Fast automatic Bayesian Cubature with Lattice sampling

Jagadeeswaran R.

Department of Applied Mathematics, Illinois Institute of Technology, Chicago IL
jrathin1@iit.edu

• Feb 27, 2019 •

Numerical Integration

A fundamental problem in various fields, including finance, machine learning and statistics.

$$\mu = \int_{\mathbb{R}^d} g(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \int_{[0,1]^d} f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \mathbb{E}[f(\boldsymbol{X})], \quad \text{where} \quad \boldsymbol{X} \sim \mathcal{U}[0,1]^d$$

$$\text{by a cubature rule} \quad \widehat{\mu}_n := w_0 + \sum_{j=1}^{n} f(\boldsymbol{x}_j) w_j$$

using points $\{\boldsymbol{x}_j\}_{j=1}^n$ and associated weights w_j.

Numerical Integration

A fundamental problem in various fields, including finance, machine learning and statistics.

$$\mu = \int_{\mathbb{R}^d} g(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \int_{[0,1]^d} f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \mathbb{E}[f(\boldsymbol{X})], \quad \text{where} \quad \boldsymbol{X} \sim \mathcal{U}[0,1]^d$$

by a cubature rule $\quad \hat{\mu}_n := w_0 + \sum_{j=1}^{n} f(\boldsymbol{x}_j) w_j$

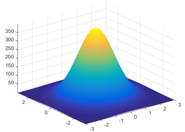
using points $\{\boldsymbol{x}_j\}_{j=1}^n$ and associated weights w_j.

The goal of this work is to

- Develop an automatic algorithm for integration
- Assume f is drawn from a Gaussian process
  - Need to estimate the mean and Covariance kernel
- MLE is expensive in general
  - Use points and kernel for which MLE is cheap
- Use an extensible point-set and an algorithm that allows to add more points if needed
- Determine n such that, given ϵ, $|\mu - \hat{\mu}_n| \leqslant \epsilon$

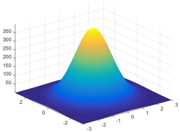
Motivating Examples

Gaussian probability $= \displaystyle\int_{[\boldsymbol{a},\boldsymbol{b}]} \frac{\mathrm{e}^{-\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{x}/2}}{(2\pi)^{d/2}\,|\Sigma|^{1/2}}\,\mathrm{d}\boldsymbol{x}$, (Genz, 1993)



Motivating Examples

$$\text{Gaussian probability} = \int_{[\boldsymbol{a},\boldsymbol{b}]} \frac{\mathrm{e}^{-\boldsymbol{x}^T \Sigma^{-1} \boldsymbol{x}/2}}{(2\pi)^{d/2}\,|\Sigma|^{1/2}}\,\mathrm{d}\boldsymbol{x},$$ (Genz, 1993)



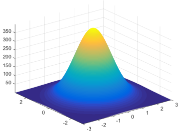
$$\text{Option pricing} = \int_{\mathbb{R}^d} \text{payoff}(\boldsymbol{x}) \underbrace{\frac{\mathrm{e}^{-\boldsymbol{x}^T \Sigma^{-1} \boldsymbol{x}/2}}{(2\pi)^{d/2}\,|\Sigma|^{1/2}}}_{\text{PDF of Brownian motion at } d \text{ times}} \mathrm{d}\boldsymbol{x},$$ (Glasserman, 2004)

$$\text{where} \quad \text{payoff}(\boldsymbol{x}) = \mathrm{e}^{-rT} \max\left(\frac{1}{d}\sum_{k=1}^{d} S_k(x_k) - K, 0\right)$$

$$S_j(x_j) = S_0 \mathrm{e}^{(r-\sigma^2/2)t_j + \sigma x_j} = \text{stock price at time } t_j = jT/d;$$

Motivating Examples

Gaussian probability $= \displaystyle\int_{[\boldsymbol{a},\boldsymbol{b}]} \frac{\mathrm{e}^{-\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{x}/2}}{(2\pi)^{d/2}\,|\Sigma|^{1/2}}\,\mathrm{d}\boldsymbol{x}$, (Genz, 1993)



Option pricing $= \displaystyle\int_{\mathbb{R}^d} \text{payoff}(\boldsymbol{x}) \underbrace{\frac{\mathrm{e}^{-\boldsymbol{x}^T\Sigma^{-1}\boldsymbol{x}/2}}{(2\pi)^{d/2}\,|\Sigma|^{1/2}}}_{\text{PDF of Brownian motion at } d \text{ times}} \mathrm{d}\boldsymbol{x}$, (Glasserman, 2004)

where $\quad \text{payoff}(\boldsymbol{x}) = \mathrm{e}^{-rT} \max\left(\frac{1}{d}\sum_{k=1}^{d} S_k(x_k) - K, 0\right)$

$S_j(x_j) = S_0\mathrm{e}^{(r-\sigma^2/2)t_j+\sigma x_j}$ = stock price at time $t_j = jT/d$;

Keister integral $= \displaystyle\int_{\mathbb{R}^d} \cos(\|\boldsymbol{x}\|)\exp(-\|\boldsymbol{x}\|^2)\,\mathrm{d}\boldsymbol{x}, \quad d = 1, 2, \ldots$ (Keister, 1996)

procedure AUTOCUBATURE(f, ϵ)

Require: a generator for the sequence $x_1, x_2, \ldots$; a black-box function, f; an absolute error tolerance, $\varepsilon > 0$; the positive initial sample size, n_0; the maximum sample size n_{max}

    $n \leftarrow n_0,\ n' \leftarrow 0,\ \text{err}_\text{n} \leftarrow \infty$
    while $\text{err}_\text{n} > \varepsilon$ and $n \leqslant n_{\text{max}}$ **do**
        Generate $\{x_i\}_{i=n'+1}^{n}$ and sample $\{f(x_i)\}_{i=n'+1}^{n}$,
        Compute θ
        Compute error bound err_n
        $n' \leftarrow n, n \leftarrow 2 \times n'$
    end while
    Sample size to compute $\hat{\mu}$, $n \leftarrow n'$
    Compute approximate $\hat{\mu}_n$, the approximate integral
    return $\hat{\mu}_n$ ▷ Integral estimate $\hat{\mu}_n$
end procedure

Problem:

- How to choose $\{x_i\}_{i=1}^{n}$, and $\{w_i\}_{i=1}^{n}$ to make $|\mu - \hat{\mu}_n|$ small? what is err_n? (Bayesian posterior error)
- How to find n such that $|\mu - \hat{\mu}_n| \leqslant \text{err}_\text{n} \leqslant \epsilon$? (automatic cubature)

```
procedure AutoCubature(f, ϵ)
Require: a generator for the sequence x_1, x_2, ...; a black-box function, f; an absolute error tolerance, ε > 0; the positive initial sample size, n_0; the maximum sample size n_max
    n ← n_0, n' ← 0, err_n ← ∞
    while err_n > ε and n ⩽ n_max do
        Generate {x_i}_{i=n'+1}^n and sample {f(x_i)}_{i=n'+1}^n,
        Compute θ
        Compute error bound err_n
        n' ← n, n ← 2 × n'
    end while
    Sample size to compute μ̂, n ← n'
    Compute approximate μ̂_n, the approximate integral
    return μ̂_n                                        ▷ Integral estimate μ̂_n
end procedure
```

Problem:

- How to choose $\{x_i\}_{i=1}^n$, and $\{w_i\}_{i=1}^n$ to make $|\mu - \hat{\mu}_n|$ small? what is $\mathrm{err_n}$? (Bayesian posterior error)
- How to find n such that $|\mu - \hat{\mu}_n| \leqslant \mathrm{err_n} \leqslant \epsilon$? (automatic cubature)

Bayesian posterior error

Assume random $f \sim \mathcal{GP}(m, s^2 C_{\boldsymbol{\theta}})$, a Gaussian process with mean m and covariance kernel, $s^2 C_{\boldsymbol{\theta}}$, $C_{\boldsymbol{\theta}} : [0,1] \times [0,1] \to \mathbb{R}$.

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

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

Bayesian posterior error

Assume random $f \sim \mathcal{GP}(m, s^2 C_\theta)$, a Gaussian process with mean m and covariance kernel, $s^2 C_\theta$, $C_\theta : [0,1] \times [0,1] \to \mathbb{R}$.

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

$$\boldsymbol{c} = \left(\int_{[0,1]} C_\theta(\boldsymbol{x}_i, \boldsymbol{t}) \mathrm{d}\boldsymbol{t} \right)_{i=1}^{n}, \quad \mathsf{C} = \Big(C_\theta(\boldsymbol{x}_i, \boldsymbol{x}_j) \Big)_{i,j=1}^{n}$$

$$\mu - \hat{\mu}_n \big| \boldsymbol{y} \ \sim \ \mathcal{N}\left(-w_0 + m(1 - \mathbf{1}^T \mathsf{C}^{-1} \boldsymbol{c}) + \boldsymbol{y}^T (\mathsf{C}^{-1} \boldsymbol{c} - \boldsymbol{w}), \quad s^2 (c_0 - \boldsymbol{c}^T \mathsf{C}^{-1} \boldsymbol{c}) \right)$$

where $\boldsymbol{y} = \big(f(\boldsymbol{x}_i)\big)_{i=1}^{n}$. Moreover m, s and θ needs to be inferred.

$$\hat{\mu}_n = w_0 + \sum_{i=1}^{n} w_i f(\boldsymbol{x}_i) = w_0 + \boldsymbol{w}^T \boldsymbol{y}$$

Bayesian posterior error

Assume random $f \sim \mathcal{GP}(m, s^2 C_\theta)$, a Gaussian process with mean m and covariance kernel, $s^2 C_\theta$, $C_\theta : [0,1] \times [0,1] \to \mathbb{R}$.

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

$$\boldsymbol{c} = \left(\int_{[0,1]} C_\theta(\boldsymbol{x}_i, \boldsymbol{t}) \mathrm{d}\boldsymbol{t}\right)_{i=1}^{n}, \quad \mathsf{C} = \Big(C_\theta(\boldsymbol{x}_i, \boldsymbol{x}_j)\Big)_{i,j=1}^{n}$$

$$\mu - \hat{\mu}_n \big| \boldsymbol{y} \sim \mathcal{N}\left(-w_0 + m(1 - \mathbf{1}^T \mathsf{C}^{-1}\boldsymbol{c}) + \boldsymbol{y}^T(\mathsf{C}^{-1}\boldsymbol{c} - \boldsymbol{w}), \quad s^2(c_0 - \boldsymbol{c}^T \mathsf{C}^{-1}\boldsymbol{c})\right)$$

where $\boldsymbol{y} = \big(f(\boldsymbol{x}_i)\big)_{i=1}^{n}$. Moreover m, s and θ needs to be inferred.

$$\hat{\mu}_n = w_0 + \sum_{i=1}^{n} w_i f(\boldsymbol{x}_i) = w_0 + \boldsymbol{w}^T \boldsymbol{y}$$

In general choosing $w_0 = m(1 - \mathbf{1}^T \mathsf{C}^{-1}\boldsymbol{c})$, $\boldsymbol{w} = \mathsf{C}^{-1}\boldsymbol{c}$, makes error unbiased

If $m = 0$ fixed, choosing $\boldsymbol{w} = \mathsf{C}^{-1}\boldsymbol{c}$, makes error unbiased

Diaconis (1988), O'Hagan (1991), Ritter (2000), Rasmussen (2003) and others

Parameter estimation - Maximum likelihood

The log-likelihood of the parameters given the data $\boldsymbol{y} = (f(\boldsymbol{x}_i))_{i=1}^n$ is :

$$l(s, \theta | y) = \log \left[\frac{1}{\sqrt{(2\pi)^n \det(s^2 \mathsf{C})}} \exp\left(-\frac{1}{2} s^{-2} (\boldsymbol{y} - m\mathbf{1})^T \mathsf{C}^{-1} (\boldsymbol{y} - m\mathbf{1}) \right) \right]$$

Parameter estimation - Maximum likelihood

The log-likelihood of the parameters given the data $\boldsymbol{y} = (f(\boldsymbol{x}_i))_{i=1}^n$ is :

$$l(s, \theta|y) = \log \left[\frac{1}{\sqrt{(2\pi)^n \det(s^2 \mathsf{C})}} \exp\left(-\frac{1}{2} s^{-2} (\boldsymbol{y} - m\boldsymbol{1})^T \mathsf{C}^{-1} (\boldsymbol{y} - m\boldsymbol{1}) \right) \right]$$

Maximising w.r.t m and then s^2, further with θ :

$$m_{\text{MLE}} = \frac{\boldsymbol{1}^T \mathsf{C}^{-1} \boldsymbol{y}}{\boldsymbol{1}^T \mathsf{C}^{-1} \boldsymbol{1}}, \quad s^2_{\text{MLE}} = \frac{1}{n} (\boldsymbol{y} - m_{\text{MLE}} \boldsymbol{1})^T \mathsf{C}^{-1} (\boldsymbol{y} - m_{\text{MLE}} \boldsymbol{1}), \qquad \text{(Explicit)}$$

$$\theta_{\text{MLE}} = \underset{\theta}{\operatorname{argmin}} \log \left(\frac{1}{2n} \log(\det \mathsf{C}) + \log(s_{\text{MLE}}) \right) \qquad \text{(numeric)}$$

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

Parameter estimation - Maximum likelihood

The log-likelihood of the parameters given the data $\boldsymbol{y} = (f(\boldsymbol{x}_i))_{i=1}^n$ is :

$$l(s, \theta|y) = \log\left[\frac{1}{\sqrt{(2\pi)^n \det(s^2\mathsf{C})}} \exp\left(-\frac{1}{2}s^{-2}(\boldsymbol{y} - m\mathbf{1})^T\mathsf{C}^{-1}(\boldsymbol{y} - m\mathbf{1})\right)\right]$$

Maximising w.r.t m and then s^2, further with θ :

$$m_{\mathsf{MLE}} = \frac{\mathbf{1}^T\mathsf{C}^{-1}\boldsymbol{y}}{\mathbf{1}^T\mathsf{C}^{-1}\mathbf{1}}, \quad s^2_{\mathsf{MLE}} = \frac{1}{n}(\boldsymbol{y} - m_{\mathsf{MLE}}\mathbf{1})^T\mathsf{C}^{-1}(\boldsymbol{y} - m_{\mathsf{MLE}}\mathbf{1}), \qquad \text{(Explicit)}$$

$$\theta_{\mathsf{MLE}} = \operatorname*{argmin}_{\theta} \log\left(\frac{1}{2n}\log(\det\mathsf{C}) + \log(s_{\mathsf{MLE}})\right) \qquad \text{(numeric)}$$

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

Why do we need θ_{MLE}? Function space spanned by C_θ customized to contain the integrand function f.

Parameter estimation - Full Bayes

Treat m and s 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 μ given the data is :

$$\rho_\mu(z|\boldsymbol{f}=\boldsymbol{y}) \propto \int_0^\infty \int_{-\infty}^\infty \rho_\mu(z|\boldsymbol{f}=\boldsymbol{y}, m=\xi, s^2=\lambda)\,\rho_{\boldsymbol{f}}(\boldsymbol{y}|\xi,\lambda)\,\rho_{m,s^2}(\xi,\lambda)\,\mathrm{d}\xi\mathrm{d}\lambda$$

$$\propto \left(1+\frac{1}{n-1}\frac{(z-\mu_{\mathsf{full}})^2}{\hat{\sigma}^2_{\mathsf{full}}}\right)^{-n/2}$$

Parameter estimation - Full Bayes

Treat m and s as hyper-parameters with a non-informative, conjugate prior, namely $\rho_{m,s^2}(\xi,\lambda) \propto 1/\lambda$. Then the posterior density for the integral μ given the data is :

$$\rho_\mu(z|\boldsymbol{f}=\boldsymbol{y}) \propto \int_0^\infty \int_{-\infty}^\infty \rho_\mu(z|\boldsymbol{f}=\boldsymbol{y}, m=\xi, s^2=\lambda)\rho_{\boldsymbol{f}}(\boldsymbol{y}|\xi,\lambda)\rho_{m,s^2}(\xi,\lambda)\,\mathrm{d}\xi\mathrm{d}\lambda$$

$$\propto \left(1 + \frac{1}{n-1}\frac{(z-\mu_{\text{full}})^2}{\hat{\sigma}^2_{\text{full}}}\right)^{-n/2}$$

Where :

$$\mu_{\text{full}} = \mu_{\text{MLE}}$$

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

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

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

Parameter estimation - Leave-one-out Cross validation

Let $\tilde{y}_i = \mathbb{E}[f(\boldsymbol{x}_i) | \boldsymbol{f}_{-i} = \boldsymbol{y}_{-i}]$. The cross-validation criterion, which is to be minimized, is sum of squares of the difference between these conditional expectations and the observed values: :

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

$$a_{ii} \text{ are diagonal elems of } \mathsf{C}^{-1} = \begin{pmatrix} a_{ii} & \boldsymbol{A}_{-i,i}^T \\ \boldsymbol{A}_{-i,i} & \mathsf{A}_{-i,-i} \end{pmatrix}$$

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

Parameter estimation - Leave-one-out Cross validation

Let $\tilde{y}_i = \mathbb{E}[f(\boldsymbol{x}_i) | \boldsymbol{f}_{-i} = \boldsymbol{y}_{-i}]$. The cross-validation criterion, which is to be minimized, is sum of squares of the difference between these conditional expectations and the observed values: :

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

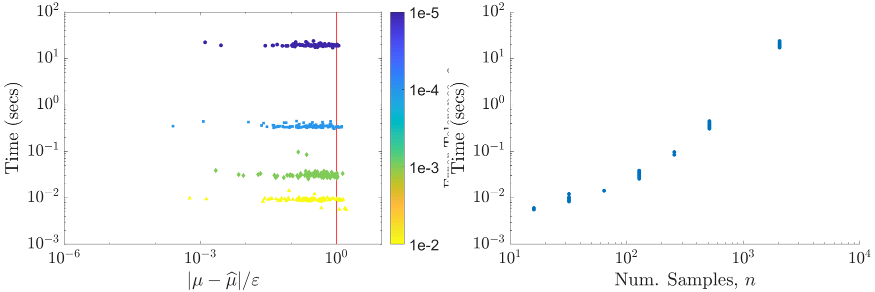
$$a_{ii} \text{ are diagonal elems of } \mathsf{C}^{-1} = \begin{pmatrix} a_{ii} & \boldsymbol{A}_{-i,i}^T \\ \boldsymbol{A}_{-i,i} & \mathsf{A}_{-i,-i} \end{pmatrix}$$

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

$$\theta_{\mathsf{GCV}} = \underset{\theta}{\operatorname{argmin}} \left\{ \log\left(\boldsymbol{y}^T \left[\mathsf{C}^{-2} - \frac{\mathsf{C}^{-2}\boldsymbol{1}\boldsymbol{1}^T\mathsf{C}^{-2}}{\boldsymbol{1}^T\mathsf{C}^{-2}\boldsymbol{1}} \right] \boldsymbol{y} \right) - 2\log\left(\operatorname{trace}(\mathsf{C}^{-1})\right) \right\}$$

$$s_{\mathsf{GCV}}^2 := \boldsymbol{y}^T \left[\mathsf{C}^{-2} - \frac{\mathsf{C}^{-2}\boldsymbol{1}\boldsymbol{1}^T\mathsf{C}^{-2}}{\boldsymbol{1}^T\mathsf{C}^{-2}\boldsymbol{1}} \right] \boldsymbol{y} \left[\operatorname{trace}(\mathsf{C}^{-1})\right]^{-1}, \quad m_{\mathsf{GCV}} := \frac{\boldsymbol{1}^T\mathsf{C}^{-2}\boldsymbol{y}}{\boldsymbol{1}^T\mathsf{C}^{-2}\boldsymbol{1}}.$$

Multivariate normal integration with Matern kernel



Problem: Computation time (in seconds) increases rapidly, so it's not practical to use more than 4000 points in the cubature.

Fast transform kernel

Choose the kernel C_θ and $\{\boldsymbol{x}_i\}_{i=1}^n$, so the Gram matrix $\mathsf{C} = \left(C_\theta(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)_{i,j=1}^n$ has the special properties:

$$\mathsf{C} = \left(C_\theta(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)_{i,j=1}^n = (\boldsymbol{C}_1, ..., \boldsymbol{C}_n) = \frac{1}{n}\mathsf{V}\Lambda\mathsf{V}^H$$
$$\mathsf{V} := (\boldsymbol{v}_1, ..., \boldsymbol{v}_n)^T = (\boldsymbol{V}_1, ..., \boldsymbol{V}_n), \quad \boldsymbol{V}_1 = \boldsymbol{v}_1 = \boldsymbol{1},$$
$$\Lambda = \mathrm{diag}(\boldsymbol{\lambda}), \quad \boldsymbol{\lambda} = (\lambda_1, ..., \lambda_n)$$

Then

Fast transform kernel

Choose the kernel C_θ and $\{\boldsymbol{x}_i\}_{i=1}^n$, so the Gram matrix $\mathsf{C} = \left(C_\theta(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)_{i,j=1}^n$ has the special properties:

$$\mathsf{C} = \left(C_\theta(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)_{i,j=1}^n = (\boldsymbol{C}_1, ..., \boldsymbol{C}_n) = \frac{1}{n}\mathsf{V}\Lambda\mathsf{V}^H$$

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

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

Then

$$\mathsf{V}^H\boldsymbol{C}_1 = \mathsf{V}^H\left(\frac{1}{n}\mathsf{V}\Lambda\boldsymbol{v}_1^*\right) = \Lambda\boldsymbol{1} = (\lambda_1, \ldots, \lambda_n)^T = \boldsymbol{\lambda}$$

C_θ is a fast transform kernel, if the transform $\boxed{\hat{\boldsymbol{z}} = \mathsf{V}^H\boldsymbol{z}}$ for arbitrary z, can be done in $\mathcal{O}(n \log n)$. Using the fast transform,

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

The covariance kernel used in practice also may be normalized

$$\int_{[0,1]^d} C(\boldsymbol{t}, \boldsymbol{x})\,\mathrm{d}\boldsymbol{t} = 1 \qquad \forall \boldsymbol{x} \in [0,1]^d,$$ leading to $c_0 = 1$ and $\boldsymbol{c} = \boldsymbol{1}$.

Faster parameters estimation

MLE and GCV estimates of θ made faster by using the properties of the fast transform kernel:

$$\theta_{\text{MLE}} = \underset{\theta}{\operatorname{argmin}} \left[\log\left(\sum_{i=2}^{n} \frac{|\hat{y}_i|^2}{\lambda_i} \right) + \frac{1}{n} \sum_{i=1}^{n} \log(\lambda_i) \right],$$

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

Also,

$$m_{\text{MLE}} = m_{\text{GCV}} = \frac{1}{n} \sum_{i=1}^{n} y_i, \quad s^2_{\text{MLE}} = \frac{1}{n} \sum_{i=2}^{n} \frac{\left|\hat{\boldsymbol{y}}_i\right|^2}{\lambda_i}, \quad s^2_{\text{GCV}} = \frac{1}{n} \sum_{i=2}^{n} \frac{|\tilde{y}_i|^2}{\lambda_i^2} \left[\sum_{i=1}^{n} \frac{1}{\lambda_i} \right]^{-1}$$

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

$$\hat{\boldsymbol{y}} = (\hat{y}_i)_{i=1}^n = \mathsf{V}^T \boldsymbol{y}, \qquad \boldsymbol{\lambda} = (\lambda_i)_{i=1}^n = \mathsf{V}^T \boldsymbol{C}_1, \quad \text{where } \boldsymbol{C}_1 = \big(C(\boldsymbol{x}_i, \boldsymbol{x}_1) \big)_{i=1}^n$$

$\mathcal{O}(n \log n)$ operations to compute $\hat{\boldsymbol{y}}$ and $\hat{\boldsymbol{\lambda}}$, So the θ_{MLE}

Computing the error bound err and $\hat{\mu}$ faster

Using the properties of the fast transform kernel, the error bound err_n can be computed faster

$$\text{err}_{\text{MLE}} = \frac{2.58}{n}\left\{\sum_{i=2}^{n}\frac{|\hat{y}_i|^2}{\lambda_i}\left(1-\frac{n}{\lambda_1}\right)\right\}^{1/2}$$

$$\text{err}_{\text{full}} = t_{n_j-1,0.995}\left\{\frac{1}{n(n-1)}\sum_{i=2}^{n}\frac{|\tilde{y}_i|^2}{\lambda_i}\left(\frac{\lambda_1}{n}-1\right)\right\}^{1/2},$$

$$\text{err}_{\text{GCV}} = \frac{2.58}{n}\left\{\sum_{i=2}^{n}\frac{|\tilde{y}_i|^2}{\lambda_i^2}\left[\frac{1}{n}\sum_{i=1}^{n}\frac{1}{\lambda_i}\right]^{-1}\times\left(1-\frac{n}{\lambda_1}\right)\right\}^{1/2}$$

similarly, $\hat{\mu}$ can be computed faster

$$\hat{\mu}_{\text{MLE}} = \hat{\mu}_{\text{full}} = \hat{\mu}_{\text{GCV}} = \boldsymbol{w}^T\boldsymbol{y} = \boxed{\sum_{i=1}^{n}\frac{y_i}{n}}$$

where

$$\hat{\boldsymbol{y}} = \mathsf{V}^T\boldsymbol{y}, \quad \boldsymbol{\lambda} = \mathsf{V}^T\boldsymbol{C}_1, \text{ where } \boldsymbol{C}_1 = (C(\boldsymbol{x}_i,\boldsymbol{x}_1))_{i=1}^{n}$$

$\mathcal{O}(n\log n)$ operations to compute the err. $\mathcal{O}(n)$ operations to compute the $\hat{\mu}$

Special shift invariant covariance kernel

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

Special shift invariant covariance kernel

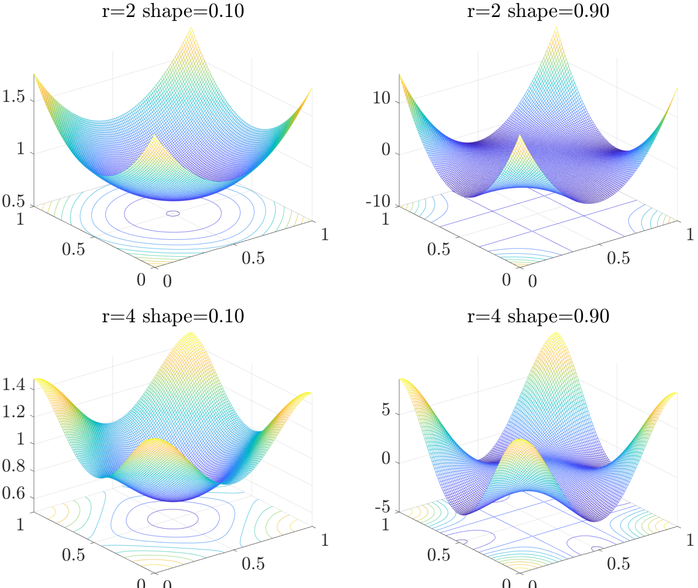
$$C_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{t}) = \prod_{l=1}^{d} 1 - \theta_l^r \frac{(2\pi\sqrt{-1})^r}{r!} B_r(|x_l - t_l|), \quad \boldsymbol{\theta} \in (0,1]^d, \quad r \in 2\mathbb{N}$$

where B_r is Bernoulli polynomial of order r (Olver et al., 2013). We call $C_{\boldsymbol{\theta}}$, Fourier kernel. Also this kernel has:

$$c_0 = \int_{[0,1]^2} C_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{t}) \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{t} = 1, \qquad \boldsymbol{c} = \left(\int_{[0,1]} C_{\boldsymbol{\theta}}(\boldsymbol{x}_i, \boldsymbol{t}) \mathrm{d}\boldsymbol{t} \right)_{i=1}^{n} = \mathbf{1}.$$

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

Fourier kernel



Rank-1 Lattice rules : low discrepancy point set

Given the "generating vector" $\boldsymbol{h}$, the construction of n - Rank-1 lattice points (Dick and Pillichshammer, 2010) is given by

$$\mathsf{L}_{n,\boldsymbol{h}} := \{\boldsymbol{x}_i := \boldsymbol{h}\phi(i-1) \textbf{ mod } 1; \;\; i = 1, \ldots, n\} \tag{1}$$

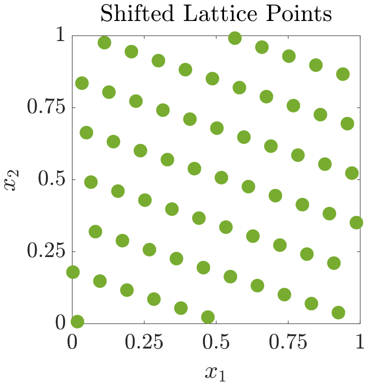
where $\boldsymbol{h}$ is a *generalized Mahler integer* (∞ digit expression) (Hickernell and Niederreiter, 2003) also called generating vector. $\phi(i)$ is the Van der Corput sequence in base 2. Then the Lattice rule approximation is

$$\frac{1}{n}\sum_{k=1}^{n} f\left(\left\{\frac{k\boldsymbol{h}}{n} + \boldsymbol{\Delta}\right\}_1\right)$$

where $\{.\}_1$ the fractional part, i.e, *modulo 1* operator and $\boldsymbol{\Delta}$ a random shift.

Extensible integration lattices : The number of points in the node set can be increased while retaining the existing points.(Hickernell and Niederreiter, 2003)

Rank-1 Lattice points in $d = 2$



Shift invariant kernel + Lattice points = '*Symmetric circulant kernel*' matrix

The shift invariant kernel with rank-1 Lattice points

- Satisfies all the requirements to be a fast transform kernel
- Fast transform = fast Fourier transform
- Complexity of fast Fourier transform is $\mathcal{O}(n \log n)$
- No need to form the kernel matrix C explicitly, so $\mathcal{O}(n^2)$ memory not required
- There are no matrix inversions, no matrix multiplications
- Factorization of matrix C does not need any computations.

  where V is just the Fourier coefficient matrix: $\mathsf{V} = \left(e^{2\pi n\sqrt{-1}(i-1)(j-1)}\right)_{i=1}^{n}$

Iterative DFT

We can avoid recomputing the whole Fourier transform for function values $vy = (y_i = f(\boldsymbol{x}_i))_{i=1}^n$ in every iteration. Discrete Fourier transform is defined as

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

Iterative DFT

We can avoid recomputing the whole Fourier transform for function values $vy = (y_i = f(x_i))_{i=1}^n$ in every iteration. Discrete Fourier transform is defined as

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

Rearrange sum into even indexed $j = 2l$ and odd indexed $j = 2l + 1$.

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

we use this concept along with *extensible point set*, to avoid recomputing the DFT of $\boldsymbol{y}$ in every iteration.

Cancellation error in err

$$\mathrm{err_n} = 2.58\sqrt{\left(1 - \frac{n}{\lambda_1}\right)\frac{1}{n^2}\sum_{i=2}^{n}\frac{\left|\widehat{y}_i\right|^2}{\lambda_i}},$$ term $1 - \dfrac{n}{\lambda_1}$ causes cancellation error

Cancellation error in err

$$\mathrm{err_n} = 2.58\sqrt{\left(1-\frac{n}{\lambda_1}\right)\frac{1}{n^2}\sum_{i=2}^{n}\frac{|\widehat{y}_i|^2}{\lambda_i}}, \quad \text{term } 1-\frac{n}{\lambda_1} \text{ causes cancellation error}$$

$$\text{Let } \widetilde{C}(\boldsymbol{x},\boldsymbol{t}) = C(\boldsymbol{x},\boldsymbol{t}) - 1, \quad \text{then } \widetilde{\mathsf{C}} = \mathsf{C} - \mathbf{1}\mathbf{1}^T, \quad \text{and } \widetilde{\mathsf{C}} = \mathsf{V}\widetilde{\Lambda}\mathsf{V}^H$$

where

$$\widetilde{\Lambda} = \mathrm{diag}(\tilde{\lambda}_1, \dots, \tilde{\lambda}_n), \text{ to compute } \boxed{(\tilde{\lambda}_i)_{i=1}^n = \mathsf{V}^T\widetilde{\boldsymbol{C}}_1}$$

$$\tilde{\lambda}_1 = \lambda_1 - n, \quad \tilde{\lambda}_j = \lambda_j, \ \forall\, j = 2, \dots, n$$

Cancellation error in err

$$\text{err}_n = 2.58\sqrt{\left(1-\frac{n}{\lambda_1}\right)\frac{1}{n^2}\sum_{i=2}^{n}\frac{|\widehat{y}_i|^2}{\lambda_i}},\quad \text{term } 1-\frac{n}{\lambda_1} \text{ causes cancellation error}$$

$$\text{Let } \widetilde{C}(\boldsymbol{x},\boldsymbol{t}) = C(\boldsymbol{x},\boldsymbol{t}) - 1,\quad \text{then } \widetilde{\mathsf{C}} = \mathsf{C} - \mathbf{1}\mathbf{1}^T,\quad \text{and } \widetilde{\mathsf{C}} = \mathsf{V}\widetilde{\Lambda}\mathsf{V}^H$$

where

$$\widetilde{\Lambda} = \text{diag}(\tilde{\lambda}_1, \dots, \tilde{\lambda}_n),\ \text{to compute } \boxed{(\tilde{\lambda}_i)_{i=1}^n = \mathsf{V}^T\widetilde{\boldsymbol{C}}_1}$$

$$\tilde{\lambda}_1 = \lambda_1 - n,\quad \tilde{\lambda}_j = \lambda_j,\ \forall\, j = 2, \dots, n$$

vector $\widetilde{\boldsymbol{C}}_1 = \widetilde{\boldsymbol{C}}_1^{(d)}$ computed iteratively

$$\widetilde{\boldsymbol{C}}_1^{(1)} = \theta\left(B(x_{i1}-x_{11})\right)_{i=1}^n,\quad \boldsymbol{C}_1^{(1)} = \mathbf{1} + \widetilde{\boldsymbol{C}}_1^{(1)},$$

$$\forall 1 < k \leqslant d,\quad \widetilde{\boldsymbol{C}}_1^{(k)} = \theta\boldsymbol{C}_1^{(k-1)} \circ \left(B(x_{ik}-x_{1k})\right)_{i=1}^n + \widetilde{\boldsymbol{C}}_{k-1},\quad \boldsymbol{C}_1^{(k)} = \mathbf{1} + \widetilde{\boldsymbol{C}}_1^{(k)}$$

where $\circ$ is elementwise multiplication. MATLAB=.*

Using this to avoid cancellation error

$$1-\frac{n}{\lambda_1} = 1-\frac{n}{n+\tilde{\lambda}_1} = \boxed{\frac{\tilde{\lambda}_1}{n+\tilde{\lambda}_1}}$$

Periodization transforms

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

C0 $: \tilde{f}(\boldsymbol{t}) = f\left(\bar{g}_0(\boldsymbol{t})\right) \prod_{j=1}^{d} g_0'(t_j), \quad g_0(t) = 3t^2 - 2t^3, \quad g_0'(t) = 6t(1-t))$

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

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

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

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

Sidi's C2 $: \tilde{f}(\boldsymbol{t}) = f\left(\bar{\psi}_3(\boldsymbol{t})\right) \prod_{j=1}^{d} \psi_3'(t_j), \quad \psi_3(t) = \frac{1}{16}\left(8 - 9\cos(\pi t) + \cos(3\pi t)\right),$

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

Multivariate normal probability

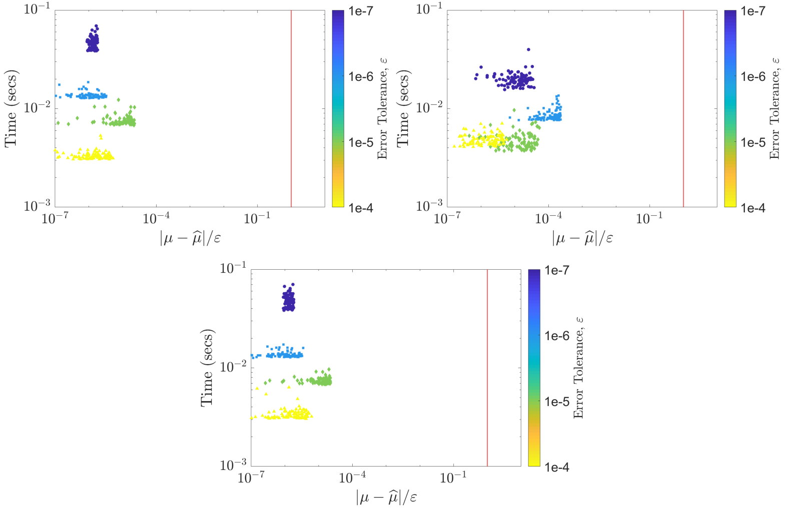


Figure: Multivariate normal probability example using 1) Empirical Bayes, 2) GCV, 3) Full Bayes stopping criterion

Keister Integral with arb mean m

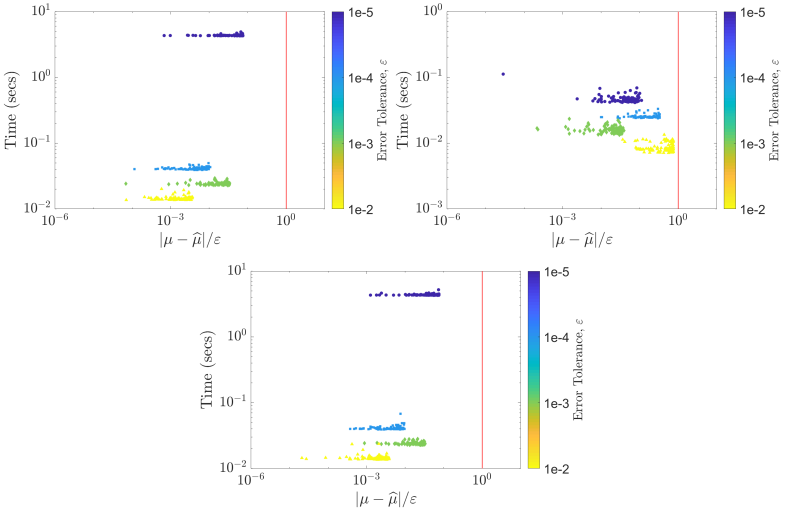


Figure: Integrating Keister function using 1) Empirical Bayes, 2) GCV, 3) Full Bayes stopping criterion

Option pricing

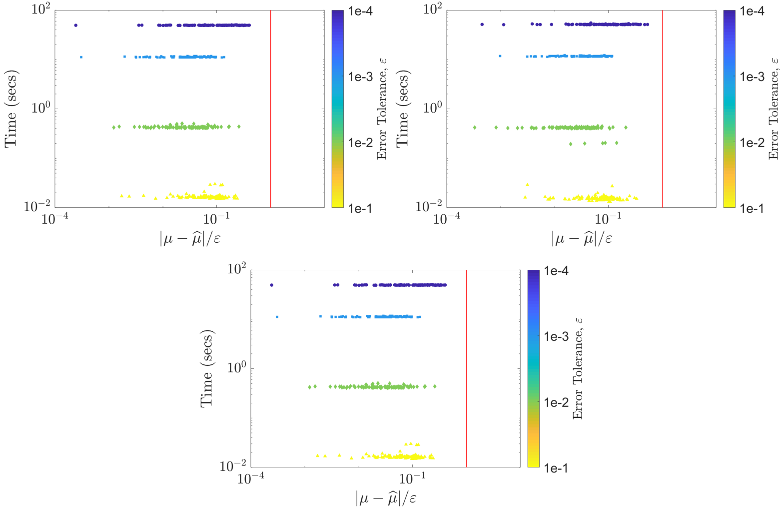


Figure: Option pricing using 1) Empirical Bayes, 2) GCV, 3) Full Bayes stopping criterion

Summary

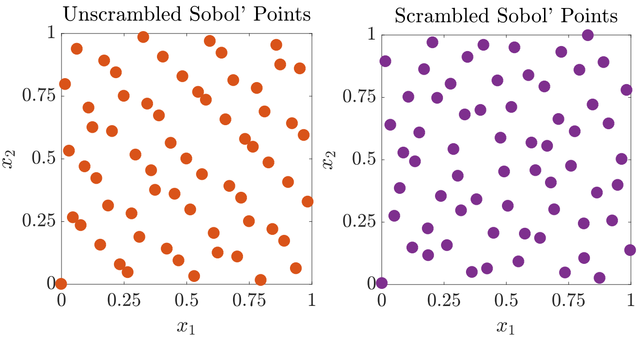
- Developed a *general technique* for a Fast transform kernel
- Developed a fast automatic Bayesian cubature with $\mathcal{O}(n \log n)$ complexity
- Having the advantages of a kernel method and the low computation cost of Quasi Monte carlo
- Scalable based on the complexity of the Integrand
  i.e, Kernel order and Lattice-points can be chosen to suit the smoothness of the integrand
- Conditioning problem if the kernel C is very smooth
- Source code : `https://github.com/GailGithub/GAIL_Dev/tree/feature/BayesianCubature`
- More about Guaranteed Automatic Algorithms (GAIL): `http://gailgithub.github.io/GAIL_Dev/`

Future work

- Choosing the kernel order r and periodization transform automatically
- Deterministic interpretation of Bayesian cubature
- Broaden the choice of numerical examples
- Better handling of conditioning problem and numerical errors
- Sobol pointset and Fast Walsh Transform with smooth kernels ...

Future work : Sobol points and Fast walsh transform

- Using the established generalized theory for a *Fast transform kernel*, we could use other kernels with suitable point sets to achieve similar or better performance and accuracy.
- One such point sets to consider in future is, *Sobol points* and with appropriate choice of smooth kernel, should lead to *Fast Walsh Transform*.



Future work : More applications

- Control variates : We would like to approximate a function of the form $(f - \beta_1 g_1 -, ..., -\beta_p g_p)$, then

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

- Function approximation : consider approximating a function of the form

$$\int_{[0,1]^d} \underbrace{f(\boldsymbol{\phi}(\boldsymbol{t})).\left|\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{t}}\right|}_{g(\boldsymbol{t})} \mathrm{d}\boldsymbol{t}, \quad \text{where } \left|\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{t}}\right| \text{ is Jacobian, then}$$

$$g(\boldsymbol{\psi}(\boldsymbol{x})) = f(\underbrace{\boldsymbol{\phi}(\boldsymbol{\psi}(\boldsymbol{x}))}_{\boldsymbol{x}}).\left|\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{t}}\right|(\boldsymbol{\psi}(\boldsymbol{x})), \quad f(\boldsymbol{x}) = g(\boldsymbol{\psi}(\boldsymbol{x})).\frac{1}{\left|\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{t}}\right|(\boldsymbol{\psi}(\boldsymbol{x}))}$$

Finally, the function approximation is

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

Thank you!

References I

Diaconis, P. 1988. *Bayesian numerical analysis*, Statistical decision theory and related topics iv, papers from the 4th purdue symp., west lafayette/indiana 1986, pp. 163–175.

Dick, J. and F. Pillichshammer. 2010. *Digital nets and sequences: Discrepancy theory and quasi-Monte Carlo integration*, Cambridge University Press, Cambridge.

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

Glasserman, P. 2004. *Monte Carlo methods in financial engineering*, Applications of Mathematics, vol. 53, Springer-Verlag, New York.

Hickernell, F. J. and H. Niederreiter. 2003. *The existence of good extensible rank-1 lattices*, J. Complexity **19**, 286–300.

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

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

Olver, F. W. J., D. W. Lozier, R. F. Boisvert, C. W. Clark, and A. B. O. Dalhuis. 2013. *Digital library of mathematical functions*.

Rasmussen, C. E. 2003. *Bayesian Monte Carlo*, Advances in Neural Information Processing Systems, pp. 489–496.

References II

Ritter, K. 2000. *Average-case analysis of numerical problems*, Lecture Notes in Mathematics, vol. 1733, Springer-Verlag, Berlin.