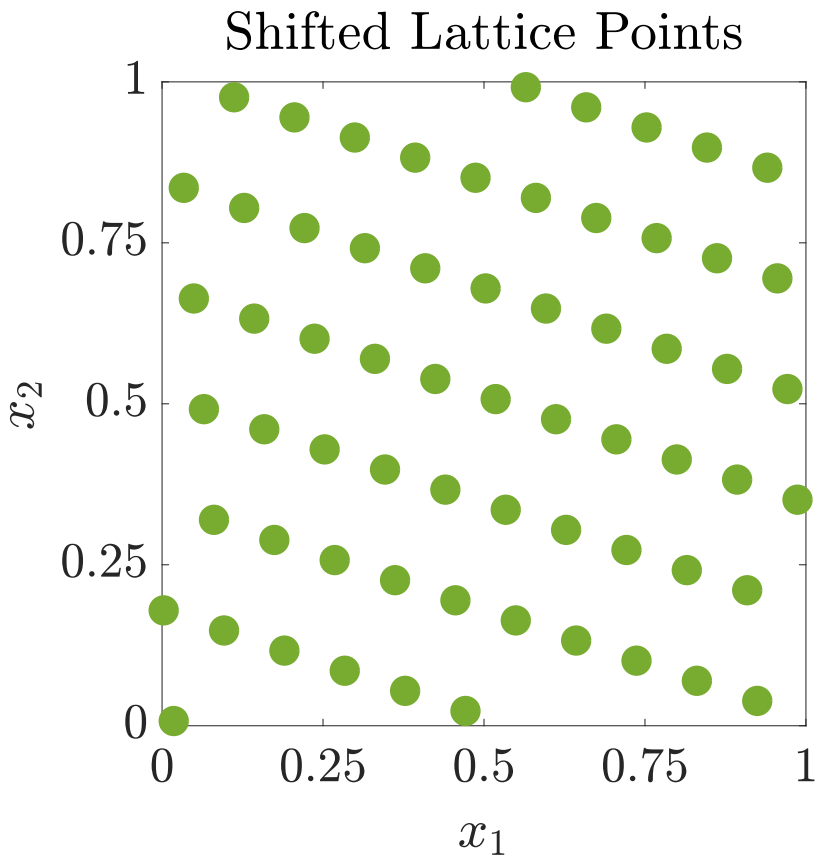

Figure 4.1. Example of a shifted integration lattice node set in $d = 2$

4.2 Shift Invariant Kernels

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

$$C_{\boldsymbol{\theta}}(\boldsymbol{t}, \boldsymbol{x}) = K_{\boldsymbol{\theta}}(\boldsymbol{t} - \boldsymbol{x} \bmod \mathbf{1}). \tag{4.3}$$

This is called a *shift invariant kernel* because shifting both arguments of the covariance function by the same amount leaves the value unchanged. By a proper scaling of the kernel $K_{\boldsymbol{\theta}}$ it follows that assumption (3.4) is satisfied. Of course, $K_{\boldsymbol{\theta}}$ must also be of the form that ensures that $C_{\boldsymbol{\theta}}$ is symmetric and positive definite, as assumed in (2.1).

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

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

where d is number of dimensions and $\alpha_{\boldsymbol{k}}$ is a scalar. The Gram matrix formed by this kernel is Hermitian. The *shape parameter* $\boldsymbol{\theta}$ changes the kernel's shape, so that the function space spanned by the kernel closely resembles the space bearing the integrand. This form of the kernel is very convenient to use in any analytical derivations, but not suitable for use with finite precision computers as this involves infinite sum. If the coefficients are chosen as

$$\alpha_{\boldsymbol{k},\boldsymbol{\theta}} := \prod_{l=1}^{d} \frac{1}{\max(\frac{|k_l|}{\eta_l},1)^r_{\eta_l\leq 1}}\,, \quad \text{with } \alpha_{\boldsymbol{0},\boldsymbol{\theta}} = 1,$$

then there exists a simpler closed form expression without infinite sum

$$C_{\boldsymbol{\theta}}(\boldsymbol{t},\boldsymbol{x}) = \prod_{l=1}^{d}\Big[1-(-1)^r\eta B_{2r}(|x_l-t_l|)\Big], \quad \forall \boldsymbol{t},\boldsymbol{x}\in[0,1]^d, \tag{4.4}$$

$$\boldsymbol{\theta} = (r,\eta),\ r\in\mathbb{N},\ \eta>0. \tag{4.5}$$

Symmetric, periodic, positive definite kernels of this form appear in [8] and [12]. The Bernoulli polynomials $B_r(x)$ are described in [27, Chapter 24]

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

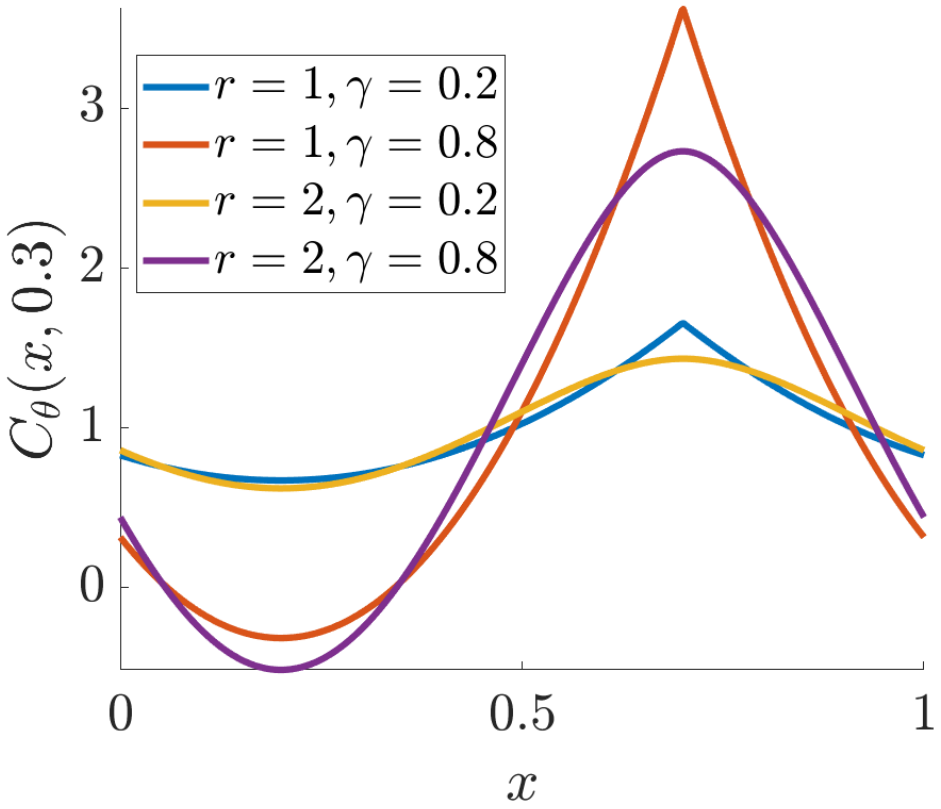

Figure 4.2. Shift invariant kernel in dim=1 shifted by 0.3 to show the discontinuity
JR: Update plot with γ to η

Larger 'r' implies a greater degree of smoothness of the kernel. Larger η implies greater fluctuations of the output with respect to the input. Plots of $C(\cdot, 0.3)$ are given in Figure 4.2 for various r and η values.

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

$$\begin{aligned}\mathsf{C} &= \big(C(\boldsymbol{x}_i, \boldsymbol{x}_j)\big)_{i,j=1}^{n} \\ &= \Big(K(\boldsymbol{h}(\phi(i-1) - \phi(j-1)) \bmod \mathbf{1})\Big)_{i,j=1}^{n}.\end{aligned}$$

We now demonstrate that the eigenvector matrix for C is

$$\mathsf{V} = \Big(\mathrm{e}^{2\pi n\sqrt{-1}\phi(i-1)\phi(j-1)}\Big)_{i=1}^{n}. \tag{4.6}$$

Assumption (3.2b) 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 (3.1).

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

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

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 $(j-1)/n$:

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

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

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

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

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

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

This establishes $\mathsf{V}^H\mathsf{C}\mathsf{V}$ as a diagonal matrix whose diagonal elements are n times the eigenvalues, i.e., $\lambda_k = \omega_{k,k}/n$. Furthermore, V is the matrix of eigenvectors, which satisfies assumption (3.2a).

4.2.2 Iterative Computation of the Fast Transform. Assumption (3.2a) is that computing $\mathsf{V}^H\boldsymbol{b}$ requires only $\mathcal{O}(n\log(n))$ operations. Recall that we assume that n is a power of 2. This can be accomplished by an iterative algorithm. Let $\mathsf{V}^{(n)}$ denote the $n \times n$ matrix V defined in (4.6). We show how to compute $\mathsf{V}^{(2n)H}\boldsymbol{b}$ quickly for all $\boldsymbol{b} \in \mathbb{R}^{2n}$ assuming that $\mathsf{V}^{(n)H}\boldsymbol{b}$ can be computed quickly for all $\boldsymbol{b} \in \mathbb{R}^n$.

From the definition of the van der Corput sequence in (4.2) it follows that

$$\phi(2i) = \phi(i)/2,\ \phi(2i+1) = [\phi(i)+1]/2, \quad i \in \mathbb{N}_0 \tag{4.7}$$

$$\phi(i+n) = \phi(i) + 1/(2n), \qquad i = 0, \ldots, n-1, \tag{4.8}$$

$$n\phi(i) \in \mathbb{N}_0, \qquad i = 0, \ldots, n-1, \tag{4.9}$$

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

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

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

It follows from these definitions and the definition of V in (4.6) that

$$\begin{aligned}
\widetilde{\boldsymbol{b}}^{(1)} &= \left(\sum_{j=1}^{2n} \mathrm{e}^{-4\pi n\sqrt{-1}\phi(2i-2)\phi(j-1)} b_j\right)_{i=1}^{n} \\
&= \left(\sum_{j=1}^{2n} \mathrm{e}^{-2\pi n\sqrt{-1}\phi(i-1)\phi(j-1)} b_j\right)_{i=1}^{n} \qquad \text{by (4.7)} \\
&= \left(\sum_{j=1}^{n} \mathrm{e}^{-2\pi n\sqrt{-1}\phi(i-1)\phi(j-1)} b_j\right)_{i=1}^{n} + \left(\sum_{j=1}^{n} \mathrm{e}^{-2\pi n\sqrt{-1}\phi(i-1)\phi(n+j-1)} b_{n+j}\right)_{i=1}^{n} \\
&= \mathsf{V}^{(n)H}\boldsymbol{b}^{(1)} + \left(\mathrm{e}^{-\pi\sqrt{-1}\phi(i-1)} s \sum_{j=1}^{n} \mathrm{e}^{-2\pi n\sqrt{-1}\phi(i-1)\phi(j-1)} b_{n+j}\right)_{i=1}^{n} \qquad \text{by (4.8)} \\
&= \mathsf{V}^{(n)H}\boldsymbol{b}^{(1)} + \left(\mathrm{e}^{-\pi\sqrt{-1}\phi(i-1)}\right)_{i=1}^{n} \odot \left(\mathsf{V}^{(n)H}\boldsymbol{b}^{(2)}\right),
\end{aligned}$$

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

$$\begin{aligned}
\widetilde{\boldsymbol{b}}^{(2)} &= \left(\sum_{j=1}^{2n} \mathrm{e}^{-4\pi n\sqrt{-1}\phi(2i-1)\phi(j-1)} b_j\right)_{i=1}^{n} \\
&= \left(\sum_{j=1}^{2n} \mathrm{e}^{-2\pi n\sqrt{-1}[\phi(i-1)+1]\phi(j-1)} b_j\right)_{i=1}^{n} \qquad \text{by (4.7)} \\
&= \left(\sum_{j=1}^{n} \mathrm{e}^{-2\pi n\sqrt{-1}[\phi(i-1)+1]\phi(j-1)} b_j\right)_{i=1}^{n} + \left(\sum_{j=1}^{n} \mathrm{e}^{-2\pi n\sqrt{-1}[\phi(i-1)+1]\phi(n+j-1)} b_{n+j}\right)_{i=1}^{n} \\
&= \mathsf{V}^{(n)H}\boldsymbol{b}^{(1)} + \left(\mathrm{e}^{-\pi\sqrt{-1}[\phi(i-1)+1]} \sum_{j=1}^{n} \mathrm{e}^{-2\pi n\sqrt{-1}\phi(i-1)\phi(j-1)} b_{n+j}\right)_{i=1}^{n} \\
&\qquad\qquad \text{by (4.8) and (4.9)} \\
&= \mathsf{V}^{(n)H}\boldsymbol{b}^{(1)} - \left(\mathrm{e}^{-\pi\sqrt{-1}\phi(i-1)}\right)_{i=1}^{n} \odot \left(\mathsf{V}^{(n)H}\boldsymbol{b}^{(2)}\right).
\end{aligned}$$

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

4.3 Continuous valued kernel order

JR: Need better and more convincing motivation

We assumed so far, order of the shift-invariant kernel is an even valued integer and also fixed. It requires the practitioner to be aware the smoothness of the integrand to precisely hand pick the kernel order to match the smoothness of the integrand. However, It is not possible to know the smoothness of the integrand in most of practical applications. This constraint limits the ability to vary the kernel smoothness to match the integrand like the shape parameter is chosen to match.

Integer kernel order is not suitable to optimally search by an optimization algorithm. As a consequence, we usually end up choosing a higher kernel order when the integrand is not smooth or lower kernel order when the integrand is very smooth. Often it leads to longer computation time or poor accuracy in the numerical integration. Here we explore an alternative form of the kernel which allows the kernel order to be a positive continuous greater than one. Let us recollect the kernel used so far,

$$C_{\boldsymbol{\theta}}(\boldsymbol{x},\boldsymbol{t}) := \sum_{\boldsymbol{k}\in\mathbb{Z}^d} \alpha_{\boldsymbol{k},\boldsymbol{\theta}} e^{2\pi\sqrt{-1}\boldsymbol{k}^T\boldsymbol{x}} e^{-2\pi\sqrt{-1}\boldsymbol{k}^T\boldsymbol{t}}, \quad \alpha_{\boldsymbol{k},\boldsymbol{\theta}} = \prod_{l=1}^{d} \frac{1}{\max(\frac{|k_l|}{\eta_l},1)^r_{\eta_l\leq 1}}$$

where $\theta = (\eta, r)$. To make the derivations easier to follow, let us start with the dimension $d = 1$ case,

$$C_{\boldsymbol{\theta}}(x,t) = 1 + \eta \sum_{k\in\mathbb{Z},k\neq 0} \frac{1}{|k|^r} e^{2\pi\sqrt{-1}kx} e^{-2\pi\sqrt{-1}kt}$$

4.3.1 Exponentially decaying kernel. We propose the following alternative form of which can also provide exponential decay

$$C_{\boldsymbol{\theta}}(x,t) = 1 + \eta \sum_{k\in\mathbb{Z},k\neq 0} b^{|k|} e^{2\pi\sqrt{-1}k(x-t)}, \quad \text{with} \quad 0 < b < 1$$

which can be rewritten as

$$C_{\boldsymbol{\theta}}(x,t) = 1 + \eta \sum_{k\in\mathbb{Z},k\neq 0} e^{2\pi\sqrt{-1}k(x-t)+|k|\log(b)}$$

$$
\begin{aligned}
&=1+\eta\left(\sum_{k=1}^{\infty}e^{2\pi\sqrt{-1}k(x-t)+|k|\log(b)}+\sum_{k=\infty}^{-1}e^{2\pi\sqrt{-1}k(x-t)+|k|\log(b)}\right),\\
&=1+\eta\left(\sum_{k=1}^{\infty}e^{2\pi\sqrt{-1}k(x-t)+|k|\log(b)}+\sum_{k=-\infty}^{-1}e^{2\pi\sqrt{-1}k(x-t)+|k|\log(b)}\right),\\
&=1+\eta\left(\underbrace{\sum_{k=1}^{\infty}e^{2\pi\sqrt{-1}k(x-t)+|k|\log(b)}}_{*}+\sum_{k=1}^{\infty}e^{-2\pi\sqrt{-1}k(x-t)+|k|\log(b)}\right),
\end{aligned}
$$

Let us focus on the term $(*)$,

$$
\begin{aligned}
(*)&=\sum_{k=1}^{\infty}e^{2\pi\sqrt{-1}k(x-t)+|k|\log(b)}=\sum_{k=1}^{\infty}\left[e^{2\pi\sqrt{-1}(x-t)+\log(b)}\right]^k\\
&=\frac{e^{2\pi\sqrt{-1}(x-t)+\log(b)}}{1-e^{2\pi\sqrt{-1}(x-t)+\log(b)}}=\frac{1}{e^{-2\pi\sqrt{-1}(x-t)-\log(b)}-1}\\
&=\frac{1}{b^{-1}e^{-2\pi\sqrt{-1}(x-t)}-1}
\end{aligned}
$$

Using this result

$$
\begin{aligned}
C_{\boldsymbol{\theta}}(x,t)&=1+\eta\left(\frac{1}{b^{-1}e^{-2\pi\sqrt{-1}(x-t)}-1}+\frac{1}{b^{-1}e^{2\pi\sqrt{-1}(x-t)}-1}\right),\\
&=1+\eta\left(\frac{b^{-1}\left(e^{2\pi\sqrt{-1}(x-t)}+e^{-2\pi\sqrt{-1}(x-t)}\right)-2}{b^{-2}-b^{-1}\left(e^{2\pi\sqrt{-1}(x-t)}+e^{-2\pi\sqrt{-1}(x-t)}\right)+1}\right),\\
&=1+\eta\left(\frac{2b^{-1}\cos(2\pi\sqrt{-1}(x-t))-2}{b^{-2}-2b^{-1}\cos(2\pi\sqrt{-1}(x-t))+1}\right),\\
&=1+2\eta b\left(\frac{\cos(2\pi\sqrt{-1}(x-t))-b}{b^2-2b\cos(2\pi\sqrt{-1}(x-t))+1}\right),
\end{aligned}
$$

Using the fact $\cos^2(t)+\sin^2(t)=1$

$$
C_{\boldsymbol{\theta}}(x,t)=1+2\eta b\left(\frac{\cos(2\pi\sqrt{-1}(x-t))-b}{\left[\cos(2\pi\sqrt{-1}(x-t))-b\right]^2+\sin^2(2\pi\sqrt{-1}(x-t))}\right),
$$

We need to have $\eta>0$ and $0<b<1$ while searching for the optimum value, so we use the following transformations

$$
\begin{aligned}
b:&\quad g(t)=\frac{1}{1+e^{-t}}, && g:(-\infty,\infty)\to(0,1)\\
\theta:&\quad h(t)=e^{t}, && h:(-\infty,\infty)\to(0,\infty)
\end{aligned}
$$

4.3.2 Truncated series kernel. If we use the original definition of the kernel, we could make the kernel order r explicit such that it does not have to be an even integer, which was the constraint previously. For $d = 1$,

$$C_{\boldsymbol{\theta}}(x,t) = 1 + \eta \sum_{k\in\mathbb{Z},k\neq 0} \frac{1}{|k|^r} e^{2\pi\sqrt{-1}k(x-t)}$$

where $\theta = (\eta, r)$ But it has the infinite sum which cannot be used in practical applications, so we truncate to length N,

$$C_{\boldsymbol{\theta},N}(x,t) = 1 + \eta \sum_{k=-N/2}^{N/2} \frac{1}{|k|^r} e^{2\pi\sqrt{-1}k(x-t)}$$

The Gram matrix is written as

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

The first column of the Gram matrix is

$$\begin{aligned} \boldsymbol{C}_{\boldsymbol{\theta},N} &= \left(C_{\boldsymbol{\theta},N}(\boldsymbol{x}_i, \boldsymbol{x}_1) \right)_{i=1}^{n} \\ &= \left(\prod_{l=1}^{d} \left[1 + \eta_l \sum_{k=-N/2,k\neq 0}^{N/2} \frac{1}{|k_l|^r} e^{2\pi\sqrt{-1}k_l(x_{il}-x_{1l})} \right] \right)_{i=1}^{n} \end{aligned}$$

where d is number of dimensions and n is the number of points. But the direct computation of involves nN computations, or n^2 computations if $N \approx n$. We can reduce the computations to $\mathcal{O}(n \log n)$ using the FFT. Let us define

$$\mathfrak{C}(t,r) = \sum_{k=-N/2}^{N/2} \frac{1}{|k|^r} e^{2\pi\sqrt{-1}k\,t}$$

Using the notation $\mathfrak{C}$, rewrite

$$\boldsymbol{C}_{\boldsymbol{\theta},N} = \left(\prod_{l=1}^{d} \left[1 + \eta \mathfrak{C}(|x_{il} - x_{1l}|) \right] \right)_{i=1}^{n}$$

By the definition of lattice points from (4.1), we can observe $x_{il} - x_{1l} \in \{0, \frac{1}{n}, \frac{2}{n}, \ldots \frac{n-1}{n}\}$. This can be used to rewrite in a much simpler form,

$$\mathfrak{C}\left(\frac{j}{N}, r\right) = \sum_{k=-N/2}^{N/2} \frac{1}{|k|^r} e^{2\pi\sqrt{-1}k(\frac{j}{N})}, \quad \text{where } j = 0, 1, \ldots N-1$$

These notations were introduced to help us compute the discrete Fourier transform of g easily.

$$\begin{aligned}\widetilde{\mathfrak{C}}(m,r) &:= \sum_{j=0}^{N-1} \mathfrak{C}(j/n,r)e^{-2\pi\sqrt{-1}jm/N} \\ &= \sum_{k=-N/2,k\neq 0}^{N/2} \sum_{j=0}^{N-1} \frac{1}{|k|^r} e^{2\pi\sqrt{-1}kj/N} e^{-2\pi\sqrt{-1}jm/N} \\ &= \sum_{k=-N/2,k\neq 0}^{N/2} N\delta_{k-m \bmod N,0} \ \frac{1}{|k|^r}.\end{aligned}$$

By simply observing the above result, we can analytically compute the values

$$\widetilde{\mathfrak{C}}(m,r) = \begin{cases} 0, & \text{for} \quad m = 0, \\ \frac{N}{|m|^r}, & \text{for} \quad m = 1,\ldots,N/2-1, \\ \frac{N}{|N-m|^r}, & \text{for} \quad m = N/2,\ldots,N-1 \end{cases} \tag{4.10}$$

Having these values, We can easily back-compute $\mathfrak{C}$ using inverse discrete Fourier transform. It can be easily shown that inverse DFT of $\widetilde{\mathfrak{C}}_r$ returns $\mathfrak{C}$ indeed,

$$\begin{aligned}&\frac{1}{N}\sum_{m=0}^{N-1} \widetilde{\mathfrak{C}}_r(m) e^{2\pi\sqrt{-1}lm/N} \\ &= \frac{1}{N}\sum_{m=0}^{N-1}\sum_{j=0}^{N-1} \mathfrak{C}_r(j/n) e^{-2\pi\sqrt{-1}jm/N} e^{2\pi\sqrt{-1}lm/N} \\ &= \frac{1}{N}\sum_{j=0}^{N-1} \mathfrak{C}_r(j/n) \sum_{m=0}^{N-1} e^{2\pi\sqrt{-1}(l-j)m/N} \\ &= \frac{1}{N}\sum_{j=0}^{N-1} \mathfrak{C}_r(j/n) N\delta_{(l-j) \bmod N,0} \\ &= \mathfrak{C}_r(l/n), \quad \text{for} \quad l = 0,\ldots,N-1\end{aligned}$$

This shows to compute the n values of $\left(C_{\boldsymbol{\theta},N}(\boldsymbol{x}_i,\boldsymbol{x}_1)\right)_{i=1}^{n}$, we need $N = n$. Here we summarize the steps to compute $\mathfrak{C}$ using FFT

1. Analytically compute $\left(\widetilde{\mathfrak{C}}_r(m)\right)_{m=0}^{N-1}$, the discrete Fourier coefficients of $(\mathfrak{C}(j/N))_{j=0}^{N-1}$ using (4.10)

2. Take inverse FFT of $\widetilde{\mathfrak{C}}_r$ to get $\mathfrak{C}$

where the computational cost is $\mathcal{O}(n \log n)$ instead of $\mathcal{O}(n^2)$. Another major advantage is, the FFT approach is numerically stable than the direct sum approach.

4.4 Summary

We summarize the results of this section and the previous one as follows:

Proposition 1 *Any periodic, symmetric, positive definite, shift-invariant covariance kernel of the form* (4.3) *scaled to satisfy* (3.4)*, when matched with rank-1 lattice data-sites, must satisfy assumptions* (3.2)*. The* fast Fourier transform *(FFT) can be used to expedite the estimates of* $\boldsymbol{\theta}$ *in* (6.1) *and the credible interval widths* (6.2) *in* $\mathcal{O}(n \log(n))$ *operations. The cubature,* $\widehat{\mu}$*, is just the sample mean.*

We have implemented the fast adaptive Bayesian cubature algorithm in MATLAB as part of the Guaranteed Adaptive Integration Library (GAIL) [3] as `cubBayesLattice_g`. This algorithm uses the kernel defined in (4.4) with $r = 1, 2$ and the periodizing variable transforms in Section 4.5. The rank-1 lattice node generator is taken from [23] (`exod2_base2_m20`).

4.5 Periodizing Variable Transformations

The shift-invariant covariance kernels underlying our `cubBayesLattice_g` Bayesian cubature assume that the integrand has a degree of periodicity, with the smoothness assumed depending on the smoothness of the kernel. In other-words, non-periodic functions do not live in the space spanned by the shift-invariant covariance kernels. While integrands arising in practice may be smooth, they might not be periodic. Variable transformation or periodization transform techniques are typically used to enforce the periodicity in multi-dimensional numerical integrations where

boundary conditions needs to be enforced. These transformations could be either polynomial, exponential and also trigonometric nature. Some of the most popular transformation are listed here for reference. Suppose that the original integral has been expressed as

$$\mu := \int_{[0,1]^d} g(\boldsymbol{t})\,\mathrm{d}\boldsymbol{t}$$

where g has sufficient smoothness, but lacks periodicity. The Baker's transform,

$$\boldsymbol{\Psi} : \boldsymbol{x} \mapsto (\Psi(x_1), \ldots, \Psi(x_d)), \quad \Psi(x) = 1 - 2\,|x - 1/2|\,, \tag{4.11}$$

allows us to write μ in the form of (1.1), where $f(\boldsymbol{x}) = g(\boldsymbol{\Psi}(\boldsymbol{x}))$. We must emphasize, $\Psi(x)$ for the Baker's transform does not have any derivative, so we presented it separately.

A family of smoother variable transforms, i.e., with derivatives take the form

$$\boldsymbol{\Psi} : \boldsymbol{x} \mapsto (\Psi(x_1), \ldots, \Psi(x_d)), \quad \Psi : [0,1] \mapsto [0,1],$$

which allows us to write μ in the form of (1.1) with

$$f(\boldsymbol{x}) = g(\boldsymbol{\Psi}(\boldsymbol{x})) \prod_{\ell=1}^{d} \Psi'(x_l).$$

If Ψ is sufficiently smooth, $\lim_{x\downarrow 0} x^{-r}\Psi'(x) = \lim_{x\uparrow 1}(x-1)^{-r}\Psi'(x) = 0$ for $r \in \mathbb{N}_0$, and $g \in C^{(r,\ldots,r)}[0,1]^d$, then f has continuous, periodic derivatives up to order r in each direction. Examples of this kind of transform include [31]:

$$\begin{aligned}
C^0 &: \Psi(x) = 3x^2 - 2x^3, \quad \Psi'(x) = 6x(1-x),\\
C^1 &: \Psi(x) = x^3(10 - 15x + 6x^2),\\
&\qquad\qquad \Psi'(x) = 30x^2(1-x)^2\\
\text{Sidi's } C^1 &: \Psi(x) = x - \frac{\sin(2\pi x)}{2\pi},\\
&\qquad\qquad \Psi'(x) = 1 - \cos(2\pi x),\\
\text{Sidi's } C^2 &: \Psi(x) = \frac{8 - 9\cos(\pi x) + \cos(3\pi x)}{16},
\end{aligned}$$

$$\Psi'(x) = \frac{3\pi[3\sin(\pi x) - \sin(3\pi x)]}{16}.$$

These transforms vary in terms of computational complexity and accuracy, shall be chosen on a need basis. Baker's transform or called tent map in each coordinate. It preserves only continuity but it is easier to compute and it does not product term up to the length dimension of the integrand, making it more numerically stable. C^0 is a polynomial transformation only and ensures periodicity of function. C^1 is a polynomial transformation and preserving the first derivative. Sidi's C^1, a transform which uses trigonometric Sine, preserves the first derivative and is, in general, a better option than C^1. Sidi's C^2, also a transform which uses trigonometric Sine, preserves upto second derivative. We use this when smoothness of Sidi's C^1 is not sufficient and need to preserve upto second derivative.

Periodizing variable transforms are used for the numerical examples below. In some cases they can speed the convergence of the Bayesian cubature.

CHAPTER 5

SOBOL NETS AND WALSH KERNELS

5.1 Sobol Nets

The previous section showed an automatic Bayesian cubature algorithm using rank-1 lattice nodes and shift-invariant kernels. In this chapter, We demonstrate a second approach to formulate fast transform using matching kernel and point sets. Scrambled Sobol nets and Walsh kernels are paired to achieve $\mathcal{O}(N^{-1+\epsilon})$ order error convergence. Sobol nets [33] are low discrepancy quasi-random points, used extensively in numerical integration, simulation, and optimization.

Nets were developed to provide deterministic sample points for quasi-Monte Carlo rules [22]. The (t, m, d)-nets in base b introduced by Niederreiter are such point sets consist of b^m points in $[0, 1)^d$, whose quality is governed by t, in particular, lower values of t correspond to (t, m, d)-nets of higher quality [1]. Digital (t, m, d)-nets are a special case of (t, m, d)-nets, constructed using matrix-vector multiplications over finite fields.

Definition 1 *Let $d \geq 1$, $b \geq 2$ and given two integers $0 \leq t \leq m$, then a (t, m, d)-net in base b is a sequence of x_i of b^m points of $[0, 1]^d$ with the property that every elementary interval in base b of volume b^{t-m} contains precisely b^t points from x_i. Sobol nets are special case of (t, m, d)-nets when base $b = 2$. Digital sequences are infinite length digital nets.*

Digital sequences are defined using digitwise operations. Let b be a prime number. Digitwise addition, $\oplus$, negation $\ominus$, are defined in terms of b-ary expansions of points in $[0, 1)^d]$,

$$\boldsymbol{x} = \left(\sum_{l=1}^{\infty} x_{jl} b^{-l} \right)_{j=1}^{d}, \quad \boldsymbol{t} = \left(\sum_{l=1}^{\infty} t_{jl} b^{-l} \right)_{j=1}^{d}, \quad x_{jl}, t_{jl} \in \mathbb{F}_b := \{0, \ldots, b-1\},$$

Digital nets have a group structure under the digit-wise addition, which is a very useful property to use with Bayesian cubature, especially to speedup computations. Let $\{\boldsymbol{x}_i\}_{i=0}^{2^m-1}$ be the digital net, then

$$\forall i_i, i_2 \in \{0, \dots, 2^m - 1\}, \quad \boldsymbol{x}_{i_1} \oplus \boldsymbol{x}_{i_2} = \boldsymbol{x}_{i_3}, \quad \text{for some } i_3 \in \{0, \dots, 2^m - 1\},$$

where

$$\boldsymbol{x} \oplus \boldsymbol{t} = \left(\sum_{l=1}^{\infty} [x_{jl} + t_{jl} \bmod b] b^{-l} \bmod 1 \right)_{j=1}^{d}$$

An example of 64 nodes is given in Figure 4.1. The even coverage of the unit cube is ensured by a well chosen generating matrix. The choice of generating vector is typically done offline by computer search. See [21] and [24] for more on generating matrices.

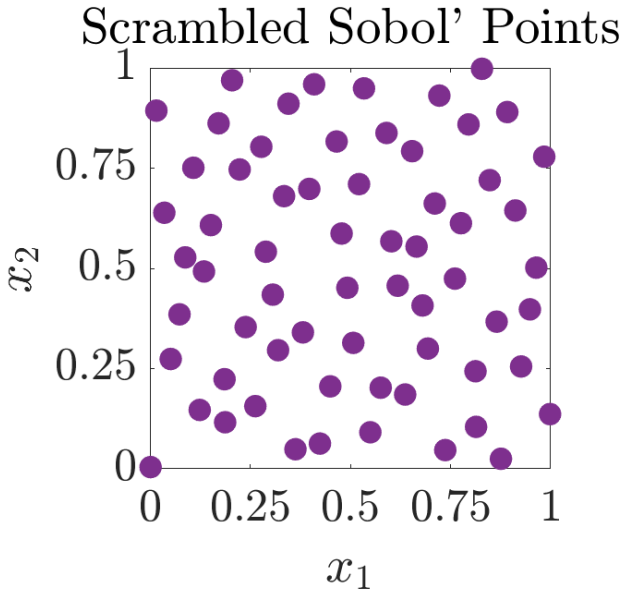

Figure 5.1. Example of a scrambled Sobol node set in $d = 2$

5.2 Walsh Kernels

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

5.2.1 Walsh functions. Like Fourier transform naturally occurred with Lattice points (Section 4.2), Hadamard-Walsh transform happens for the nets, which we will call simply Walsh transform. Walsh transform is defined using Walsh functions. Define $\mathbb{N} := \{1, 2, \dots\}$ and $\mathbb{N}_0 : \{0, 1, 2, \dots\}$. The one-dimensional Walsh functions in base b are defined as

$$\mathrm{wal}_{b,h}(x) := e^{2\pi i(x_1 h_0 + x_2 h_1 + \cdots + x_n h_{n-1})/b} = e^{2\pi i[\vec{h}]_n^T [\vec{x}]_n / b}$$

for $x \in [0, 1)$ and $h \in \mathbb{N}_0$ and the unique base b expansions $x = \sum_{i \geq 1} x_i b^{-i} = (0.x_1 x_2 \dots)_b$, and $h = \sum_{i \geq 1} h_i b^{-i} = (0.h_1 h_2 \dots)_b, [\vec{h}] = (h_1, \dots h_n)^T$, with n at least as large as the number of digits to represent x or h. Multivariate Walsh functions are defined as the product of the one-dimensional Walsh functions

$$\mathrm{wal}_{b,\boldsymbol{h}}(\boldsymbol{x}) := \prod_{j=1}^{s} \mathrm{wal}_{b,h_j}(x_j)$$

5.2.2 Walsh kernels. Consider the kernels of the form [25]

$$C(\boldsymbol{x}, \boldsymbol{t}) = 1 + \theta \omega_r(\boldsymbol{x} \ominus \boldsymbol{t}) \tag{5.1}$$

where

$$\omega_r(x) = \sum_{h=1}^{\infty} \frac{\mathrm{wal}_{b,h}(x)}{b^{qr\lfloor \log_b h \rfloor}}$$

where r is kernel order, $\frac{1}{p} + \frac{1}{q} = 1$ are Hölder inequality coefficients. Explicit expression available for ω_r [25] in case of order $r = 1$, $b = 2$ and Hilbert space setup $q = 2$

$$\omega_1(\boldsymbol{x}) = \sum_{h=1}^{\infty} \frac{\mathrm{wal}_{b,h}(x)}{b^{2\lfloor \log_b h \rfloor}} = \frac{1}{6} - 2^{\lfloor \log_2 \boldsymbol{x} \rfloor - 1}. \tag{5.2}$$

We have implemented a fast adaptive Bayesian cubature algorithm using the kernel (5.1) with $r = 1$ and Sobol points in MATLAB as part of the Guaranteed Adaptive Integration Library (GAIL) [3] as `cubBayesNet_g`. The Sobol points used in

this algorithm are generated using Matlab's builtin function `sobolset` and scrambled using Matlab function `scramble`.

Higher order digital nets are a modified (t, m, s)-nets, introduced in [7] can be used to numerically integrate smoother functions which are not necessarily periodic, but have square integrable mixed partial derivatives of order α, at a rate of $\mathcal{O}(N^{-\alpha})$ multiplied by a power of a $\log N$ factor using rules corresponding to the modified (t, m, s)-nets. We want to emphasizes that quasi-Monte Carlo rules based on these point sets can achieve convergence rates faster than $\mathcal{O}(N^{-1})$.

Higher order digital nets are constructed using matrix-vector multiplications over finite fields. Bayesian cubatures using higher order digital nets are the topic for future research.

CHAPTER 6

NUMERICAL IMPLEMENTATION

6.1 Overcoming the Cancellation Error

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

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

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

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

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

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

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

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

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

Computing the eigenvalues of $\mathring{\mathsf{C}}$ via the procedure given in (3.3) yields $\mathring{\lambda}_1 = \lambda_1 - n, \lambda_2, \ldots, \lambda_n$. The estimates of $\boldsymbol{\theta}$ are computed in terms of the eigenvalues of $\mathring{\mathsf{C}}$, so (3.5) and (3.8) become

$$\boldsymbol{\theta}_{\text{MLE}} = \underset{\boldsymbol{\theta}}{\text{argmin}} \left[\log \left(\sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i} \right) + \frac{1}{n} \sum_{i=1}^{n} \log(\lambda_i) \right], \tag{6.1a}$$

$$\boldsymbol{\theta}_{\text{GCV}} = \underset{\boldsymbol{\theta}}{\text{argmin}} \left[\log \left(\sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i^2} \right) - 2 \log \left(\sum_{i=1}^{n} \frac{1}{\lambda_i} \right) \right], \tag{6.1b}$$

where $\lambda_1 = n + \mathring{\lambda}_1$. The widths of the credible intervals in (3.6), (3.7), and (3.9) become

$$\text{err}_{\text{MLE}} = \frac{2.58}{n} \sqrt{\frac{\mathring{\lambda}_1}{\lambda_1} \sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i}}, \tag{6.2a}$$

$$\text{err}_{\text{full}} = \frac{t_{n_j-1,0.995}}{n} \sqrt{\frac{\mathring{\lambda}_1}{n-1} \sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i}}, \tag{6.2b}$$

$$\text{err}_{\text{GCV}} = \frac{2.58}{n} \sqrt{\frac{\mathring{\lambda}_1}{\lambda_1} \sum_{i=2}^{n} \frac{|\widetilde{y}_i|^2}{\lambda_i^2} \left[\frac{1}{n} \sum_{i=1}^{n} \frac{1}{\lambda_i} \right]^{-1}}. \tag{6.2c}$$

Since $\mathring{\lambda}_1 = \lambda_1 - n$ and $\lambda_1 \sim n$ it follows $\mathring{\lambda}_1/\lambda_1 \approx \mathring{\lambda}_1/(n-1)$ and is small for large n. Moreover, the credible intervals via empirical Bayes and full Bayes are similar, since $t_{n_j-1,0.995}$ is approximately 2.58.

CHAPTER 7

NUMERICAL RESULTS AND OBSERVATIONS

Bayesian cubature algorithms developed in this work are demonstrated using widely used integration examples. Three integrals were evaluated using both the algorithms `cubBayesLattice_g` and `cubBayesNet_g`. The first example shows evaluating a multivariate normal probability given the intervals. The second example shows integrating the Keister's function, and the final example shows computing an Asian arithmetic option pricing.

7.1 Testing Methodology

Four different error tolerances, ε, were set for each example, with the tolerances chosen depending on the difficulty of the problem. The nodes used in `cubBayesLattice_g` were the randomly shifted lattice rules supplied by GAIL. Where as the nodes used in `cubBayesNet_g` were the randomly scrambled Sobol rules supplied by Matlab's Sobol sequence generator.

For each integral, each tolerance, and each of our stopping criteria—empirical Bayes, full Bayes, and generalized cross-validation—our algorithm is run 100 times. For each test, the execution time is plotted against $|\mu - \widehat{\mu}|/\varepsilon$. We expect $|\mu - \widehat{\mu}|/\varepsilon$ to be no greater than one, but hope that it is not too much smaller than one, which would indicate a stopping criterion that is too conservative.

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

7.2 Multivariate Normal Probability

This example was already introduced in Section 2.5, where we used the Matérn

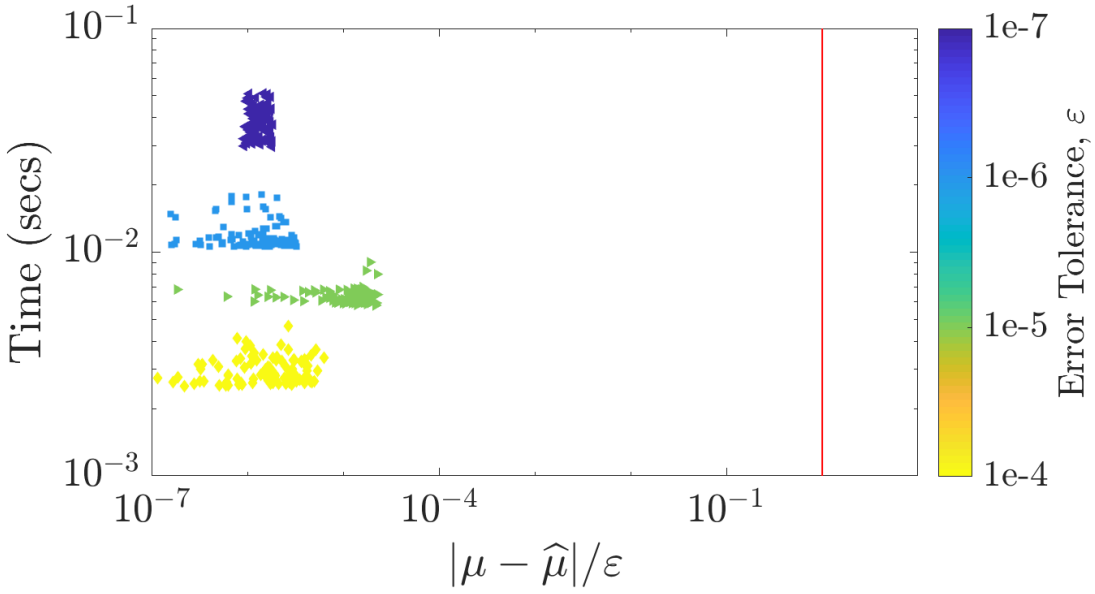

Figure 7.1. `cubBayesLattice_g`: Multivariate normal probability example using the empirical Bayes stopping criterion.

covariance kernel.

7.2.1 Using `cubBayesLattice_g`. Here we apply Sidi's C^2 periodization to f_{Genz} (2.21), and chose $d = 3$ and $r = 2$. The simulation results for this example function are summarized in Figures 7.1, 7.2, and 7.3. In all cases the Bayesian cubature returns an approximation within the prescribed error tolerance. We used the same setting as before with generic slow Bayesian cubature in Section 2.5 for comparision. For error threshold $\varepsilon = 10^{-5}$ with empirical stopping criterion, our fast algorithm takes 0.001 seconds as shown in Figure 7.1 whereas the basic algorithm takes 30 seconds as shown in Figure 2.4. Amongst the three stopping criteria, GCV achieved the results faster than others. One can also observe from the figures, the credible intervals are wider, causing the true error much smaller than requested. This could be due to the periodization transformed f_{Genz} is smoother than the $r = 2$ kernel could approximate. Using a kernel of matching smoothness could produce right credible intervals.

7.2.2 Using `cubBayesNet_g`. Here we use f_{Genz} (2.21) without any periodization,

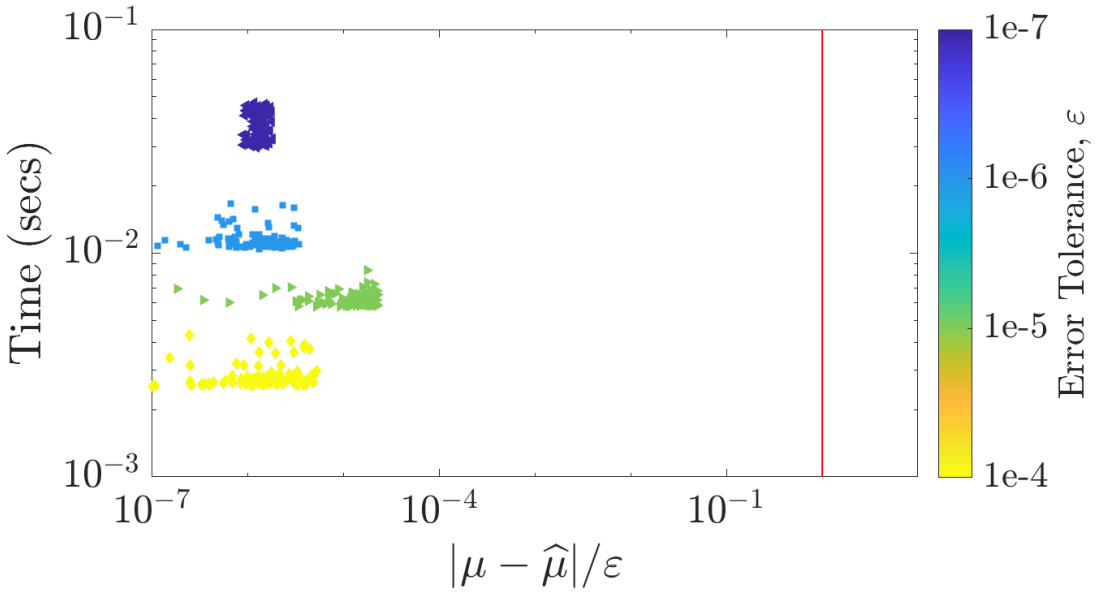

Figure 7.2. `cubBayesLattice_g`: Multivariate normal probability example using the full Bayes stopping criterion.

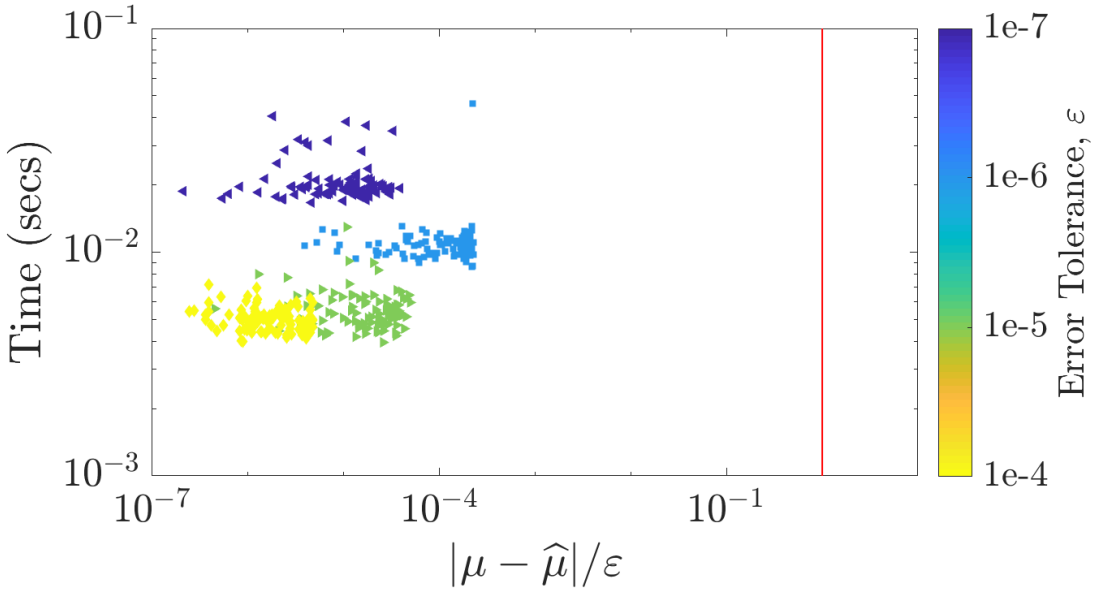

Figure 7.3. `cubBayesLattice_g`: Multivariate normal probability example using the GCV stopping criterion.

and chose $d = 3$ and $r = 1$. The simulation results for this example function are summarized in Figures 7.4, 7.5, and 7.6. In all cases the `cubBayesNet_g` returns an approximation within the prescribed error tolerance. We used the same setting as before with generic slow Bayesian cubature in Section 2.5 for comparision. For error threshold $\varepsilon = 10^{-5}$ with empirical stopping criterion, our fast algorithm takes about 2 seconds as shown in Figure 7.1 whereas the basic algorithm takes 30 seconds as shown in Figure 2.4. `cubBayesNet_g` uses fast Walsh transform which is slower in Matlab due to the way it was implemented. This is reason it takes more time than the `cubBayesLattice_g`. But comparing the number of samples, n, used for integration provides more insight which directly relates to the alogrithm's computational cost. The `cubBayesLattice_g` used $n = 16384$ samples whereas `cubBayesNet_g` used $n = 32768$ samples even with $r = 1$ order kernel.

Amongst the three stopping criteria, GCV achieved the results faster than others. One can also observe from the figures, the credible intervals are narrower than we observed in Figure 7.1. This shows `cubBayesNet_g` with $r = 1$ kernel more accurately approximates the integrand.

7.3 Keister's Example

This multidimensional integral function comes from [20] and is inspired by a physics application:

$$\begin{aligned} \mu &= \int_{\mathbb{R}^d} \cos(\|\boldsymbol{t}\|) \exp(-\|\boldsymbol{t}\|^2) \, \mathrm{d}\boldsymbol{t} \\ &= \int_{[0,1]^d} f_{\mathrm{Keister}}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}, \end{aligned} \tag{7.1}$$

where

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

and again Φ is the standard normal distribution. The true value of μ can be calculated

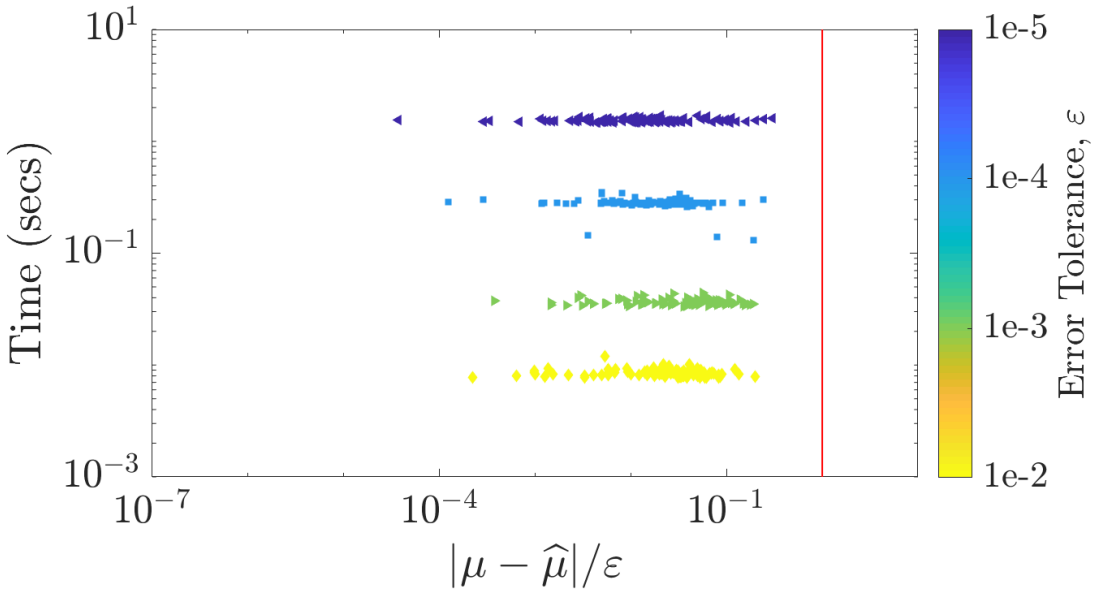

Figure 7.4. `cubBayesNet_g`: Multivariate normal probability example with empirical Bayes stopping criterion.

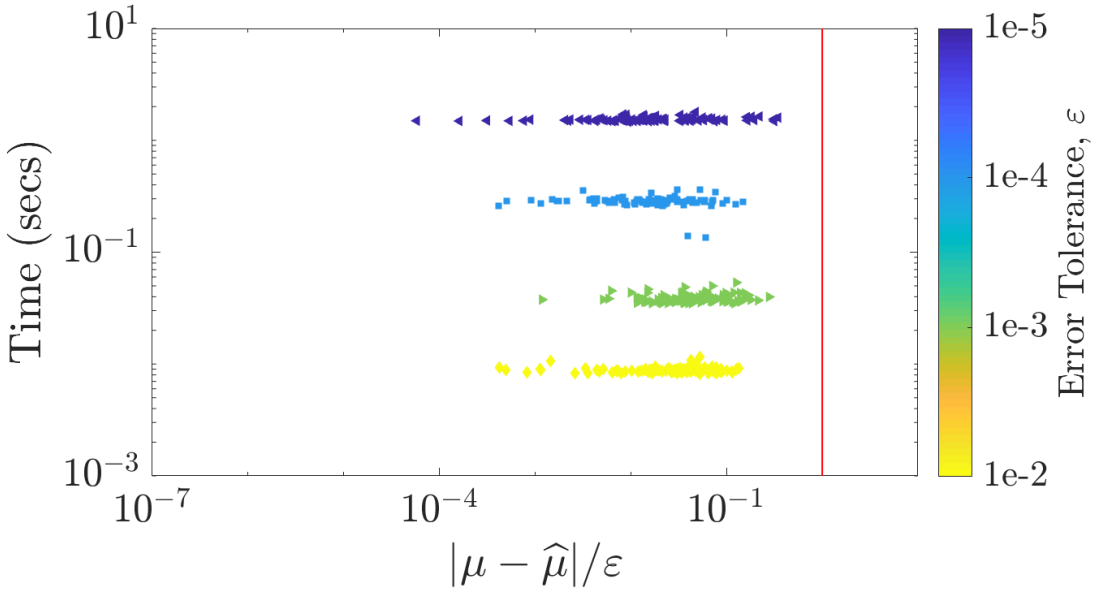

Figure 7.5. `cubBayesNet_g`: Multivariate normal probability example with the full-Bayes stopping criterion.

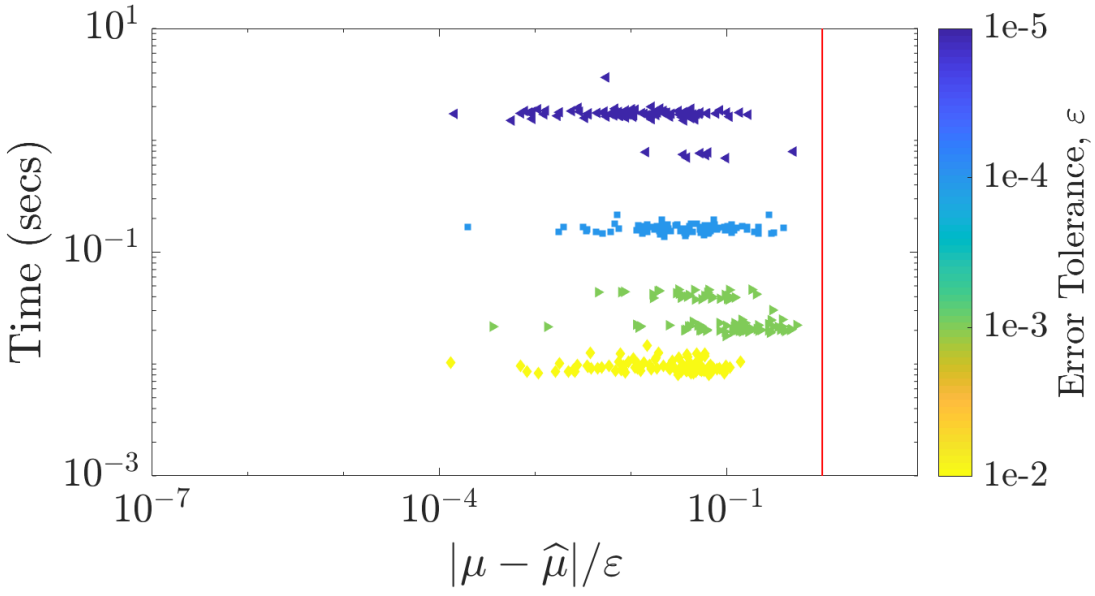

Figure 7.6. `cubBayesNet_g`: Multivariate normal probability example with the GCV stopping criterion.

iteratively in terms of a quadrature as follows:

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

where Γ denotes the gamma function, and

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

7.3.1 Using `cubBayesLattice_g`.

JR: Discuss the big gap between 1e-5 and 1e-4

Figures 7.7, 7.8 and 7.9 summarize the numerical tests for this integral. We used the Sidi's C^1 periodization, dimension $d = 4$, and $r = 2$. As we can see the GCV stopping

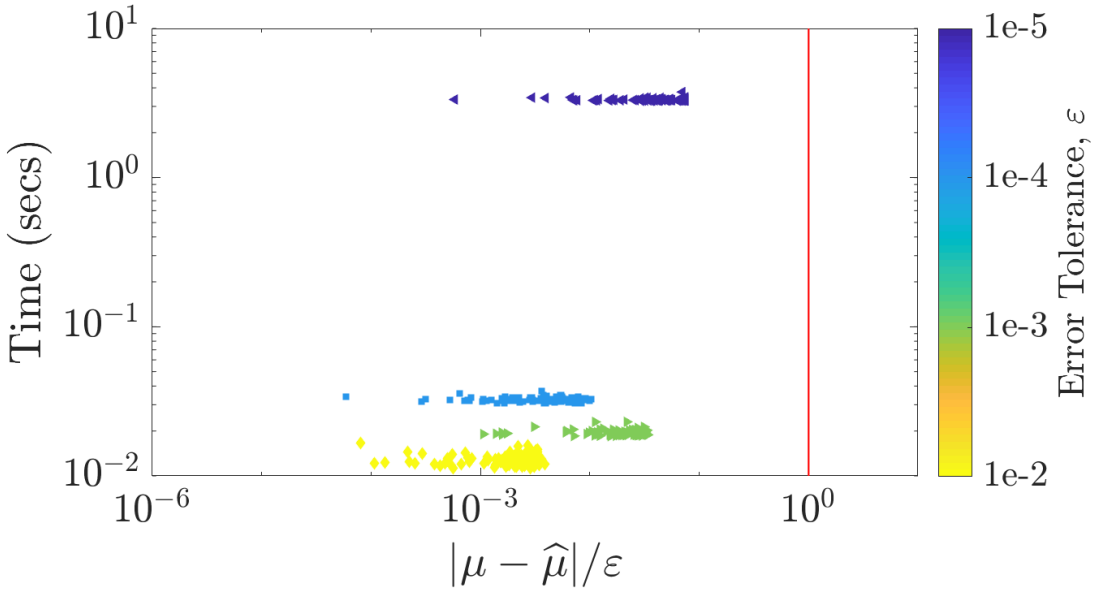

Figure 7.7. `cubBayesLattice_g`: Keister example using the empirical Bayes stopping criterion.

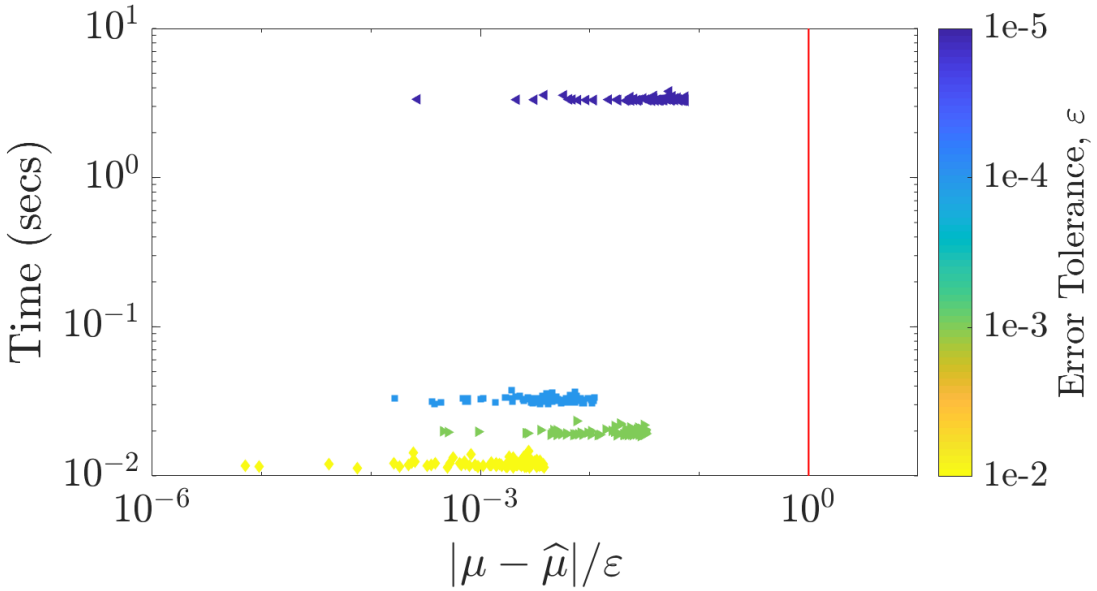

Figure 7.8. `cubBayesLattice_g`: Keister example using the full Bayes stopping criterion.

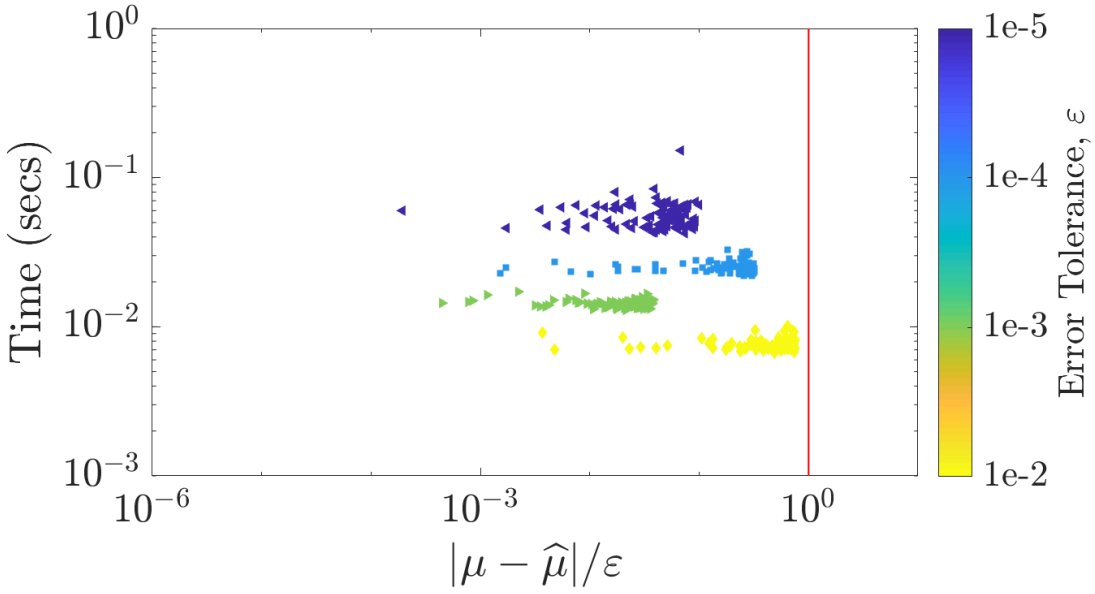

Figure 7.9. `cubBayesLattice_g`: Keister example using the GCV stopping criterion.

criterion achieved the results faster than the others similar to the multivariate normal case.

7.3.2 Using `cubBayesNet_g`. Figures 7.10, 7.11 and 7.12 summarize the numerical tests for this case. We used dimension $d = 4$, and $r = 1$. No periodization transform was used as the integrand need not be periodic. In this example, we use $r = 1$ order kernel where as in Section 7.3.1, $r = 2$ kernel was used, which necessitates the `cubBayesNet_g` to use more samples for integration. As we can see the GCV stopping criterion achieved the results faster than the others similar to the multivariate normal case.

7.4 Option Pricing

The price of financial derivatives can often be modeled by high dimensional integrals. If the underlying asset is described in terms of a discretized geometric

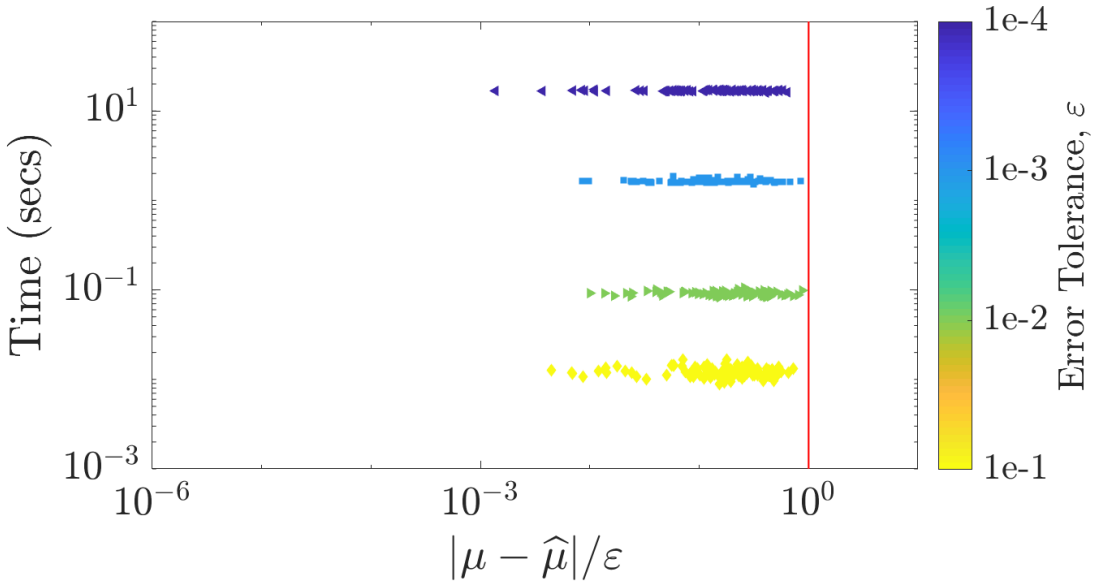

Figure 7.10. `cubBayesNet_g`: Keister example using the empirical Bayes stopping criterion.

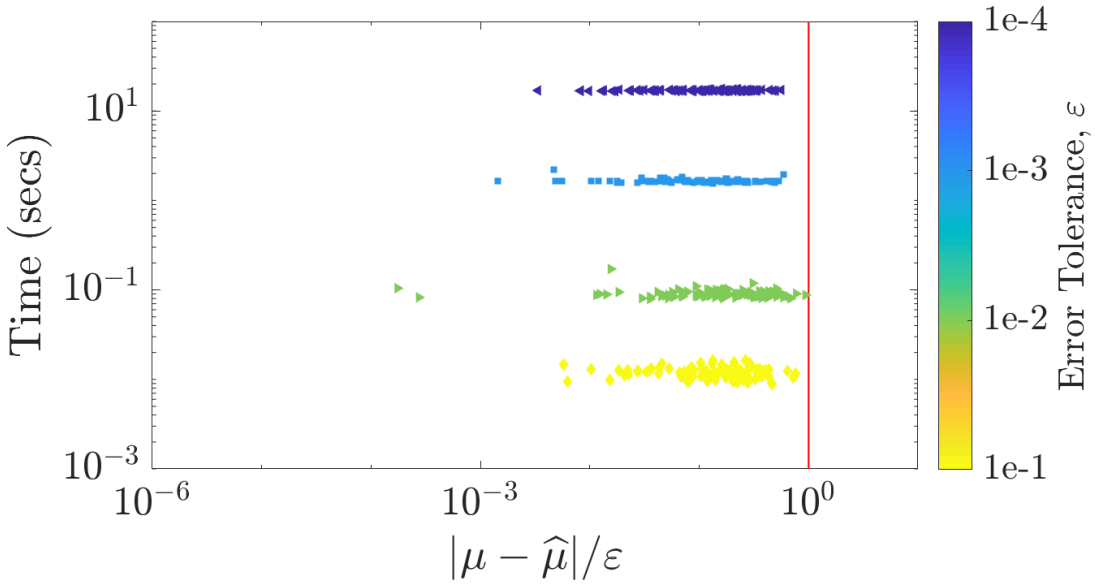

Figure 7.11. `cubBayesNet_g`: Keister example using the full-Bayes stopping criterion.

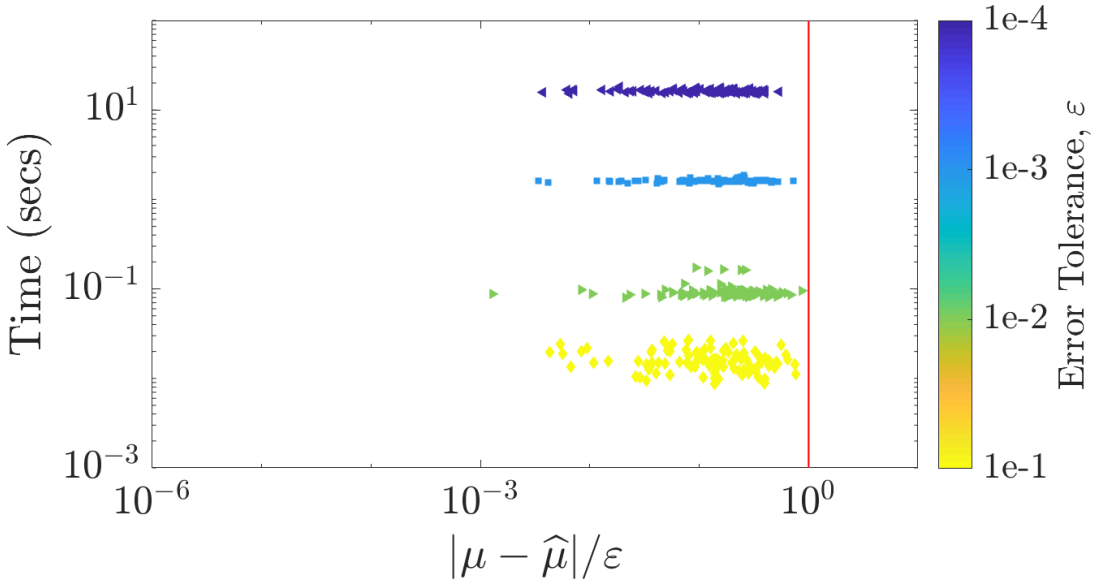

Figure 7.12. `cubBayesNet_g`: Keister example using the GCV stopping criterion.

Brownian motion, then the fair price of the option is:

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

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

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

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

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

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

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

Here, T denotes the time to maturity of the option, d the number of time steps, S_0

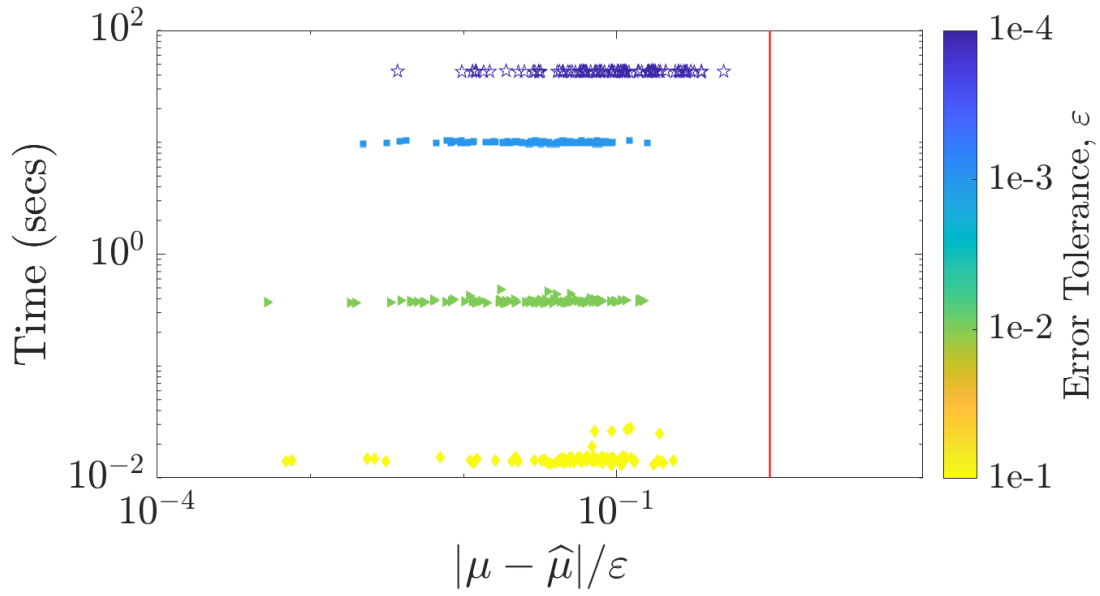

Figure 7.13. `cubBayesLattice_g`: Option pricing using the empirical Bayes stopping criterion.

the initial price of the stock, r the interest rate, σ the volatility, and K the strike price.

7.4.1 Using `cubBayesLattice_g`. The Figures 7.13, 7.14 and 7.15 summarize the numerical results for this example using $T = 1/4$, $d = 13$, $S_0 = 100$, $r = 0.05$, $\sigma = 0.5$, $K = 100$. Moreover, L is chosen to be the matrix of eigenvectors of $\mathsf{\Sigma}$ times the square root of the diagonal matrix of eigenvalues of $\mathsf{\Sigma}$. Because the integrand has a kink caused by the max function, it does not help to use a periodizing transform that is very smooth. We choose the baker's transform (4.11) and $r = 1$.

7.4.2 Using `cubBayesNet_g`. The Figures 7.16, 7.17 and 7.18 summarize the numerical results for the option pricing example using the same values as in Section 7.4.1, $T = 1/4$, $d = 13$, $S_0 = 100$, $r = 0.05$, $\sigma = 0.5$, $K = 100$. As mentioned before, this integrand has a kink caused by the max function, So, `cubBayesNet_g` could be more efficient than `cubBayesLattice_g`, as no periodization transform is required. This can be observed from the number of samples used for intgration to meet the

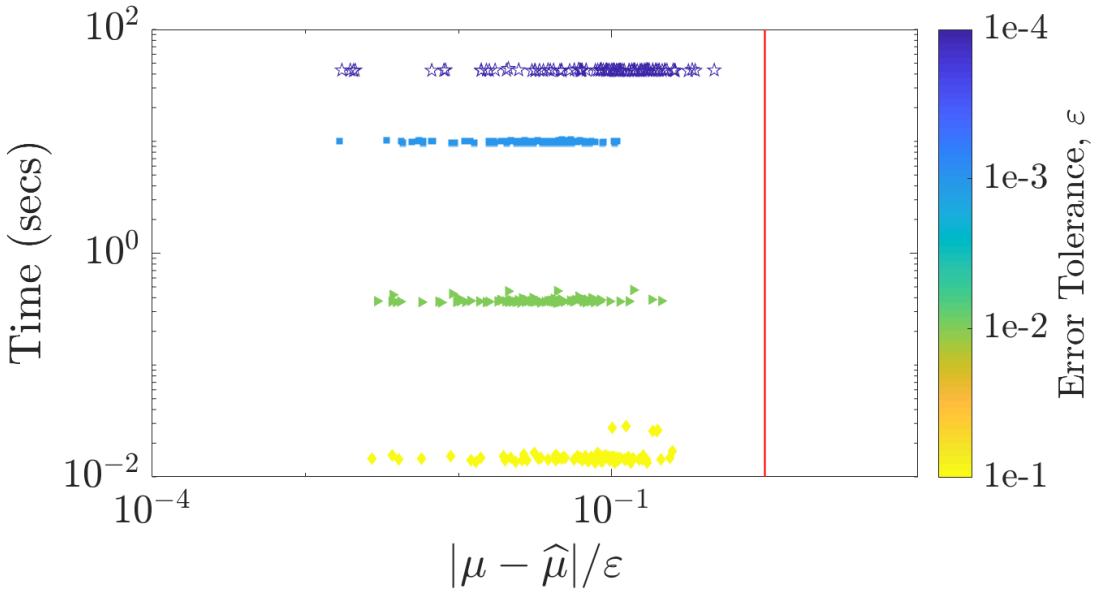

Figure 7.14. `cubBayesLattice_g`: Option pricing using the full Bayes stopping criterion.

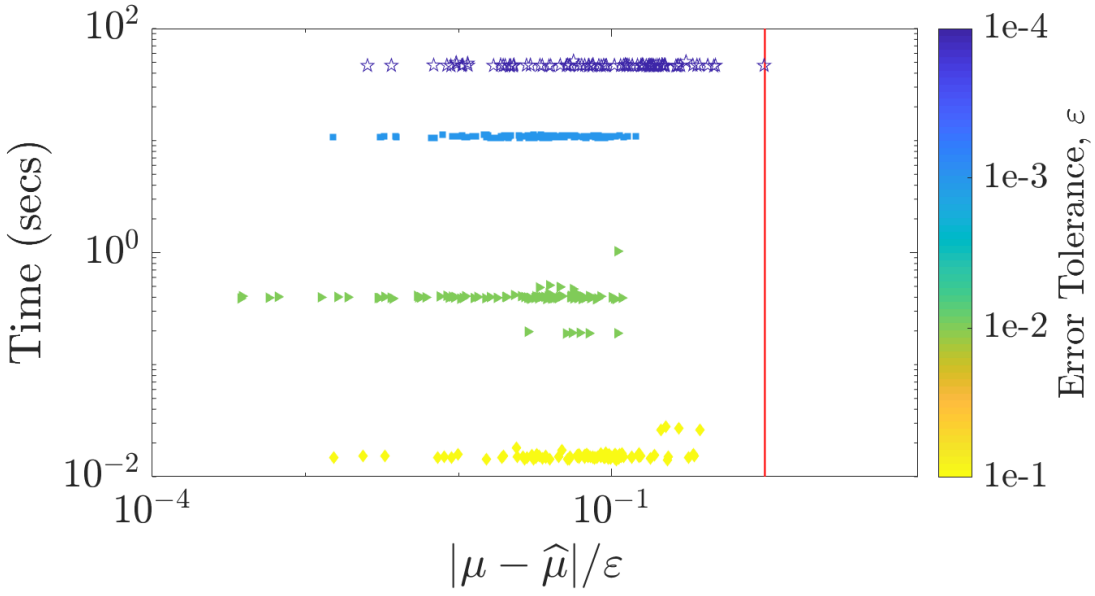

Figure 7.15. `cubBayesLattice_g`: Option pricing using the GCV stopping criterion.

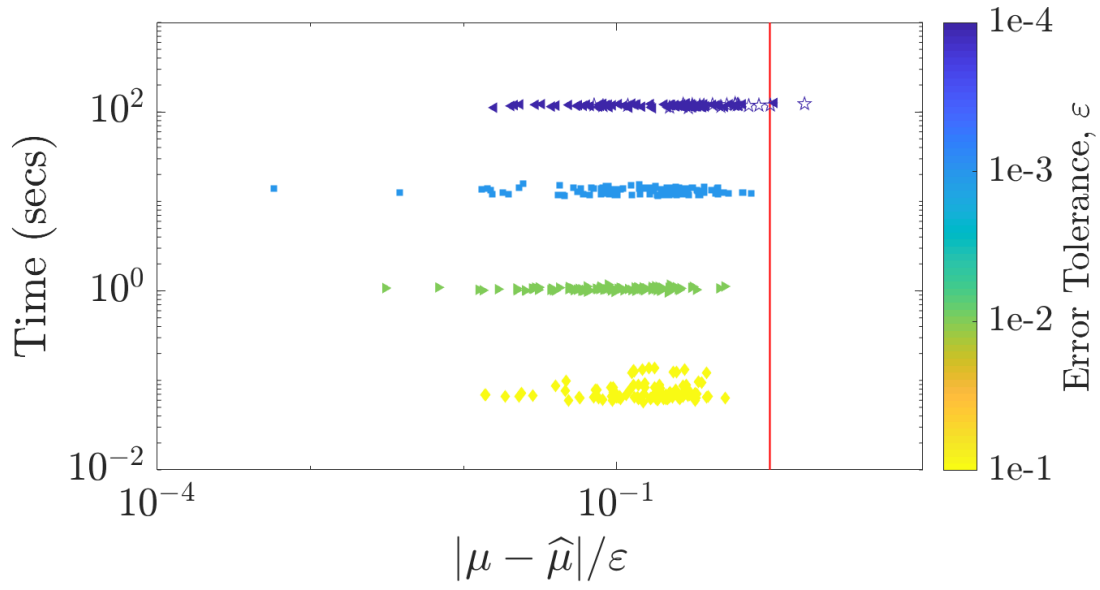

Figure 7.16. `cubBayesNet_g`: Option pricing using the empirical Bayes stopping criterion.

same error threshold. For the error tolerance $\varepsilon = 10^{-3}$, the `cubBayesLattice_g` used $n = 2^{20}$ samples, whereas the `cubBayesNet_g` used $n = 2^{17}$ samples.

7.5 Discussion

As shown in Figures 7.1 to 7.18, both the algorithms computed integral within user specified threshold most of the times except on a few occasions. This is especially the case with option pricing example due to the complexity and high dimension of the integrand. Also notice the `cubBayesLattice_g` algorithm finished within 10seconds of time for Keister and multivariate Normal. For the option pricing it look closer to 70seconds. This is again due to the complexity of the integrand.

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

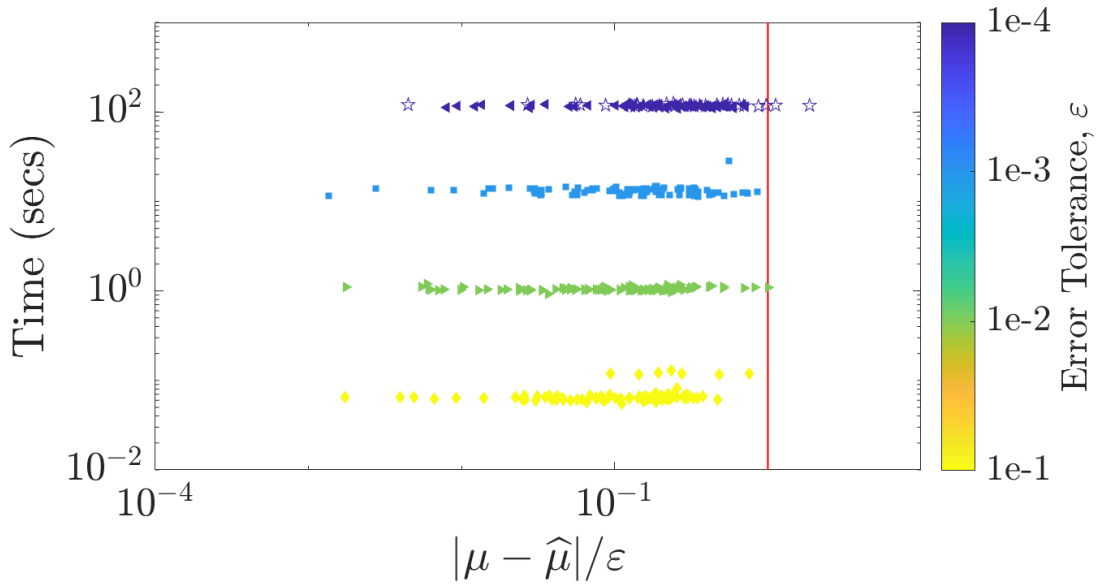

Figure 7.17. `cubBayesNet_g`: Option pricing using the full-Bayes stopping criterion.

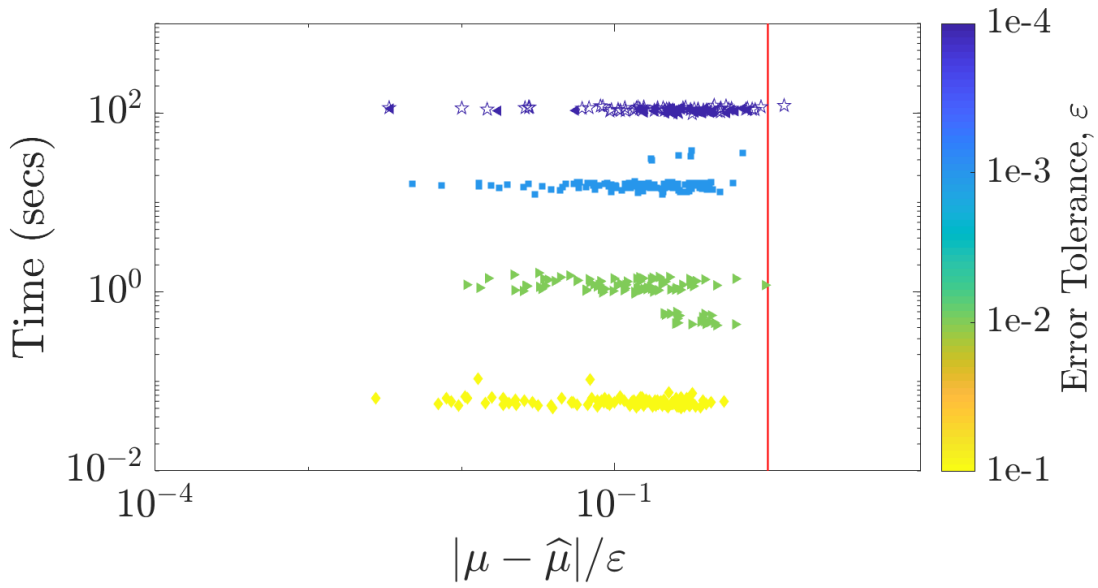

Figure 7.18. `cubBayesNet_g`: Option pricing using the GCV stopping criterion.

Most noticeable aspect from the plots of `cubBayesNet_g` is how closer the error bounds are to the true error. This shows the `cubBayesNet_g`'s estimation of expected error in the stopping criterion is very accurate. Similar to `cubBayesLattice_g`, it missed to meet the given error threshold for the option pricing example, as marked by the hollow stars, for $\varepsilon = 10^{-4}$. This is because the complexity of the integrand and the algorithm reached max allowed number of samples $n = 2^{20}$.

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

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

7.6.1 Keister integral. The Table 7.1 summarizes the performance of the methods MC, Lattice, Sobol, BayesLat, and BayesSob—they refer to the GAIL cubatures, `cubMC_g`, `cubLattice_g`, `cubSobol_g`, `cubBayesLattice_g`, `cubBayesNet_g`, respectively for estimating Keister integral defined in (7.1). We conducted two simulations with $d = 3, 8$. In the case of $d = 3$, all five methods succeeded completely meaning the absolute error is less than given tolerance, i.e., $|\mu - \hat{\mu}| \leq \varepsilon$, where $\hat{\mu}$ is a cubature's approximated value. The fastest method was `cubBayesLattice_g`. In the case of $d = 8$, `cubSobol_g` achieved 100% success rate and was the fastest. But `cubBayesLattice_g` was competitive and had the smallest average absolute error. `cubBayesNet_g` used lowest number of samples in case of $d = 8$ but slower than `cubSobol_g`.

Table 7.1. Comparison of average performance of cubatures for estimating the integral (7.1) for 1000 independent runs. These results can be conditionally reproduced with the script, `KeisterCubatureExampleBayes.m`, in GAIL.

$d = 3,\ \varepsilon = 0.005$

Method	MC	Lattice	Sobol	BayesLat	BayesSobol
Absolute Error	1.10×10^{-3}	5.10×10^{-4}	5.20×10^{-4}	4.30×10^{-4}	5.60×10^{-4}
Tolerance Met	100%	100%	100%	100%	100%
n	2 500 000	4100	3900	1000	1900
Time (seconds)	0.1800	0.0069	0.0054	0.0029	0.0700

$d = 8,\ \varepsilon = 0.050$

Method	MC	Lattice	Sobol	BayesLat	BayesSobol
Absolute Error	1.20×10^{-2}	1.50×10^{-2}	7.30×10^{-3}	1.80×10^{-3}	8.30×10^{-3}
Tolerance Met	100%	99%	100%	100%	100%
n	7 400 000	15 000	16 000	66 000	8200
Time (seconds)	1.2000	0.0220	0.0160	0.2100	0.3500

7.6.2 Multi-variate Normal

The Table 7.2 summarizes the performance of the methods MC, Lattice, Sobol, BayesLat, and BayesSob for estimating the multi-dimensional Normal probability $\mathbf{X} \sim \mathbf{N}(\mu, \mathbf{\Sigma})$. This experiment demonstrates our algorithm's ability to handle very high-dimensional integral.

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

Table 7.2. Comparison of average performance of cubatures for estimating the $d = 20$ Multi-variate Normal (2.21) for 1000 independent runs with $\varepsilon = 10^{-3}$. These results can be conditionally reproduced with the script, `MVNCubatureExampleBayes.m`, in GAIL.

$$\Sigma = \mathbf{I_d},\ \mathbf{b} = -\mathbf{a} = (\mathbf{3.5}, \ldots, \mathbf{3.5})$$

Method	MC	Lattice	Sobol	BayesLat	BayesSobol
Absolute Error	2.20×10^{-16}	2.70×10^{-14}	2.70×10^{-14}	2.20×10^{-16}	2.20×10^{-16}
Tolerance Met	100%	100%	100%	100%	100%
n	10 000	1000	1000	1000	260
Time (seconds)	0.0410	0.0820	0.0710	0.0650	0.0790

$$\Sigma = 0.4\mathbf{I_d} + \mathbf{0.611^T},\ \mathbf{a} = (-\infty, \ldots, -\infty),\ \mathbf{b} = \sqrt{\mathbf{d}}(\mathbf{U_1}, \ldots, \mathbf{U_d})$$

Method	MC	Lattice	Sobol	BayesLat	BayesSobol
Absolute Error	2.30×10^{-4}	2.10×10^{-4}	4.40×10^{-4}	1.00×10^{-4}	4.80×10^{-5}
Tolerance Met	100%	100%	100%	100%	100%
n	10 000	1000	1000	1000	260
Time (seconds)	0.0350	0.0120	0.0140	0.0150	0.0300

7.7 Shape Parameter Fine-tuning

JR: Numerical examples for the case of shape parameter per dimension

TBD

CHAPTER 8

DISCUSSION AND FUTURE WORK

We have developed a fast, automatic Bayesian cubature that estimates the high dimensional integral within a user defined error tolerance that occur in many scientific computing such as finance, machine learning, or imaging, etc. The stopping criteria arise from assuming the integrand to be a Gaussian process. In Section 2.2, we developed three versions: empirical Bayes, full Bayes, and generalized cross-validation. Empirical-Bayes uses maximum-likelihood to optimally choose the parameters, where posterior of the parameters given the integrand values is maximized. Alternatively, full-Bayes assumes non-informative prior on the parameters and then computes posterior distribution of the integral μ, which leads to a t-distribution to obtain the parameters. Generalized cross-validation extends the concept of cross-validation to construct an objective which in turn maximized.

The computational cost of the automatic Bayesian cubature can be dramatically reduced if the covariance kernel matches the nodes. We have demonstrated two such matches in practice. The first algorithm was based on rank-1 lattice nodes and shift-invariant kernels where the matrix-vector multiplications can be accomplished using the fast Fourier Transform. The second algorithm was based on Sobol points with first order Walsh kernel where the matrix-vector multiplications can be accomplished using the fast Walsh transform. Three integration problems illustrate the performance of our automatic Bayesian cubature.

For faster computations one could use fixed order kernels in `cubBayesLattice_g`, but for more advanced usage, we have added a kernel variation in Section 4.3 that allows one to optimally choose the kernel order without the constraint of being even integer.

During the numerical experiments we noticed a computation step that causes inaccuracy due to a cancellation error in the estimation of stopping criterion. We have developed a novel technique in Section 6.1, to overcome this cancellation error using the inherent structure of the shift invariant kernel used in our algorithm.

In Section 3.5.1, we have analytically computed the gradient of the objective function and the shift invariant kernel to use with steepest descent in kernel parameters search. Quasi-Monte Carlo cubature methods are efficient [32] even if the dimension is high given the effective dimension is low. To take advantage of low effective dimension, one should not fix the kernel shape parameter across all the dimensions. In this situation steepest descent method come in handy as one search parameters in multi-dimensions.

8.1 Future work

We demonstrated the capability our new Bayesian cubature algorithms to successfully compute the integrals faster within the user defined error tolerances. But there are few potential improvements and new areas of applications. Some of the improvement ideas are listed here:

- Higher order digital sequences and digital shift and/or scramble invariant kernels [25] [18]: We could improve the computation speed of `cubBayesNet_g` for smoother integrands using higher order digital sequences and matching kernels which have the potential of being another match that satisfies the conditions in Section 3. The fast transform would correspond to a fast Walsh transform similar to the second algorithm we demonstrated. For such kernels and the first order Walsh kernel we demonstrated, periodicity is not assumed, however, special structure of both the sequences and the kernels are required to take advantage of integrand smoothness.

- Control variates: One should be able to adapt our Bayesian cubature to control variates, i.e., assuming

$$f = \mathcal{GP}\left(\beta_0 + \beta_1 g_1 + \cdots + \beta_p g_p, s^2 C\right),$$

  for some choice of $g_1, \ldots, g_p$ whose integrals are known, and some parameters $\beta_0, \ldots, \beta_p$ in addition to the s and C. The efficacy of this approach has not yet been explored.

- Steepest descent: The kernels's optimal shape parameter searched using steepest descent with kernels gradient could sometime get into local minima. This needs more understanding and enhancements.

- Gaussian diagnosis: We assumed the integrand an instance of a Gaussian process. Is there a way to prove that is a good assumption based on the results we have?

BIBLIOGRAPHY

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

[2] F-X Briol, C. J. Oates, M. Griolami, M. A. Osborne, and D. Sejdinovic. Probabilistic integration: A role in statistical computation? *Statist. Sci.*, 2018+. to appear.

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

[4] R. Cools and D. Nuyens, editors. *Monte Carlo and Quasi-Monte Carlo Methods: MCQMC, Leuven, Belgium, April 2014*, volume 163 of *Springer Proceedings in Mathematics and Statistics*. Springer-Verlag, Berlin, 2016.

[5] P. Craven and G. Wahba. Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numer. Math.*, 31:307–403, 1979.

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

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

[8] J. Dick, F. Kuo, and I. H. Sloan. High dimensional integration — the Quasi-Monte Carlo way. *Acta Numer.*, 22:133–288, 2013.

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

[10] A. Genz. Comparison of methods for the computation of multivariate normal probabilities. *Computing Science and Statistics*, 25:400–405, 1993.

[11] G. H. Golub, M. Heath, and G. Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21:215–223, 1979.

[12] F. J. Hickernell. Quadrature error bounds with applications to lattice rules. *SIAM J. Numer. Anal.*, 33:1995–2016, 1996. corrected printing of Sections 3-6 in ibid., **34** (1997), 853–866.

[13] F. J. Hickernell. The trio identity for quasi-Monte Carlo error analysis. In P. Glynn and A. Owen, editors, *Monte Carlo and Quasi-Monte Carlo Methods: MCQMC, Stanford, USA, August 2016*, Springer Proceedings in Mathematics and Statistics, pages 13–37. Springer-Verlag, Berlin, 2018. arXiv:1702.01487.

[14] F. J. Hickernelland Ll. A. Jiménez Rugama. Reliable adaptive cubature using digital sequences. In Cools and Nuyens [4], pages 367–383. arXiv:1410.8615 [math.NA].

[15] F. J. Hickernell, Ll. A. Jiménez Rugama, and D. Li. Adaptive quasi-Monte Carlo methods for cubature. In J. Dick, F. Y. Kuo, and H. Woźniakowski, editors, *Contemporary Computational Mathematics — a celebration of the 80th birthday of Ian Sloan*, pages 597–619. Springer-Verlag, 2018.

[16] F. J. Hickernelland H. Niederreiter. The existence of good extensible rank-1 lattices. *J. Complexity*, 19:286–300, 2003.

[17] Nicholas J. Higham. *Functions of matrices: theory and computation*. SIAM, 2008.

[18] G. Leobacher D. Nuyens F. Pillichshammer J. Baldeaux, J. Dick. Efficient calculation of the worst-case error and (fast) component-by-component construction of higher order polynomial lattice rules. *Numerical Algorithms*, 59:403–431, March 2012.

[19] Ll. A. Jiménez Rugama and F. J. Hickernell. Adaptive multidimensional integration based on rank-1 lattices. In Cools and Nuyens [4], pages 407–422. arXiv:1411.1966.

[20] B. D. Keister. Multidimensional quadrature algorithms. *Computers in Physics*, 10:119–122, 1996.

[21] Frances Y. Kuo and Dirk Nuyens. Application of quasi-monte carlo methods to elliptic pdes with random diffusion coefficients a survey of analysis and implementation. *Foundations of Computational Mathematics*, 16(6):1631–1696, 2016.

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

[23] D. Nuyens.

[24] D. Nuyens.

[25] Dirk Nuyens. The construction of good lattice rules and polynomial lattice rules. 08 2013.

[26] A. O'Hagan. Bayes-Hermite quadrature. *J. Statist. Plann. Inference*, 29:245–260, 1991.

[27] F. W. J. Olver, D. W. Lozier, R. F. Boisvert, C. W. Clark, and A. B. O. Dalhuis. Digital library of mathematical functions, 2018.

[28] C. E. Rasmussen and Z. Ghahramani. Bayesian Monte Carlo. In S. Thrun, L. K. Saul, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 489–496. MIT Press, 2003.

[29] C. E. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, Massachusetts, 2006. (online version at `http://www.gaussianprocess.org/gpml/`).

[30] K. Ritter. *Average-Case Analysis of Numerical Problems*, volume 1733 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 2000.

[31] A. Sidi. Further extension of a class of periodizing variable transformations for numerical integration. *J. Comput. Appl. Math.*, 221:132–149, 2008.

[32] I. H. Sloan and H. Woźniakowski. When are quasi-Monte Carlo algorithms efficient for high dimensional integrals? *J. Complexity*, 14:1–33, 1998.

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

[34] G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, 1990.