Probabilistic Quadrature Rules

Shashank Sule

December 2019

1 Introduction

Suppose we want to find the integral

$$I = \int_{[0,1]} f(x)\, dx$$

but we cannot calculate it analytically. Such problems of finding complicated inetgrals are ubiquitous in almost every quantitative field. For example, a physicist might be able to measure the velocity field and want to calculate the flux out of a surface using an integral. On the other hand, a statistician might want to calculate the expectation of a random variable in a statistical model of some data. The inability to make analytical calculations calls for numerical methods for computing integrals. The problem of computing an integral numerically is termed "Quadrature" and the various algorithms used for quadrature are termed "Quadrature rules". There are two types of quadrature rules: deterministic and probabilistic. Deterministic quadrature rules constitute the classical approach to quadrature, dating back to at least the days of Newton. The deterministic approach essentially constitutes of approximating the integral as a weighted sum of function evaluations at finitely many points:

$$\int_X f(x)\, dx \approx \sum_{i=1}^{N} w_i f(x_i)$$

Here the weights w_i are typically non-negative and $x_i \in X$. Note that the intution for the weighted sum arises from the integral as a limit of a Riemann sum. The quadrature rule essentially computes a Riemann sum and as $N \to \infty$ the approximation converges to the actual value of the integral. Under the **Newton-Cotes** rules, the points x_i are equally spaced. Another way to derive quadrature rules is to assume that f is contained in a finite dimensional subspace of a suitable vector space of functions. If we know the integral of the basis elements and the coefficients of f in the basis, then we can recover integral which is a linear functional on f. How do we recover the coefficients? If the chosen basis is orthonormal, then the coefficients are the inner products of f with each basis element. This approach leads to the set of **Gauss** quadrature rules, which pick orthogonal polynomials as a finite basis and pick the roots of the polynomial as the sample points.

Deterministic quadrature rules are desirable for a number of reasons. First, they are reasonably inexpensive to compute, especially under the Gaussian scheme. Second, they converge algebraically ($\mathcal{O}(N^{-p})$ to the desired answer where N is the number of sample points. Furthermore, deterministic quadrature rules can be tailored to specific applications because of the freedom to choose sample points and basis functions.

On the other hand, deterministic rules are often not ideal for integrating in higher dimensions. For example, the 2-order Newton-Cotes rule (i.e Simpson's rule) has a convergence rate of $\mathcal{O}(N^{-4/n})$ where n is the dimension of the ambient space in which the integrating domain lies. This means that as the dimension gets larger, the approximation converges more slowly to the actual value, thus necessitating a higher number of samples. Sampling a function is often quite expensive in many applications and so there arises a need for dimension-independent quadrature rules.

Probabilistic quadrature rules fix this problem. The main strategy in the probablistic approach is to think of the integral as an expectation of a function of a random variable and devise an estimator for the expectation. For example the integral in (1) can be understood as the expectation of $f(X)$ where $X \sim \text{Uniform}(0,1)$. Indeed,

$$\mathbb{E}[f(X)] = \int_{[0,1]} f(x)\chi_{[0,1]}\, dx = \int_{[0,1]} f(x)\, dx$$

.

A straightforward estimator for the expectation is the following random variable:

$$\mathbb{E}[f(X)] \approx \langle f \rangle = \frac{1}{N}\sum_{i=1}^{N} f(X_i)$$

where X_i are i.i.d random variables. What follows is a discussion on the particular implementations of the above estimator in standard algorithms, which are collectively term. In the second part of this note, we will compare these algorithms to the Bayesian approach which proposes a different estimator than $\langle f \rangle$.

2 Monte Carlo Integration

2.1 Simple MC

We demonstrate the Simple Monte Carlo approach by finding the volume of the D dimensional 1-ball, denoted $B_1^D(0)$. Note that we can express the volume as an integral:

$$V(B_1^D(0)) = \int_{[-1,1]^D} \chi_{||x||<1}\, dx$$

Using the above equation, if we consider $f = \chi_{||x||<1}$, then the Monte Carlo estimator can be applied to the above function with $\text{Uni}([-1,1]^D)$ as the underlying random variable.

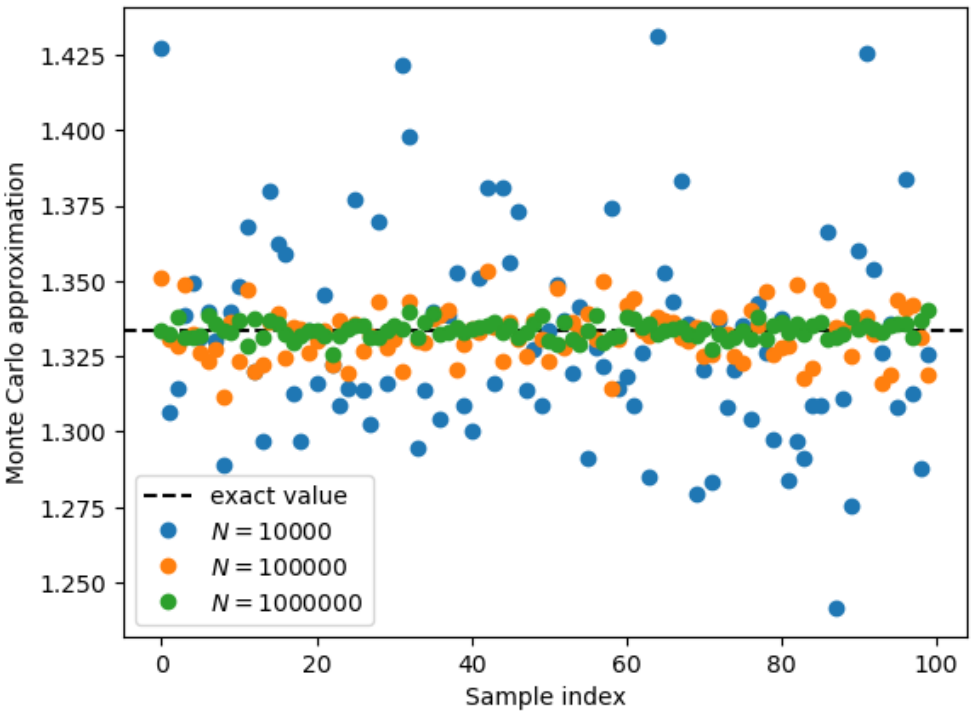

Figure 1: Using Simple Monte Carlo to estimate the size of the unit L^1 ball

As seen in Figure 1, the Monte Carlo approach estimates the true volume of the 3 dimensional unit L^1 ball more precisely with more samples. We can quantify the precision, or error, using the variance of $\langle f \rangle$ because

$$\text{Var}(\langle f \rangle) = \text{Var}(\frac{1}{N}\sum_{i=1}^{N} f(X_i)) = \frac{1}{N^2}\sum_{i=1}^{N}\text{Var}(f(X_i)) = \frac{1}{N^2}N\text{Var}(f(X))\frac{\text{Var}(f(X))}{N} \approx \mathcal{O}(N^{-1})$$

Note that the variance here is independent of dimension because in general the variance of X may not scale with dimension. In any case, the order of convergence is (or the exponential factor on the number of samples) for the variance is 1 (for standard deviation it is $1/2$), so while deterministic rules may be better for low dimensions, the Monte Carlo rule is certainly favourable for higher dimensions. However, note that Simple Monte Carlo still requires quite a large number of samples to be accurate up to 4 orders of precision. To improve the error, we implement sampling from a non-uniform distribution in an algorithm called importance sampling.

2.2 Importance Sampling

Consider the following integral:

$$I = \int_{[0,1]} 2(1-x)e^x \, dx$$

The Simple Monte Carlo algorithm produces the following results for this integral:

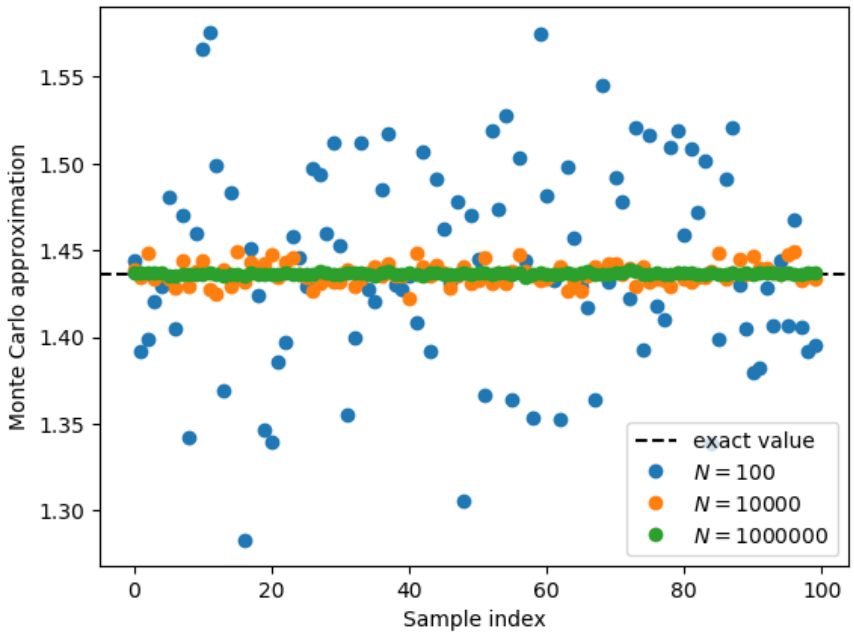

Figure 2: Simple Monte Carlo for $I = \int_{[0,1]} 2(1-x)e^x \, dx$

Note that the $N = 100$ case is especially poor. We can actually improve the standard deviation using a tweak in the underlying distribution of X. Simple Monte Carlo samples x the uniform distribution and then computes the value of $f(x)$. However, we could have interpreted this integral as

$$I = \mathbb{E}(f(X)) = \int_{[0,1]} e^x(2(1-x)) \, dx = \int_{[0,1]} f(x)p(x) \, dx$$

where $f(x) = e^x$ and X such that $f_X(x) = 2(1-x)$. Obsserving that $f_X(x)$ satisfies the criteria for a density function, we may now sample X from this distribution and compute $f(X)$. Comparing with Simple Monte Carlo we obtain the following results:

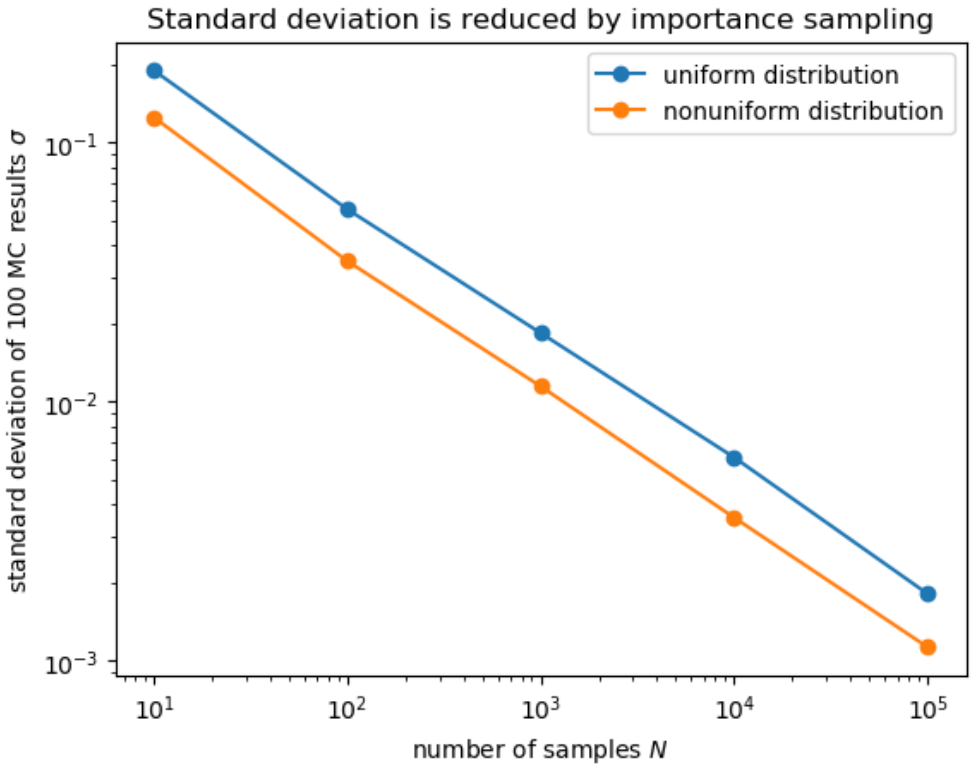

Figure 3: Simple Monte Carlo vs Importance Sampling

The above algorithm is termed importance sampling and from the example it is clear that it improves the precision of our results slightly. This is because the variance of $\langle f \rangle$ relies on the variance of X so by tweaking the distribution of X we can, in some cases, improve its variance and hence improve the variance of $\langle f \rangle$.

2.3 Recursive Stratified Sampling

The Recursive Stratified Sampling method improves on the variance in the Monte Carlo method by implementing the Monte Carlo algorithm recursively and by using the parallel axis theorem. Suppose the domain D of integration is partitioned into two domains, A and B of equal volume. Then we have the following estimator for the expectation of $f(X)$ where X is distributed uniformly on D:

$$\begin{aligned}\mathbb{E}[f(X)] &= \frac{1}{V(D)} \int_D f(x)\, dx \\ &\approx \frac{1}{2}\Big(\frac{1}{N/2} \sum_{i=1}^{N/2} f_A(X_i) + \frac{1}{N/2} \sum_{i=1}^{N/2} f_B(X_i)\Big) = \langle f \rangle'\end{aligned}$$

Here f_A and f_B are restrictions of f to A and B respectively. Note that the estimator $\langle f \rangle'$ can be viewed as a restricted case of the standard Monte Carlo estimator $\langle f \rangle$ where

half the points are sampled independently from A and the other half is sampled independently from B. Its variance is as follows

$$\sigma^2(\langle f\rangle') = \sigma^2\Big(\frac{1}{2}\frac{1}{N/2}\sum^{N/2} f_A + \frac{1}{2}\frac{1}{N/2}\sum^{N/2} f_A\Big) = \frac{1}{4}\Big(\sigma^2\Big(\frac{1}{N/2}\sum^{N/2} f_A\Big) + \sigma^2\Big(\frac{1}{N/2}\sum^{N/2} f_B\Big)\Big)$$

$$= \frac{1}{4}\Big(\frac{4}{N^2}\frac{N}{2}(\sigma^2(f_A) + \sigma^2(f_B))\Big) = \frac{1}{2N}\Big(\sigma^2(f_A) + \sigma^2(f_B)\Big)$$

On the other hand, the parallel axis theorem for $\mathrm{Var}(f(X)) = \sigma^2(f(X))$ states that

$$\sigma^2(f) = \frac{1}{2}\Big(\sigma^2(f_A) + \sigma^2(f_A)\Big) + \Big(\frac{\mathbb{E}[f_A(X)] - \mathbb{E}[f_B(X)]}{2}\Big)^2$$

As a consequence, we have the following inequality relating $\sigma^2(\langle f\rangle)$ and $\sigma^2(\langle f\rangle')$:

$$\begin{aligned}
\sigma^2(\langle f\rangle) &= \frac{1}{N}\sigma^2(f) \\
&= \frac{1}{N}\frac{1}{2}\Big(\sigma^2(f_A) + \sigma^2(f_A)\Big) + \frac{1}{N}\Big(\frac{\mathbb{E}[f_A(X)] - \mathbb{E}[f_B(X)]}{2}\Big)^2 \\
&= \sigma^2(\langle f\rangle') + \frac{1}{N}\Big(\frac{\mathbb{E}[f_A(X)] - \mathbb{E}[f_B(X)]}{2}\Big)^2 \\
&\geq \sigma^2(\langle f\rangle')
\end{aligned}$$

Thus, $\sigma^2(\langle f\rangle) \geq \sigma^2(\langle f\rangle')$ and equality holds if and only if the average values of f on A and B are identical. Magically, we have improved our variance by considering a different estimator by subdividing the region. The algorithm, then, can be set up in the following way:

1. Fix a number of function evaluations, N.
2. Keep subdividing the domain until the number of samples on each subdomain is reduced to under a threshold (say 100)
3. Compute the integral on the minimal subbdomains using simple monte carlo.
4. Compute the integral on a subdomain using the estimator for its two subdomains.

Step number 4 is the recursive step. Step number 2 is the step where we "stratify" the domain into subdomains; hence the name, Recursive Stratified Sampling. Let's use RSS to compute the volume of the L^1 unit ball in 3 dimensions. The details on the implementation and the proof of the parallel axis theorem are spelled out in the appendix.

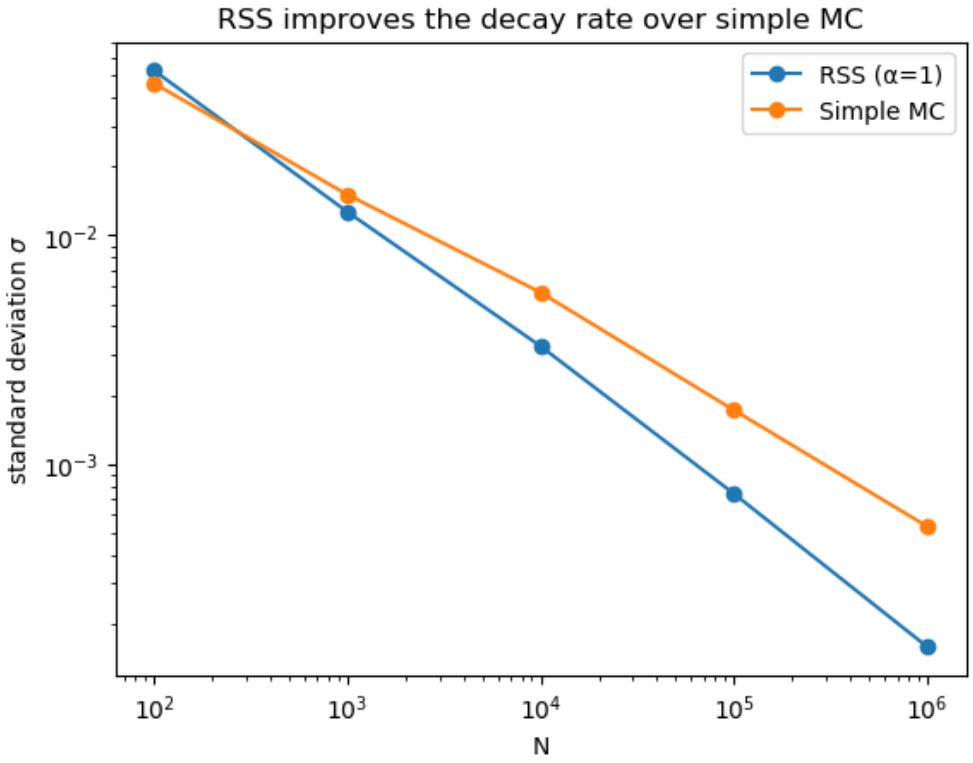

Figure 4: Comparing Recursive Stratified Sampling with Simple Monte Carlo

Thus far, we have seen that improving the Monte Carlo estimator using importance and recursive stratified sampling strategies yields slight improvements in the variance of the integral estimator. While importance sampling maintains the $1/2$ rate of convergence, picking a different distribution function improves the variance by a constant. On the other hand, the recursive method improves the rate of convergence so it is the more desirable method when the function is cheap to sample. But despite these improvements, we still need about 1 million function samples to get to within 4 digits of the true answer. The Bayesian estimator solves this accuracy problem dramatically.

3 Bayesian Quadrature

The Bayesian approach interprets quadrature as a statistical problem. We are given some function f and we can only know its values on finitely many points. Given this sample of values, we must infer something about a statistic I, which depends only on f. So computing the integral is really just an inference problem, where we must infer the value of a statistic using some data which represents an underlying model. Of course, we don't know f, so there is some level of uncertainty about $f(x)$, the value f takes at x. But we can express some prior beliefs about f and then use the data we collect to sharpen our beliefs. But how do we compute the statistic, $\mathbb{E}[f(X)]$? We need an estimator for it! We make the weak assumption that f is continuous. Suppose we give a prior for f, termed $\mathcal{F}$ whose density is denoted $\pi(f)$ (thus $\mathcal{F}$ is a prior on continuous functions). Suppose we

sample the function at n points and represent the sampled values as the vector and obtain the posterior distribution $\mathcal{F} \mid$. Then $\mathbb{E}_X[f]$ is a function of the random variable $\mathcal{F} \mid \mathbf{f}$, denoted $\mathbb{E}[\mathcal{F} \mid \mathbf{f}]$. Clearly, the first moment of $\mathbb{E}_X[\mathcal{F} \mid \mathbf{f}]$, i.e the expectation over $\mathcal{F} \mid \mathbf{f}$, is a good estimator for this random variable. The following computation Ghahramani and Rasmussen [2003] hints at how we might actually able to compute this estimator:

$$\begin{aligned} I \approx \langle f \rangle_B = \mathbb{E}_{\mathcal{F}|\mathbf{f}}(\mathbb{E}[f]) &= \int_{C[0,1]} \int_{[0,1]} f(x)p(x)\,dx\,\pi(f \mid \mathbf{f})\,df \\ &= \int_{[0,1]} \Big(\int_{C[0,1]} f(x)\pi(f \mid \mathbf{f})\,df \Big) p(x)\,dx \\ &= \int_{[0,1]} \overline{f}(x)p(x)\,dx \\ &= \mathbb{E}_X[\overline{f}(X)] \end{aligned}$$

Here $\overline{f}(x)$ is the posterior mean of $\mathcal{F} \mid \mathbf{f}$ (since $\mathcal{F} \mid \mathbf{f}$ is a distribution on continuous functions, its mean is a continuous function). From the above computation, it is clear that the estimator is the expectation of the posterior mean over X. So if the prior-likelihood model lends itself to an easy computation of the posterior mean, then the estimator can actually be calculated, leading to another algorithm for computing I. One such prior-likelihood model is called the **Gaussian process**.

3.1 Gaussian processes

The Gaussian process (G.P) model posits that since $f(x)$ is uncertain, it needs to be modeled by a random variable. The most natural choice is to assume that $f(x) \sim N(\mu(x), \sigma(x))$. Secondly, given two distinct points x and y, the correlation between $f(x)$ and $f(y)$ is higher if x and y are closer. Then $f(x)$ and $f(y)$ can have a joint normal distribution where the covriance matrix approaches the identity matrix as x and y get closer (i.e the covariance converges to 0). The two-point case can be suitably generalized. In other words, given $x_1, \ldots, x_n$ (denoted as the vector), the values $f(x_1), \ldots, f(x_n)$ (denoted as the vector) follow a joint normal distribution, with mean $\mu(\mathbf{x})$ and an $n \times n$ covariance matrix $K(\mathbf{x})$, where the entries, termed $k(x_i, x_j)$ satisfy the following properties:

1. Symmetry: $k(x_i, x_j) = k(x_j, x_i)$
2. Positivity: $k(x_i, x_j) > 0$
3. 1-Definiteness: $k(x_i, x_1) = 1$

These conditions ensure that $K(\mathbf{x})$ is indeed symmetric and positive-definite, thus satisfying the conditions for a covariance matrix. In summary, a distribution on continuous functions termed $\mathcal{F} : C[0,1] \mapsto \mathbb{R}$ wherein $\mathbf{f} \mid \mathcal{F} = f$ follows a joint normal distribution with a covariance matrix having the above properties, is termed a **Gaussian Process**. The following figure illustrates what a Gaussian process with a constant variance "looks like":

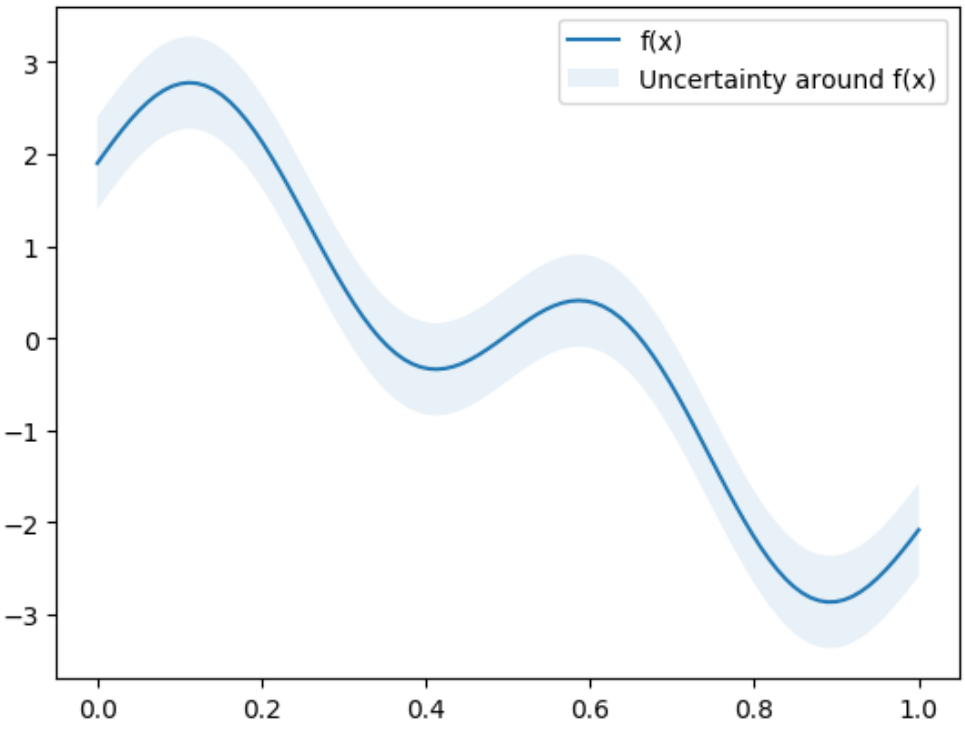

Figure 5: A Gaussian Process with constant variance

Suppose $\mathcal{F} \sim GP$. The following two theorems give a grip on the posterior distribution $\mathcal{F} \mid \mathbf{f}$:

Theorem 3.1 (Gortler et al. [2019]). *Let $f \sim GP$ and $\boldsymbol{f}$ be a set of finite evaluations of f on the set $\boldsymbol{x}$. Then $f \mid \boldsymbol{f} \sim GP$*

The proof follows from the fact that the conditional distribution of a joint normal distribution is also joint normal.

Theorem 3.2 (Minka [2000]). *Fix $x \in [0,1]$. Then*

$$f(x) \mid \boldsymbol{f} \sim N(k(x,D)k(D,D)^{-1}f(D), k(x,x) - k(x,D)k(D,D)^{-1}k(D,x))$$

Thus, the posterior mean of $\mathcal{F} \mid \mathbf{f}$ is given by $m(x) = k(x,\mathbf{x})K(\mathbf{x})^{-1}\mathbf{x}$.

From the computation earlier in the section, the estimator for the integral is given by integrating m against p:

$$\begin{aligned} I &\approx \mathbb{E}_X[\overline{f}(X)] \\ &= \int_{[0,1]} m(x)\mu(x)\,dx \\ &= (u(\mathbf{x}))^\top K(\mathbf{x})^{-1}\mathbf{f} \end{aligned}$$

where $(u(\mathbf{x}))^\top = \int_{[0,1]} k(x,\mathbf{x})\mu(x)\,dx$

This is the algorithm!

3.2 A Simple Bayesian Quadrature Algorithm

To compute the Bayesian estimator for I, we need to define the covariance function and ensure that it can be integrated analytically. In the following implementation, I pick k to be the **Lorentzian Kernel**, where

$$k(x, y) = \frac{1}{1 + |x - y|^2}$$

Note that k satisfies all the three properties of a covariance function. Furthermore, observe that the vector $u(\mathbf{x})$ can be computed analytically when $\mu(x) \equiv 1$. The ith entry in $u(\mathbf{x})$ is as follows:

$$\begin{aligned}\int_{[0,1]} k(x, x_i)\mu(x)\, dx &= \int_{[0,1]} \frac{1}{1 + |x - x_i|^2}\, dx \\ &= \arctan(1 - x_i) - \arctan(-x_i) \\ &= \arctan(1 - x_i) + \arctan(x_i)\end{aligned}$$

Then the algorithm is as follows:

1. Sample N points uniformly in [0,1], termed $\mathbf{x}$.
2. Compute f at each point in $\mathbf{x}$ and term the vector of evaluations $\mathbf{f}$.
3. Compute $(u(\mathbf{x}))^\top$ using the above formula.
4. Compute $(u(\mathbf{x}))^\top K(\mathbf{x})^{-1}\mathbf{f}$

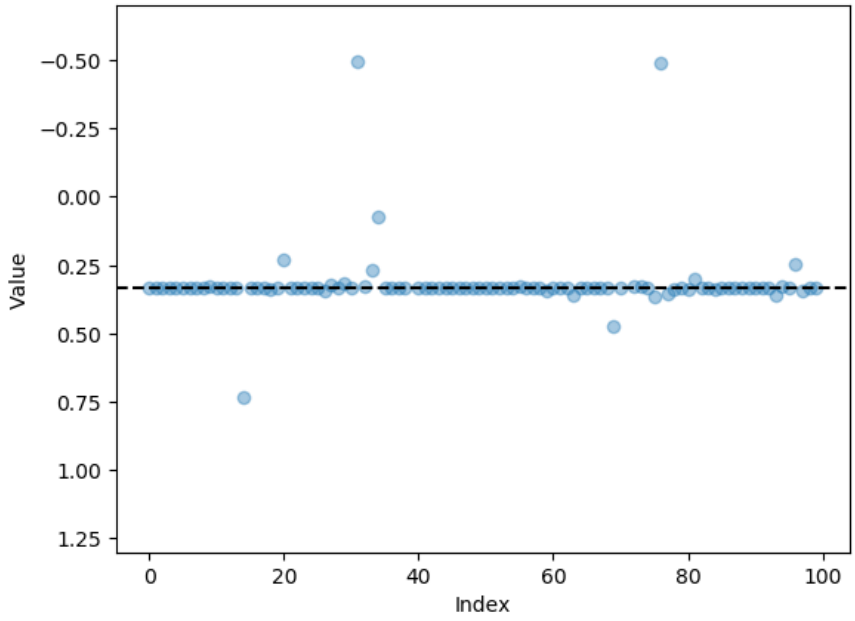

Figure 6: Bayesian Quadrature for computing $I = \int_0^1 x^2\, dx$

Bayesian Quadrature is beautifully accurate only with 10 samples. A comparison with Simple Monte Carlo reveals its stunning accuracy:

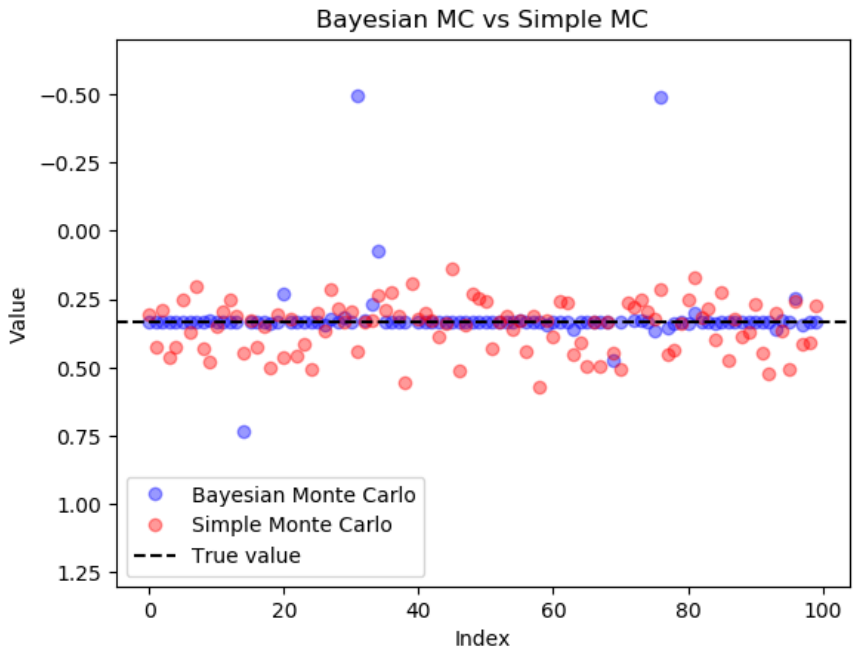

Figure 7: Comparing Bayesian Quadrature with Simple Monte Carlo

The formula given in Minka [2000] reveals that the estimator variance does not depend on **f**:

$$\sigma^2(\langle f \rangle_B) = u - (u(\mathbf{x}))^\top K(\mathbf{x})^{-1} u(\mathbf{x})$$

where $u = \int_{[0,1]} \int_{[0,1]} k(x,y)\mu(x)\mu(y)\,dx\,dy$

Thus, we may pick $\mathbf{x}$ to minimze variance. In fact, in many special cases, the variance minimizing sample points are actually the sample points found in Newton-Cotes rules. Diaconis provides an excellent summary of this method in Diaconis [1988]. However, in this article, I keep the algorithm probablistic so I sample $\mathbf{x}$ randomly from the uniform distribution.

Although the Simple Bayesian Quadrature method outperforms Monte Carlo, it is not quite scalable in dimension or sample size. This is because of two reasons: first of all, matrix inversion is a poorly conditioned problem even when done through spectral decomposition. Secondly, the rule does not scale appropriately with dimension because the size of the matrix itself grows with dimension. Most practical implementations of Bayesian quadrature are thus determinsitic, but it can be viewed as an algorithm that amalgamates both determinstic and probabilistic techniques to create efficient and accurate results.

4 Conclusion

A word of caution: there is no one algorithm that is best for all integrals. Most of the implementations discussed in the article were carefully chosen to illustrate certain advantages of the methods being used. That was to demonstrate their unique qualities and failures; but it does not mean, for example, that Monte Carlo integration is always bad and Bayesian quadrature is always good. Indeed, Monte Carlo integration was motivated as a way to solve the curse of dimensionality that blighted deterministic methods. The importance sampling and RSS algorithms were shown to be more accuracy-efficient implementations of the Monte Carlo estimator, yet none were as accurate as Bayesian Quadrature. But the narrative goes back full circle as Bayesian Quadrature once again suffers from the curse of dimensionality, the very problem that motivates Probablistic quadrature in general.

References

Zoubin Ghahramani and Carl E. Rasmussen. Bayesian monte carlo. pages 505–512, 2003. URL http://papers.nips.cc/paper/2150-bayesian-monte-carlo.pdf.

Jochen Gortler, Rebecca Kehlbeck, and Oliver Deussen. A visual exploration of gaussian processes. *Distill*, 2019. doi: 10.23915/distill.00017. https://distill.pub/2019/visual-exploration-gaussian-processes.

Thomas P Minka. Deriving quadrature rules from gaussian processes. Technical report, 2000.

Persi Diaconis. Bayesian numerical analysis. *Statistical decision theory and related topics IV*, 1:163–175, 1988.