Chapter 1

Fast Automatic Bayesian Cubature Using Sobol' Sampling

R. Jagadeeswaran and Fred J. Hickernell

Abstract Automatic cubatures approximate integrals to user-specified error tolerances. For high dimensional problems, it is difficult to adaptively change the sampling pattern to focus on peaks because peaks can hide more easily in high dimensional space. But, one can automatically determine the sample size, n, given a reasonable, fixed sampling pattern. This approach is pursued in Jagdeeswaran and Hickernell, Stat. Comput., 29:1214-1229, 2019, where a Bayesian perspective is used to construct a credible interval for the integral, and the computation is terminated when the half-width of the interval is no greater than the required error tolerance. Our earlier work employs integration lattice sampling, and the computations are expedited by the fast Fourier transform because the covariance kernels for the Gaussian process prior on the integrand are chosen to be shift-invariant. In this chapter, we extend our fast automatic Bayesian cubature to digital net sampling via digital shift-invariant covariance kernels and fast Walsh transforms. Our algorithm is implemented in the MATLAB Guaranteed Automatic Integration Library (GAIL) and QMCPy Python library.

R. Jagadeeswaran
Department of Applied Mathematics,
Illinois Institute of Technology
10 W. 32nd St., Room 220, Chicago IL 60616
e-mail: jrathin1@iit.edu

Fred J. Hickernell
Center for Interdisciplinary Scientific Computation and
Department of Applied Mathematics
Illinois Institute of Technology
10 W. 32nd St., Room 220, Chicago IL 60616
e-mail: hickernell@iit.edu

1.1 Introduction

Cubature, or numerical multivariate integration, is the problem of inferring a numerical value for a definite integral,

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

when no closed-form analytic expression exists. Typically, values of f are accessible through a black-box function routine. Our goal is to construct a cubature, $\widehat{\mu}_n = \widehat{\mu}_n(f)$, depending only on integrand values at the nodes $\{\boldsymbol{x}_i\}_{i=1}^n$, and determine the n that satisfies the error criterion

$$|\mu - \widehat{\mu}_n| \le \varepsilon \tag{1.2}$$

with high probability. This article extends the fast Bayesian cubature ideas presented in [11] to digital sequences [7].

Cubature is a key component of many problems in scientific computing, finance [9], statistical modeling, imaging [13], uncertainty quantification, and machine learning [10]. The original form of the integral may require a suitable variable transformation to become (1.1). This process is addressed in [1, 5, 14, 21, 22].

Following the Bayesian numerics approach of [2, 6, 18, 19] and others, we assume that our integrand is an instance of a Gaussian process, $\mathcal{GP}(m, s^2 C_{\boldsymbol{\theta}})$, and construct a probabilistic error bound for μ via a Bayesian credible interval. The integrand is sampled until the credible interval becomes small enough to satisfy (1.2) with high probability.

Our approach to fast Bayesian cubature [11] emphasizes two key points

i) Choosing covariance kernels, $C_{\boldsymbol{\theta}} : [0,1]^d \times [0,1]^d \to \mathbb{R}$, for which the symmetric, positive definite Gram matrices,

$$\mathsf{C}_{\boldsymbol{\theta}} = \left(C_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)_{i,j=1}^n, \tag{1.3}$$

have an eigenvalue-eigenvector decomposition of the form[1] $\mathsf{C}_{\boldsymbol{\theta}} = \mathsf{V}\mathsf{\Lambda}_{\boldsymbol{\theta}}\mathsf{V}^H/n$, where

$$\mathsf{V} \text{ may be identified analytically,} \tag{1.4a}$$

$$\text{the first row and column of } \mathsf{V} \text{ are } \boldsymbol{1}, \tag{1.4b}$$

$$\text{Computing } \mathsf{V}^H\boldsymbol{b} \text{ requires only } \mathcal{O}(n \log n) \text{ operations } \forall \boldsymbol{b}, \text{ and} \tag{1.4c}$$

$$\int_{[0,1]^d} C_{\boldsymbol{\theta}}(\boldsymbol{t}, \boldsymbol{x}) \, \mathrm{d}\boldsymbol{t} = 1 \qquad \forall \boldsymbol{x} \in [0,1]^d, \tag{1.4d}$$

and

ii) The hyperparameters of the Gaussian process, m, s, and $\boldsymbol{\theta}$ are tuned to increase the likelihood that f is a typical integrand and not an outlier.

[1] The presence of $1/n$ in the eigenvalue-eigenvector decomposition arises from the assumption that the first column of V is $\boldsymbol{1}$. It could be removed by assuming that the first column of V is $\boldsymbol{1}/\sqrt{n}$.

We call the transformation $\boldsymbol{b} \mapsto \mathsf{V}^H\boldsymbol{b}$ a *fast Bayesian transform*. Our earlier work [11] focuses on lattice nodes and shift-invariant kernels. In that context the computation of $\mathsf{V}^H\boldsymbol{b}$ is a one-dimensional fast Fourier transform. This chapter focuses on digital sequences, such as Sobol' sequences [23], and covariance kernels that are digital shift-invariant. In this case the computation of $\mathsf{V}^H\boldsymbol{b}$ is a fast Walsh-Hadamard transform.

The next section summarizes the key formulae from [11]. Section 1.3 extends fast Bayesian cubature to digital nets and digital-shift invariant kernels defined in terms of Walsh functions. Section 1.4 presents numerical experiments that illustrate these ideas.

1.2 Bayesian Cubature

We assume the integrand, f, is an instance of a real-valued stochastic Gaussian process, i.e., $f \sim \mathcal{GP}(m, s^2 C_{\boldsymbol{\theta}})$. Specifically, the random function f has constant mean, m, and covariance kernel $s^2 C_{\boldsymbol{\theta}}$, where s is a positive scale factor, and $C_{\boldsymbol{\theta}} : [0,1]^d \times [0,1]^d \to \mathbb{R}$ is a symmetric, positive-definite kernel parameterized by $\boldsymbol{\theta}$. The parameter $\boldsymbol{\theta}$ may affect the shape or smoothness of $C_{\boldsymbol{\theta}}$. In [11] we introduced three methods for resolving the hyperparameters: empirical Bayes (EB), full Bayes (full), and generalized cross-validation (GCV). Under assumptions (1.4) for the covariance kernel and sampling sites, it was shown in [11, Theorem 2] that under these three approaches the credible interval for the integral takes the form

$$\mathbb{P}_f\left[|\mu - \widehat{\mu}_n| \le \mathrm{err}_{\mathrm{CI}}\right] = 99\%, \tag{1.5}$$

where

$$\begin{aligned}
\widehat{\mu}_n &= \frac{1}{n}\sum_{i=1}^n y_i = \widetilde{y}_1, && (1.6)\\
\widetilde{\boldsymbol{y}} &:= \mathsf{V}^H \boldsymbol{y},\\
s^2_{\mathrm{EB}} &= \frac{1}{n^2}\sum_{i=2}^n \frac{|\widetilde{y}_i|^2}{\lambda_{\boldsymbol{\theta},i}},\\
s^2_{\mathrm{GCV}} &= \frac{1}{n}\sum_{i=2}^n \frac{|\widetilde{y}_i|^2}{\lambda^2_{\boldsymbol{\theta},i}}\left[\sum_{i=1}^n \frac{1}{\lambda_{\boldsymbol{\theta},i}}\right]^{-1},\\
\widehat{\sigma}^2_{\mathrm{full}} &= \frac{1}{n(n-1)}\sum_{i=2}^n \frac{|\widetilde{y}_i|^2}{\lambda_{\boldsymbol{\theta},i}}\left(\frac{\lambda_{\boldsymbol{\theta},1}}{n} - 1\right),
\end{aligned}$$

$$\boldsymbol{\theta}_{\mathrm{EB}} = \underset{\boldsymbol{\theta}}{\mathrm{argmin}}\left[\log\left(\sum_{i=2}^n \frac{|\widetilde{y}_i|^2}{\lambda_{\boldsymbol{\theta},i}}\right) + \frac{1}{n}\sum_{i=1}^n \log(\lambda_{\boldsymbol{\theta},i})\right], \tag{1.7a}$$

$$\boldsymbol{\theta}_{\mathrm{GCV}} = \underset{\boldsymbol{\theta}}{\mathrm{argmin}}\left[\log\left(\sum_{i=2}^n \frac{|\widetilde{y}_i|^2}{\lambda^2_{\boldsymbol{\theta},i}}\right) - 2\log\left(\sum_{i=1}^n \frac{1}{\lambda_{\boldsymbol{\theta},i}}\right)\right]. \tag{1.7b}$$

$$\mathrm{err}_{\mathrm{CI},\mathsf{x}} = \mathrm{err}_{\mathsf{x}} = 2.58 s_{\mathsf{x}} \sqrt{1 - \frac{n}{\lambda_{\boldsymbol{\theta},1}}}, \qquad \mathsf{x} \in \{\mathrm{EB}, \mathrm{GCV}\}, \tag{1.8a}$$

$$\mathrm{err}_{\mathrm{CI,full}} = t_{n-1,0.995}\widehat{\sigma}_{\mathrm{full}} > \mathrm{err}_{\mathrm{EB}}. \tag{1.8b}$$

In the formulas for the credible interval half-widths, $\boldsymbol{\theta}$ is assumed to take on the values $\boldsymbol{\theta}_{\mathrm{EB}}$ or $\boldsymbol{\theta}_{\mathrm{GCV}}$ as appropriate. There is no suitable $\boldsymbol{\theta}_{\mathrm{full}}$.

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; these steps are detailed in Algorithm 1.

Algorithm 1 Automatic Bayesian Cubature

```
Require: a generator for the sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$; a black-box function, $f$; an absolute error tolerance, $\varepsilon > 0$; the positive initial sample size, $n_0$; the maximum sample size $n_{\max}$
$n \leftarrow n_0$, $n' \leftarrow 0$, $\mathrm{err}_{\mathrm{CI}} \leftarrow \infty$
while $\mathrm{err}_{\mathrm{CI}} > \varepsilon$ and $n \le n_{\max}$ do
    Generate $\{\boldsymbol{x}_i\}_{i=n'+1}^{n}$ and sample $\{f(\boldsymbol{x}_i)\}_{i=n'+1}^{n}$
    Compute $\boldsymbol{\theta}$ by (1.7a) or (1.7b)
    Compute $\mathrm{err}_{\mathrm{CI}}$ according to (1.8a) or (1.8b)
    $n' \leftarrow n$, $n \leftarrow 2n'$
end while
Sample size to compute $\widehat{\mu}_n$, $n \leftarrow n'$
Compute $\widehat{\mu}_n$, the approximate integral, according to (1.6)
return $\widehat{\mu}_n$, $n$ and $\mathrm{err}_{\mathrm{CI}}$
```

We recognize that multiple applications of our credible intervals in one run of the algorithm is not strictly justified. However, if our integrand comes from the middle of the sample space and not the extremes, we expect our automatic Bayesian cubature to approximate the integral within the desired error tolerance with high probability. The examples in Section 1.4 support that expectation.

The credible intervals in our automatic algorithm are homogeneous with respect to the function data. If they are valid for some integrand, f, they are also valid for the integrand af for any constant a.

1.3 Digital Nets and Walsh Kernels

The previous section does not mention which sequences of data sites and kernels satisfy assumptions (1.4). We demonstrated in [11] that rank-1 lattice points and shift-invariant kernels do so. In this section, we give another example, namely digital sequences nets and Walsh kernels The results of this chapter can be summarized as follows:

Theorem 1 *Any symmetric, positive definite, digital shift-invariant kernel of the form* (1.14) *scaled to satisfy* (1.4d)*, when matched with digital sequence data sites, satisfies assumptions* (1.4)*. The* fast Walsh-Hadamard transform *performs the fast Bayesian transform, $\boldsymbol{b} \mapsto \mathsf{V}^H \boldsymbol{b}$, in $\mathcal{O}(n \log n)$ operations.*

This section defines digital sequences, digital shift-invariant kernels, and the fast Walsh-Hadamard transform.

1.3.1 Digital Sequences

The first example of a digital sequence was proposed by Sobol' [23], and his Sobol' sequences are the most popular. These were later generalized. In this chapter we specialize to the case of base 2, which includes Sobol' sequences. Here is a definition of a digital sequence in base 2.

Definition 1 For any non-negative integer $i = (\dots i_3 i_2 i_1)_2$, define $\overrightarrow{i} = (i_1, i_2, \dots)^T$ as the $\infty \times 1$ vector $\overrightarrow{i}$ of its binary digits. For any point $z = {}_2 0.z_1 z_2 \cdots \in [0, 1)$, define $\overrightarrow{z} = (z_1, z_2, \dots)^T$ as the $\infty \times 1$ vector of its binary digits. Let $\mathsf{G}_1, \dots, \mathsf{G}_d$ denote predetermined $\infty \times \infty$ generator matrices whose elements are zeros and ones. A digital sequence in base 2 is $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3, \dots\}$, where each $\boldsymbol{z}_i = (z_{i1}, \dots, z_{id})^T \in [0, 1)^d$ is defined by

$$\overrightarrow{z}_{i+1,\ell} = \mathsf{G}_\ell \overrightarrow{i} \pmod 2, \quad \ell = 1, \dots, d, \quad i = 0, 1, \dots .$$

It is common to index digital sequences starting with 0, whereas our data sites and matrices in the earlier section are indexed starting with 1. To keep our notation consistent, we define $\boldsymbol{z}_{i+1}$ in terms of $\overrightarrow{i}$.

Digital sequences have a group structure under digitwise, element-by-element addition modulo the base, which we denote by $\oplus$ and which also corresponds to an exclusive-or. Here and in what follows we ignore the cases of measure zero for which the $\oplus$ operation leads to a binary represntation ending in an infinite string of ones. The following lemma summarizes some important properties of digital sequences.

Lemma 1 *Let $\{\boldsymbol{z}_i\}_{i=1}^\infty$ be a digital sequence in base 2 as defined in Definition 1. Choose any digital shift $\boldsymbol{\Delta} \in [0, 1)^d$, and define $\{\boldsymbol{x}_i\}_{i=1}^\infty$ by digitwise addition, $\boldsymbol{x}_i = \boldsymbol{z}_i \oplus \boldsymbol{\Delta}$. Then for all $i, j \in \mathbb{N}_0$,*

$$\boldsymbol{x}_{i+1} \oplus \boldsymbol{x}_{i+1} = \boldsymbol{0}, \qquad \boldsymbol{x}_{i+1} \oplus \boldsymbol{x}_{j+1} = \boldsymbol{x}_{j+1} \oplus \boldsymbol{x}_{i+1} \tag{1.9}$$

$$\boldsymbol{x}_{i+1} \oplus \boldsymbol{x}_{j+1} = \boldsymbol{z}_{i+1} \oplus \boldsymbol{z}_{j+1} = \boldsymbol{z}_{(i \oplus j)+1}. \tag{1.10}$$

Therefore, $\{\boldsymbol{z}_i\}_{i=1}^\infty$ is a group. Moreover, $\{\boldsymbol{z}_i\}_{i=1}^{2^m}$ is a subgroup for $m \in \mathbb{N}_0$.

The proof follows straightforwardly from Definition 1 can be found in [20].

Digital sequence generators can be chosen by number theory, as are those of Sobol' [23] and Niederreiter and Xing [16] (see also [7, Chapter 8], or they can be chosen by computer search [7, Chapter 10]. The original generator matrices may be scrambled using linear matrix scrambling [15].

1.3.2 Covariance Kernels Constructed via Walsh Functions

The digital shift invariant kernels required for fast Bayesian cubature using digital nets are constructed via Walsh functions, again specializing to base 2. The one-dimensional Walsh functions in base 2 are defined as

$$\mathrm{wal}_k(x) := (-1)^{k_0 x_1 + k_1 x_2 + \cdots} = (-1)^{\langle \overrightarrow{k}, \overrightarrow{x} \rangle}, \qquad k \in \mathbb{N}_0,\ x \in [0,1), \qquad (1.11)$$

$$\langle \overrightarrow{k}, \overrightarrow{x} \rangle := k_0 x_1 + k_1 x_2 + \cdots .$$

where again $\overrightarrow{k}$ is a vector containing the binary digits of k, and $\overrightarrow{x}$ is a vector containing the binary digits of x. Note that by this definition, $\mathrm{wal}_k(x \oplus t) = \mathrm{wal}_k(x)\mathrm{wal}_k(t)$.

These Walsh functions can be used to construct a covariance kernel for univariate integrands as follows

$$C_{\boldsymbol{\theta}}(x,t) = K_{\boldsymbol{\theta}}(x \oplus t), \qquad K_{\boldsymbol{\theta}}(x) := 1 + \eta \omega_r(x), \qquad \boldsymbol{\theta} = (r, \eta),$$

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

$$\omega_r(x_\ell \oplus t_\ell) = \sum_{k=1}^{\infty} \frac{\mathrm{wal}_k(x)\mathrm{wal}_k(t)}{2^{2r\lfloor \log_2 k \rfloor}}.$$

The symmetric, positive definite property of $C_{\boldsymbol{\theta}}(x,t)$ follows from its definition. This kernel is digital shift invariant because

$$C_{\boldsymbol{\theta}}(x \oplus \Delta, t \oplus \Delta) = K_{\boldsymbol{\theta}}(x \oplus \Delta \oplus t \oplus \Delta) = K_{\boldsymbol{\theta}}(x \oplus t) = C_{\boldsymbol{\theta}}(x,t).$$

This follows because $\Delta \oplus \Delta = 0$.

The parameter r is a measure of the digital smoothness of the kernel, which does not correspond to ordinary smoothness. An explicit expression is available for ω_r in the case of order $r = 1$ [17]:

$$\omega_1(x) = 1 - 6 \times 2^{\lfloor \log_2 x \rfloor - 1}. \qquad (1.12)$$

Figure 1.1 shows $C_{\boldsymbol{\theta}}(x,t)$ for order $r = 1$ and various values of η in the interval $[0,1)$. Smaller η_ℓ implies lesser variation in the amplitude of the kernel. Unlike the shift-invariant kernels used with lattice nodes, low order Walsh kernels are discontinuous and are only piecewise constant.

Covariance kernels for multivariate integrands defined on $[0,1)^d$ are constructed as tensor products:

$$C_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{t}) = K_{\boldsymbol{\theta}}(\boldsymbol{x} \oplus \boldsymbol{t}), \qquad (1.13)$$

$$K_{\boldsymbol{\theta}}(\boldsymbol{x}) = \prod_{\ell=1}^{d} [1 + \eta_\ell \omega_r(x_\ell)], \quad \boldsymbol{\eta} = (\eta_1, \cdots, \eta_d), \quad \boldsymbol{\theta} = (r, \boldsymbol{\eta}). \qquad (1.14)$$

The parameter vector $\boldsymbol{\theta}$ may now be of dimension $d + 1$.

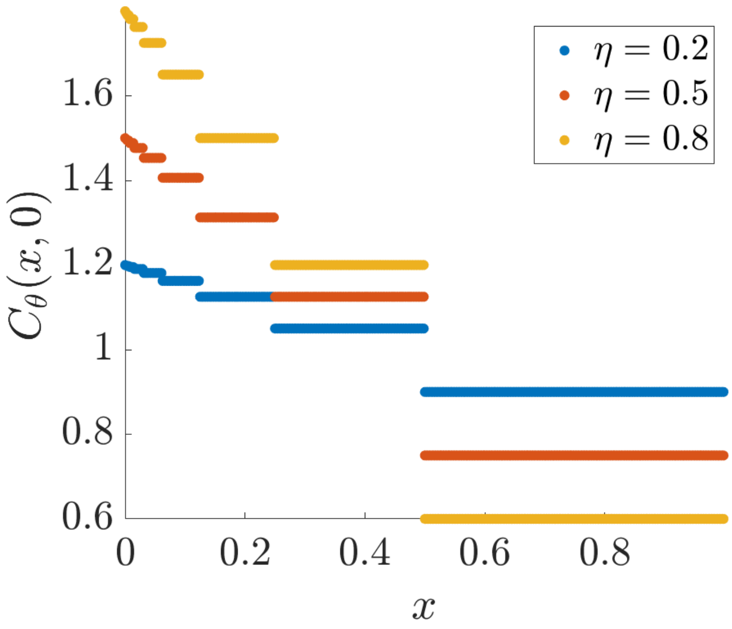

Fig. 1.1 Walsh kernel of order $r = 1$ in dimension $d = 1$. This figure can be reproduced using `plot_walsh_kernel.m`.

1.3.3 Eigenvector-Eigenvalue Decomposition of the Gram Matrix

For fast Bayesian cubature to succeed, the digital net data sites (Section 1.3.1) and the covariance kernels (Section 1.3.2) must match in a way to satisfy conditions (1.4). To do this, we notice that the eigenvectors of the Gram matrix defined in (1.3) are Walsh-Hadamard matrices, defined as follows:

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

$$\mathsf{H}^{(m)} = \mathsf{H}^{(m-1)} \otimes \mathsf{H}^{(1)} = \begin{pmatrix} \mathsf{H}^{(m-1)} & \mathsf{H}^{(m-1)} \\ \mathsf{H}^{(m-1)} & -\mathsf{H}^{(m-1)} \end{pmatrix} = \mathsf{H}^{(1)} \underbrace{\bigotimes \cdots \bigotimes}_{m \text{ times}} \mathsf{H}^{(1)} \qquad (1.15)$$

where $\bigotimes$ is Kronecker product. Note that the Walsh-Hadamard matrices are symmetric.

Lemma 2 *Let* $(\boldsymbol{x}_i)_{i=1}^{2^m}$ *be digitally shifted Sobol' nodes and let the covariance kernel take the form of* (1.13). *Then, for* $m = 1, 2, \ldots$ *the Gram matrix,*

$\mathsf{C}_{\boldsymbol{\theta}}^{(m)} = \big(C_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{x}_j)\big)_{i,j=1}^{2^m} = \big(K(\boldsymbol{x}_i \oplus \boldsymbol{x}_j)\big)_{i,j=1}^{2^m}$ is a 2×2 block-Toeplitz matrix and all the sub-blocks and their sub-sub-blocks, etc. are also 2×2 block-Toeplitz. Moreover, $\mathsf{C}_{\boldsymbol{\theta}}^{(m)} = \mathsf{H}^{(m)}\mathsf{\Lambda}^{(m)}\mathsf{H}^{(m)}$, where $\mathsf{\Lambda}^{(m)}$ is the diagonal matrix of eigenvalues of $\mathsf{C}_{\boldsymbol{\theta}}^{(m)}$.

Proof First define the matrices that by Lemma 1,

$$\begin{aligned}\mathsf{C}_{\boldsymbol{\theta}}^{(m,k)} &= \big(K(\boldsymbol{x}_i \oplus \boldsymbol{x}_{j+k2^m})\big)_{i,j=1}^{2^m} \\ &= \big(K(\boldsymbol{z}_{i\oplus j+k2^m+1})\big)_{i,j=0}^{2^m-1}, \qquad m,k = 0,1,\ldots\end{aligned}$$

which means that $\mathsf{C}_{\boldsymbol{\theta}}^{(m+1,k)}$, has the following block structure

$$\begin{aligned}&\mathsf{C}_{\boldsymbol{\theta}}^{(m+1,k)} \\ &= \begin{pmatrix} \big(K(\boldsymbol{z}_{i\oplus j+k2^{m+1}+1})\big)_{i,j=0}^{2^m-1} & \big(K(\boldsymbol{z}_{i\oplus(j+2^m)+k2^{m+1}+1})\big)_{i,j=0}^{2^m-1} \\ \big(K(\boldsymbol{z}_{(i+2^m)\oplus j+k2^{m+1}+1})\big)_{i,j=0}^{2^m-1} & \big(K(\boldsymbol{z}_{(i+2^m)\oplus(j+2^m)+k2^{m+1}+1})\big)_{i,j=0}^{2^m-1} \end{pmatrix} \\ &= \begin{pmatrix} \mathsf{C}_{\boldsymbol{\theta}}^{(m,2k)} & \mathsf{C}_{\boldsymbol{\theta}}^{(m,2k+1)} \\ \mathsf{C}_{\boldsymbol{\theta}}^{(m,2k+1)} & \mathsf{C}_{\boldsymbol{\theta}}^{(m,2k)} \end{pmatrix}, \end{aligned} \tag{1.16}$$

since for $i,j = 0,\ldots,2^m-1$, it follows that

$$\begin{aligned} i \oplus (j+2^m) + 1 &= (i \oplus j) + 2^m + 1 = (i+2^m) \oplus j + 1, \\ (i+2^m) \oplus (j+2^m) + 1 &= (i \oplus j) + (2^m \oplus 2^m) + 1 = (i \oplus j) + 1. \end{aligned}$$

The proof follows by induction. Note that for the case $m = 1$ and $k = 0,1,\ldots$

$$\mathsf{C}_{\boldsymbol{\theta}}^{(1,k)} = \begin{pmatrix} K_{\boldsymbol{\theta}}(\boldsymbol{z}_{2k+1}) & K_{\boldsymbol{\theta}}(\boldsymbol{z}_{2k+2}) \\ K_{\boldsymbol{\theta}}(\boldsymbol{z}_{2k+2}) & K_{\boldsymbol{\theta}}(\boldsymbol{z}_{2k+1}) \end{pmatrix},$$

which has the desired Toeplitz property. Note also the eigenvectors of $\mathsf{C}_{\boldsymbol{\theta}}^{(1,k)}$ are the columns of $\mathsf{H}^{(1)}$ since

$$\begin{aligned}\mathsf{C}_{\boldsymbol{\theta}}^{(1,k)}\mathsf{H}^{(1)} &= \begin{pmatrix} K_{\boldsymbol{\theta}}(\boldsymbol{z}_{2k+1}) + K_{\boldsymbol{\theta}}(\boldsymbol{z}_{2k+2}) & K_{\boldsymbol{\theta}}(\boldsymbol{z}_{2k+1}) - K_{\boldsymbol{\theta}}(\boldsymbol{z}_{2k+2}) \\ K_{\boldsymbol{\theta}}(\boldsymbol{z}_{2k+2}2k+2) + K_{\boldsymbol{\theta}}(\boldsymbol{z}_{2k+1}) & K_{\boldsymbol{\theta}}(\boldsymbol{z}_{2k+2}) - K_{\boldsymbol{\theta}}(\boldsymbol{z}_{2k+1}) \end{pmatrix} \\ &= \underbrace{\begin{pmatrix} K_{\boldsymbol{\theta}}(\boldsymbol{z}_{2k+1}) + K_{\boldsymbol{\theta}}(\boldsymbol{z}_{2k+2}) & 0 \\ 0 & K_{\boldsymbol{\theta}}(\boldsymbol{z}_{2k+1}) - K_{\boldsymbol{\theta}}(\boldsymbol{z}_{2k+2}) \end{pmatrix}}_{\mathsf{\Lambda}^{(1,k)}} \mathsf{H}^{(1)}.\end{aligned}$$

Now assume that $\mathsf{C}_{\boldsymbol{\theta}}^{(m,k)}$ is 2×2 block-Toeplitz matrix with all its sub-blocks and sub-sub-blocks, etc. also 2×2 block-Toeplitz for $k = 0,1,\ldots$. Then by (1.16), the same holds for $\mathsf{C}_{\boldsymbol{\theta}}^{(m+1,k)}$ for $k = 0,1,\ldots$.

Moreover, the hypothesized eigenvectors of $\mathsf{C}_{\boldsymbol{\theta}}^{(m+1,k)}$ can also be verified by direct calculation under the induction hypothesis:

$$\begin{aligned}&\mathsf{C}_{\boldsymbol{\theta}}^{(m+1,k)}\mathsf{H}^{(m+1)} \\ &\quad = \begin{pmatrix} \mathsf{C}_{\boldsymbol{\theta}}^{(m,2k)} & \mathsf{C}_{\boldsymbol{\theta}}^{(m,2k+1)} \\ \mathsf{C}_{\boldsymbol{\theta}}^{(m,2k+1)} & \mathsf{C}_{\boldsymbol{\theta}}^{(m,2k)} \end{pmatrix} \begin{pmatrix} \mathsf{H}^{(m)} & \mathsf{H}^{(m)} \\ \mathsf{H}^{(m)} & -\mathsf{H}^{(m)} \end{pmatrix}\end{aligned}$$

$$
\begin{aligned}
&= \begin{pmatrix} [\mathsf{C}_{\boldsymbol{\theta}}^{(m,2k)} + \mathsf{C}_{\boldsymbol{\theta}}^{(m,2k+1)}]\mathsf{H}^{(m)} & [\mathsf{C}_{\boldsymbol{\theta}}^{(m,2k)} - \mathsf{C}_{\boldsymbol{\theta}}^{(m,2k+1)}]\mathsf{H}^{(m)} \\ [\mathsf{C}_{\boldsymbol{\theta}}^{(m,2k+1)} + \mathsf{C}_{\boldsymbol{\theta}}^{(m,2k)}]\mathsf{H}^{(m)} & [\mathsf{C}_{\boldsymbol{\theta}}^{(m,2k+1)} - \mathsf{C}_{\boldsymbol{\theta}}^{(m,2k)}]\mathsf{H}^{(m)} \end{pmatrix} \\
&= \underbrace{\begin{pmatrix} \mathsf{\Lambda}_{\boldsymbol{\theta}}^{(m,2k)} + \mathsf{\Lambda}_{\boldsymbol{\theta}}^{(m,2k+1)} & 0 \\ 0 & \mathsf{\Lambda}_{\boldsymbol{\theta}}^{(m,2k)} - \mathsf{\Lambda}_{\boldsymbol{\theta}}^{(m,2k+1)} \end{pmatrix}}_{\mathsf{\Lambda}_{\boldsymbol{\theta}}^{(m+1,2k)}} \begin{pmatrix} \mathsf{H}^{(m)} & \mathsf{H}^{(m)} \\ \mathsf{H}^{(m)} & -\mathsf{H}^{(m)} \end{pmatrix}.
\end{aligned}
$$

Noting that $\mathsf{C}^{(m+1)} = \mathsf{C}^{(m+1,0)}$ completes the proof. □

Lemma 2 establishes that covariance kernels of the form (1.13) matched with shifted digital net data sites satisfy the fast Bayesian cubature assumptions

- (1.4a), since V is the Walsh-Hadamard matrix,
- (1.4b), since the first column and row of the Walsh-Hadamard matrix consist of all ones, and
- (1.4d), since all Walsh functions but the zeroth integrate to zero.

What remains to be shown is how $\mathsf{V}^H\boldsymbol{b} = \mathsf{H}\boldsymbol{b}$ can be calculated in $\mathcal{O}(n\log(n))$ operations for arbitrary $\boldsymbol{b}$.

The computation of $\widetilde{\boldsymbol{b}} = \mathsf{H}\boldsymbol{b}$ is done iteratively as follows. We claim that it requires $m2^m$ operations for vectors $\boldsymbol{b}$ of length 2^m. For the case $m=0$, $\widetilde{\boldsymbol{b}} = \mathsf{H}^{(m)}\boldsymbol{b} = \boldsymbol{b}$, which requires no arithmetic operations. Now, Let $\boldsymbol{b} = (\boldsymbol{b}_1^T, \boldsymbol{b}_2^T)$, where $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ are each of length 2^m, and so $\boldsymbol{b}$ is of length 2^{m+1}. Let $\widetilde{\boldsymbol{b}} = \mathsf{H}^{(m+1)}\boldsymbol{b}$, $\widetilde{\boldsymbol{b}}_1 = \mathsf{H}^{(m)}\boldsymbol{b}_1$, and $\widetilde{\boldsymbol{b}}_2 = \mathsf{H}^{(m)}\boldsymbol{b}_2$. It follows from the definition of the Walsh-Hadamard matrix that

$$
\begin{aligned}
\widetilde{\boldsymbol{b}} = \mathsf{H}^{(m+1)}\boldsymbol{b} &= \begin{pmatrix} \mathsf{H}^{(m)} & \mathsf{H}^{(m)} \\ \mathsf{H}^{(m)} & -\mathsf{H}^{(m)} \end{pmatrix} \begin{pmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{pmatrix} \qquad \text{by (1.15)} \\
&= \begin{pmatrix} \mathsf{H}^{(m)}\boldsymbol{b}_1 + \mathsf{H}^{(m)}\boldsymbol{b}_2 \\ \mathsf{H}^{(m)}\boldsymbol{b}_1 - \mathsf{H}^{(m)}\boldsymbol{b}_2 \end{pmatrix} = \begin{pmatrix} \widetilde{\boldsymbol{b}}_1 + \widetilde{\boldsymbol{b}}_2 \\ \widetilde{\boldsymbol{b}}_1 - \widetilde{\boldsymbol{b}}_2 \end{pmatrix}.
\end{aligned}
$$

Thus to compute $\widetilde{\boldsymbol{b}}$ requires two matrix multiplications by $\mathsf{H}^{(1)}$, at a cost of $m2^m$ each, plus an addition and a subtraction of vectors of length 2^m, for a total cost of $2 \times m2^m + 2 \times 2^m = (m+1)2^{m+1}$, which is exactly what is needed. Since $\widetilde{\boldsymbol{b}} = \mathsf{H}\boldsymbol{b}$ requires $m2^m = n\log_2(n)$ operations to compute, assumption (1.4c) is satisfied. This completes the proof of Theorem 1.

1.4 Numerical Experiments

The Bayesian cubature algorithm described here the digital shift invariant kernel in (1.13) with order $r = 1$ and the Bayesian lattice cubature algorithm described in [11] have been coded as `cubBayesNet_g` and `cubBayesLattice_g`, respectively, in GAIL [3]. We illustrate our algorithm for three common examples, which are also discussed in [11]. The first example evaluates a multivariate Gaussian probability, the second example is

Keister's function [12], and the final example is pricing an Asian arithmetic mean option. For each example we are able to find an "exact" answer, either by analytic computation, or by running a numerical algorithm with a very small error tolerance.

The nodes used in `cubBayesNet_g` are the randomly scrambled and shifted Sobol' points supplied by MATLAB's Sobol' sequence generator. Four hundred different error tolerances, ε, are randomly chosen such that $\log(\varepsilon)$ has a uniform distribution. For each integral example, each ε, and each stopping criterion—empirical Bayes, full Bayes, and generalized cross-validation—we ran `cubBayesNet_g`. For each run, the execution time is plotted against $|\mu - \widehat{\mu}_n| / \varepsilon$. We expect $|\mu - \widehat{\mu}_n| / \varepsilon$ to be no greater than one, but hope that it is not too much smaller than one, which would indicate that the stopping criterion is too conservative.

1.4.1 Multivariate Gaussian Probability

This integral is formulated as in [11] following the variable transformation introduced by Alan Genz [8]. The simulation results are summarized in Figures 1.2, 1.3, and 1.4. In all cases, `cubBayesNet_g` returns an approximation within the prescribed error tolerance. For $\varepsilon = 10^{-5}$ with the empirical Bayes stopping criterion, `cubBayesNet_g` takes about 3 seconds as shown in Figure 1.2 whereas using a Matérn kernel requires 30 seconds to obtain the same accuracy as shown in [11]. This highlights the speed-up possible using fast Bayesian cubature.

The `cubBayesNet_g` uses MATLAB's fast Walsh transform. This is slower than MATLAB's fast Fourier transform, which is implemented in compiled code. This may help explain why `cubBayesLattice_g` is faster than `cubBayesNet_g` for this example. Also, `cubBayesLattice_g` here uses higher order kernels whereas higher order digital shift-invariant kernels are inappropriate for these examples. The `cubBayesLattice_g` uses on average $n \approx 16{,}000$ samples for $\varepsilon = 10^{-5}$, whereas `cubBayesNet_g` uses on average $n \approx 32{,}000$ samples.

Amongst the three stopping criteria, GCV achieves an acceptable approximation faster than others but it is less conservative. One can also observe from the figures that the credible intervals are narrower than empirical Bayes as in Figure 1.2. This shows that `cubBayesNet_g` with $r = 1$ kernel more accurately approximates the integrand than the empirical Bayes.

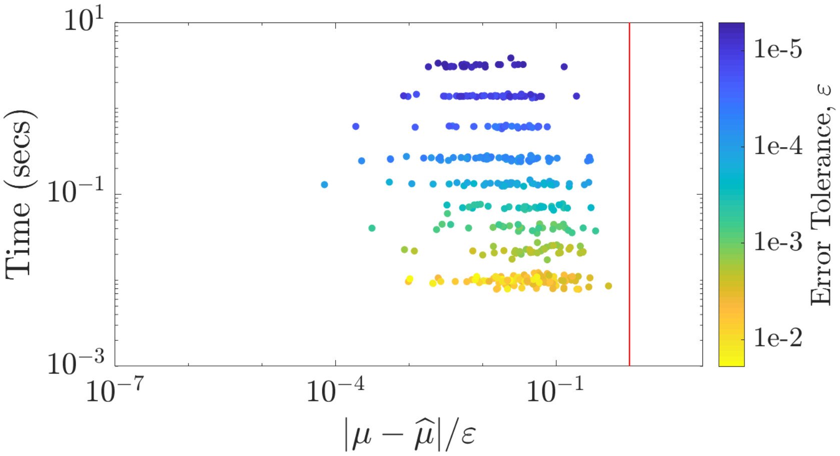

Fig. 1.2 Multivariate normal probability example with empirical Bayes stopping criterion. Algorithm meets the error threshold for all the ϵ randomly chosen in $[10^{-5}, 10^{-2}]$.

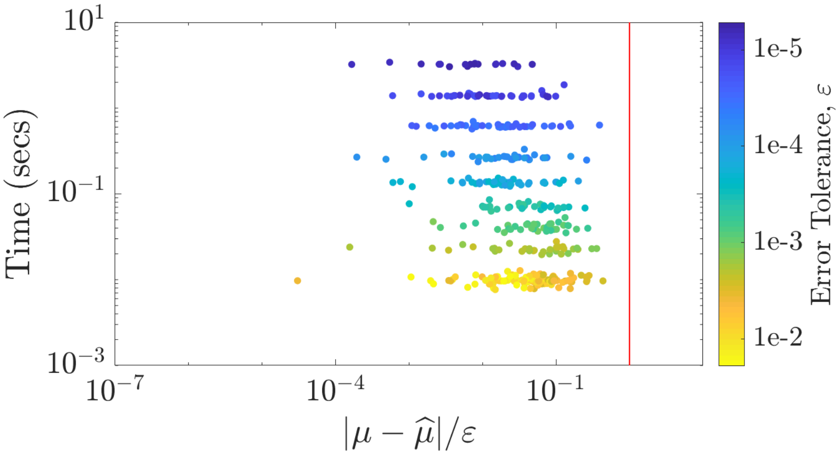

Fig. 1.3 Multivariate normal probability example with the full-Bayes stopping criterion. Algorithm meets the error threshold for all the ϵ randomly chosen in $[10^{-5}, 10^{-2}]$.

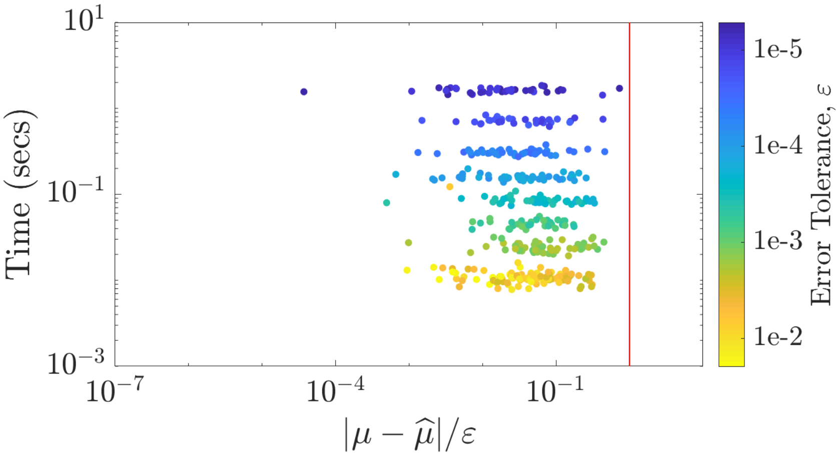

Fig. 1.4 Multivariate normal probability example with the GCV stopping criterion. Algorithm meets the error threshold for all the ϵ randomly chosen in $[10^{-5}, 10^{-2}]$.

1.4.2 Keister's Example

This multidimensional integral function comes from [12] and is inspired by a physics application:

$$\mu = \int_{\mathbb{R}^d} \cos(\|\boldsymbol{t}\|) \exp(-\|\boldsymbol{t}\|^2) \, \mathrm{d}\boldsymbol{t} = \int_{[0,1]^d} f_{\text{Keister}}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}, \qquad (1.17)$$

where

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

and $\boldsymbol{\Phi}$ is the component-wise standard normal distribution. The true value of μ can be calculated iteratively in terms of quadrature as found in [11, Section 5.2].

Figures 1.5, 1.6 and 1.7 summarize the numerical tests for this case for dimension $d = 4$ and order $r = 1$. In [11] `cubBayesLattice_g` used a much smoother kernel that used here. This explains why `cubBayesNet_g` uses more samples for integration. As observed from the figures, the GCV stopping criterion, in Figure 1.7 achieves the results faster than the others but it is less conservative which is also the case with the multivariate Gaussian example.

1.4.3 Asian Option Pricing

The price of financial derivatives can often be modeled by high dimensional integrals. We refer to the formulation of the fair price of the option as in [11],

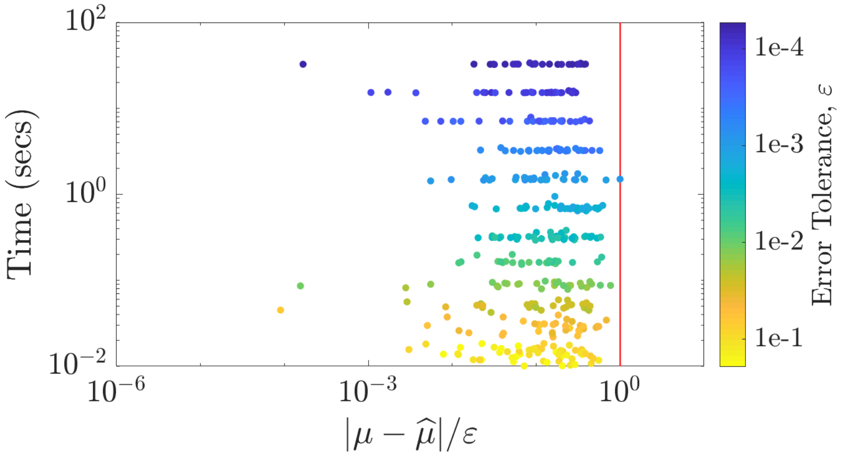

Fig. 1.5 Keister example using the empirical Bayes stopping criterion. Algorithm meets the error threshold for all the ϵ randomly chosen in $[10^{-4}, 10^{-1}]$.

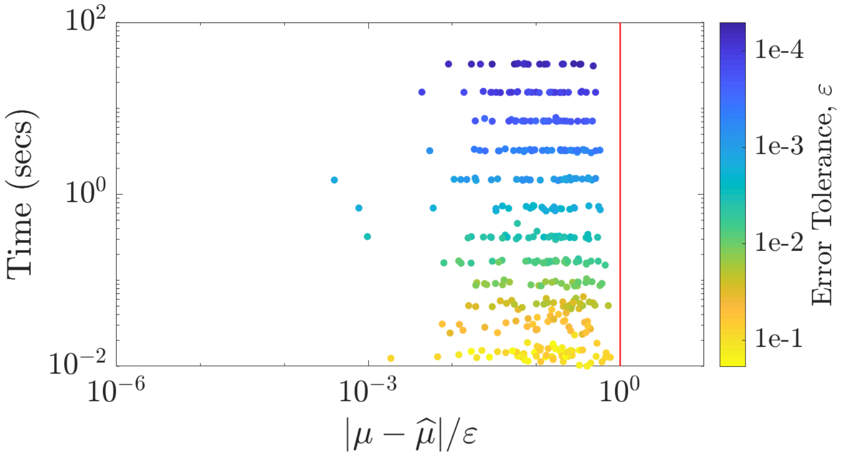

Fig. 1.6 Keister example using the full-Bayes stopping criterion. Algorithm meets the error threshold for all the ϵ randomly chosen in $[10^{-4}, 10^{-1}]$.

where the underlying asset is described in terms of a discretized geometric Brownian motion.

The Figures 1.8, 1.9 and 1.10 summarize the numerical results for the option pricing example using the values for, time horizon $T = 1/4$ of a year, $d = 13$ time steps, initial asset price of $S_0 = 100$, interest rate of $r = 0.05$ per year, volatility of $\sigma = 0.5$ per root year, and strike price of $K = 200$, the same as used in the experiments of `cubBayesLattice_g` [11]. This integrand has a kink caused by the max function, so lattice rules cannot

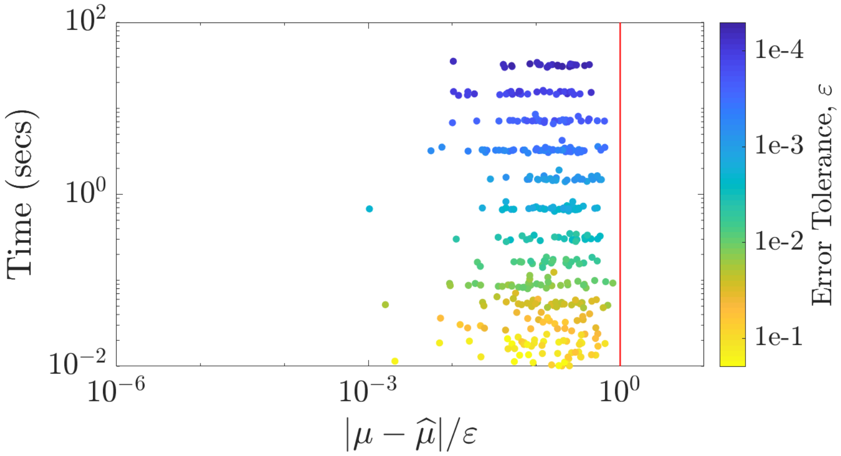

Fig. 1.7 Keister example using the GCV stopping criterion. Algorithm meets the error threshold for all the ϵ randomly chosen in $[10^{-4}, 10^{-1}]$.

perform better, even if the integrand were to be periodized. We observe that `cubBayesNet_g` is more efficient than `cubBayesLattice_g`. For the error tolerance, $\varepsilon = 10^{-3}$, `cubBayesLattice_g` uses $n \approx 2^{20}$ samples, whereas `cubBayesNet_g` uses $n \approx 2^{17}$ samples.

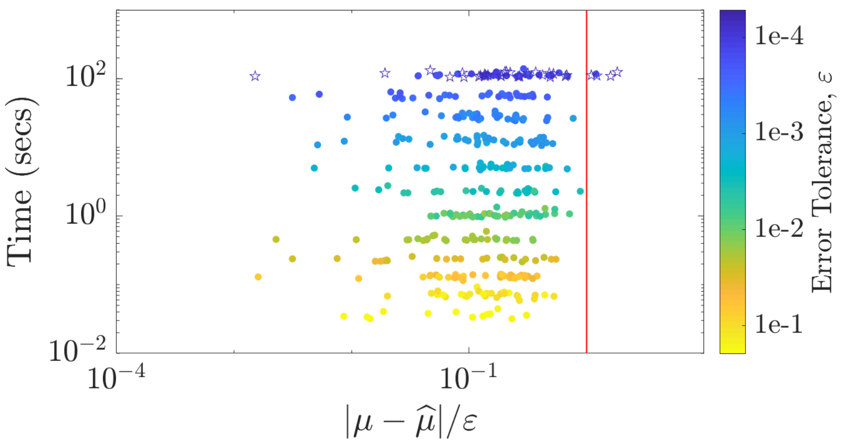

Fig. 1.8 Option pricing using the empirical Bayes stopping criterion. The hollow stars indicate the half-width of the credible interval did not meet the error threshold ϵ before reaching maximum n.

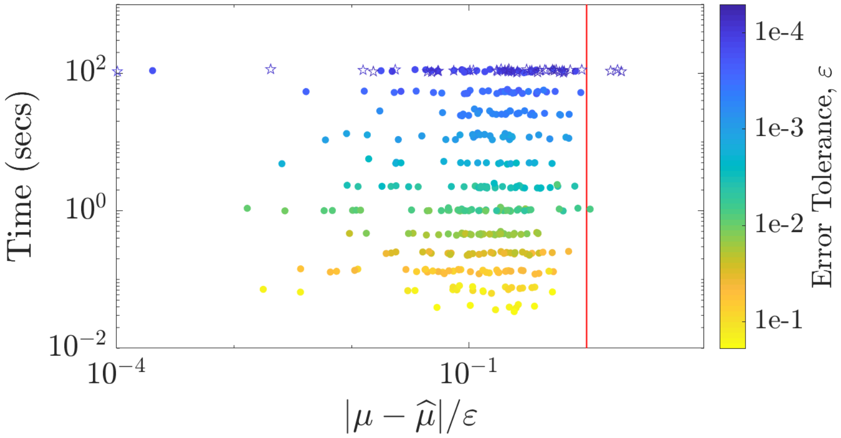

Fig. 1.9 Option pricing using the full-Bayes stopping criterion. The hollow stars indicate the half-width of the credible interval did not meet the error threshold ϵ before reaching maximum n.

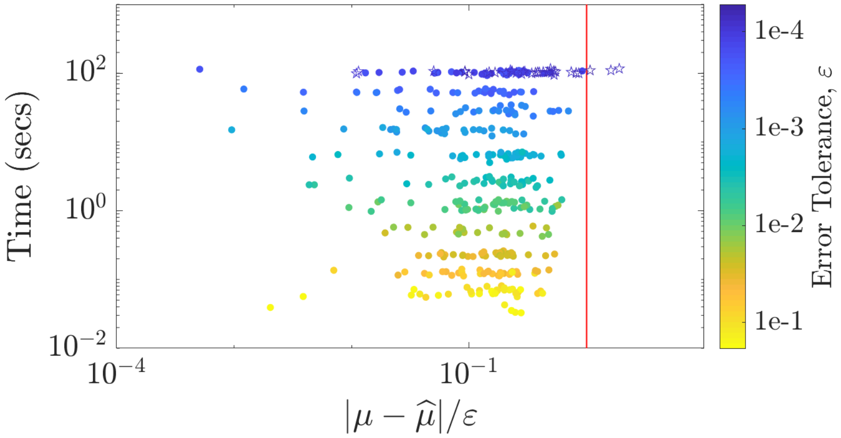

Fig. 1.10 Option pricing using the GCV stopping criterion. The hollow stars indicate the half-width of the credible interval did not meet the error threshold ϵ before reaching maximum n.

1.4.4 Discussion

As shown in Figures 1.2 to 1.10, our algorithm computed the integral within user specified threshold with few exceptions. The exceptions occurred for the option pricing example due to the complexity and high dimension of the integrand. Also notice that our algorithm `cubBayesNet_g`, finished within 40 seconds for Keister and multivariate Gaussian for the smallest tolerance. Option pricing took little above 100 seconds due to the complexity of integrand.

A most noticeable aspect from the plots of `cubBayesNet_g` is how close the error bounds are to the true error. This shows that the `cubBayesNet_g`'s error bounding is not too conservative.

In nearly all of the examples, the ratio $|\mu - \widehat{\mu}_n| / \varepsilon$ is closer to one for the Bayesian net cubature in comparison to the Bayesian lattice cubature. A possible reason is that the periodization transform speeds the convergence in the former case and makes the error bounds too conservative.

1.5 Conclusion and Future Work

We have extended our fast automatic Bayesian cubature to digital net sampling via digital shift-invariant covariance kernels and fast Walsh transforms. Implementation of our algorithm `cubBayesNet_g`, is available in the Guaranteed Automatic Integration Library (GAIL) [3] and Quasi-Monte-Carlo Software in Python (QMCPy) [4]. We demonstrated `cubBayesNet_g` using three example integrand compared with `cubBayesLattice_g`. One major advantage of this algorithm, unlike the `cubBayesLattice_g` developed in [11], is that the integrand does not have to be periodic. However, unlike `cubBayesNet_g`, `cubBayesLattice_g` is more efficient for smoother, periodic functions when using sufficiently smooth and periodic covariance kernels.

The `cubBayesNet_g` in the current implementation uses only the first order kernel and digital nets. Accuracy and speed of the algorithm could be improved by using higher order digital nets and smoother digital shift-invariant covariance kernels. This could help with the smoother integrands. This is a promising direction for future work.

For higher dimensions, both Bayesian cubature algorithms face sometimes fail to produce an acceptable answer within a reasonable amount of time. This seems to be related to an excessive amount of time required to identify the optimal shape parameter $\boldsymbol{\theta}$. The root of the problem and its resolution is a matter for future investigation.

References

1. Beckers, M., Haegemans, A.: Transformation of integrands for lattice rules. In: T.O. Espelid, A.C. Genz (eds.) Numerical Integration: Recent Developments, Software and Applications, pp. 329–340. Kluwer Academic Publishers, Dordrecht (1992)
2. Briol, F.X., Oates, C.J., Girolami, M., Osborne, M.A., Sejdinovic, D.: Probabilistic integration: A role in statistical computation? Statist. Sci. **34**, 1–22 (2019)
3. Choi, S.C.T., Ding, Y., Hickernell, F.J., Jiang, L., Jiménez Rugama, Ll.A., Li, D., Jagadeeswaran, R., Tong, X., Zhang, K., Zhang, Y., Zhou, X.: GAIL: Guaranteed Automatic Integration Library (versions 1.0–2.3.2). MATLAB software, `http://gailgithub.github.io/GAIL_Dev/` (2021). DOI 10.5281/zenodo.4018189
4. Choi, S.C.T., Hickernell, F.J., Jagadeeswaran, R., McCourt, M., Sorokin, A.: QMCPy: A quasi-Monte Carlo Python library (2020). DOI 10.5281/zenodo.3964489. URL `https://qmcsoftware.github.io/QMCSoftware/`
5. Cristea, L.L., Dick, J., Leobacher, G., Pillichshammer, F.: The tent transformation can improve the convergence rate of quasi-Monte Carlo algorithms using digital nets. Numer. Math. **105**, 413–455 (2007)
6. Diaconis, P.: Bayesian numerical analysis. In: S.S. Gupta, J.O. Berger (eds.) Statistical Decision Theory and Related Topics IV, Papers from the 4th Purdue Symp., West Lafayette, Indiana 1986, vol. 1, pp. 163–175. Springer-Verlag, New York (1988)
7. Dick, J., Pillichshammer, F.: Digital Nets and Sequences: Discrepancy Theory and Quasi-Monte Carlo Integration. Cambridge University Press, Cambridge (2010)
8. Genz, A.: Numerical computation of multivariate normal probabilities. J. Comput. Graph. Statist. **1**, 141–150 (1992)
9. Glasserman, P.: Monte Carlo Methods in Financial Engineering, *Applications of Mathematics*, vol. 53. Springer-Verlag, New York (2004)
10. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016). `http://www.deeplearningbook.org`
11. Jagadeeswaran, R., Hickernell, F.J.: Fast automatic Bayesian cubature using lattice sampling. Stat. Comput. **29**, 1215–1229 (2019). DOI 10.1007/s11222-019-09895-9
12. Keister, B.D.: Multidimensional quadrature algorithms. Computers in Physics **10**, 119–122 (1996). DOI 10.1063/1.168565
13. Keller, A.: Quasi-Monte Carlo image synthesis in a nutshell. In: J. Dick, F.Y. Kuo, G.W. Peters, I.H. Sloan (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2012, *Springer Proceedings in Mathematics and Statistics*, vol. 65, pp. 213—249. Springer Berlin Heidelberg (2013)
14. Laurie, D.: Periodizing transformations for numerical integration. J. Comput. Appl. Math. **66**, 337—344 (1996)
15. Matoušek, J.: On the L_2-discrepancy for anchored boxes. J. Complexity **14**, 527–556 (1998)
16. Niederreiter, H., Xing, C.: Rational Points on Curves over Finite Fields: Theory and Applications. No. 285 in London Math. Soc. Lecture Note Ser. Cambridge University Press, Cambridge (2001)
17. Nuyens, D.: The construction of good lattice rules and polynomial lattice rules. Uniform Distribution and Quasi-Monte Carlo Methods (2013). DOI 10.1515/9783110317930.223. URL `http://dx.doi.org/10.1515/9783110317930.223`
18. O'Hagan, A.: Bayes-Hermite quadrature. J. Statist. Plann. Inference **29**, 245–260 (1991). DOI 10.1016/0378-3758(91)90002-V

19. Rasmussen, C.E., Ghahramani, Z.: Bayesian Monte Carlo. In: S. Thrun, L.K. Saul, K. Obermayer (eds.) Advances in Neural Information Processing Systems, vol. 15, pp. 489–496. MIT Press (2003)
20. Rathinavel, J.: Ffast automatic Bayesian cubature using matching kernels and designs. Phd thesis, Illinois Institute of Technology, Chicago (Dec 2019). URL www.math.iit.edu
21. Sidi, A.: A new variable transformation for numerical integration. In: H. Brass, G. Hämmerlin (eds.) Numerical Integration IV, no. 112 in International Series of Numerical Mathematics, pp. 359–373. Birkhäuser, Basel (1993)
22. Sidi, A.: Further extension of a class of periodizing variable transformations for numerical integration. J. Comput. Appl. Math. **221**, 132–149 (2008)
23. Sobol', I.M.: 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)