Multi-armed bandit for stratified sampling: Application to numerical integration

Florian Leprêtre, Fabien Teytaud, Julien Dehos

Multi-armed bandit for stratified sampling: Application to numerical integration

Florian Leprêtre
LISIC
Université du Littoral Côte d'Opale
florian.lprt@gmail.com

Fabien Teytaud
LISIC
Université du Littoral Côte d'Opale
fabien.teytaud@univ-littoral.fr

Julien Dehos
LISIC
Université du Littoral Côte d'Opale
julien.dehos@univ-littoral.fr

***Abstract*—Contextual multi-armed bandits model decision problems, where the properties of the possible decisions are initially partially known, but may become better known as time passes. Such models have numerous applications and many algorithms have been proposed to provide approximate solutions.**

In this paper, we propose an algorithm for computing multidimensional integration problems. Such problems are very common and can be solved using the Monte-Carlo method with the stratified sampling technique. This method consists in partitioning the integration domain then randomly sampling the partitions. Our algorithm considers the selection of the best partition to sample as a multi-armed bandit problem, which can be solved using the Upper Confidence Bound technique. We have experimented this approach for several integration problems and observed faster convergence rates.

***Index Terms*—numerical integration, stratified sampling, machine learning, UCB**

I. Introduction

Computing an integral numerically is a common technique when the analytical result is not known. This oftenly occurs in some fields (computational physics, computational finance, image synthesis...) where efficient numerical integration methods are needed. In one dimension, algorithms based on numerical quadrature are generally satisfactory and standard implementations (for instance QUADPACK [1]) are available. However, the situation is less satisfactory for multidimensional integrals. Indeed, the number of function evaluations required by the quadrature methods, grows exponentially with the dimension leading to the so-called curse of dimensionality [2].

Several methods have been proposed for computing multidimensional integrals [3]. The cubature method is an extension of the quadrature method to high dimensions. It is often implemented with a deterministic algorithm using an adaptive subdivision strategy (DCUHRE algorithms [4]). The sparse grid method [5], [6] is also based on quadrature formulas but uses a linear combination of tensor products (the Smolyak algorithm [7]). These deterministic methods can give good results but are generally limited to a moderate number of dimensions or smooth integrands.

The Monte-Carlo method is a classic numerical integration method for high dimensions. Indeed, the method is based on function evaluations at random samples and does not depend on the dimension [8]. However, its convergence rate is slow so many variance-reduction techniques have been proposed to improve the method, for instance importance sampling and stratified sampling [9]. The idea of importance sampling is to choose a sampling distribution which encourages the interesting regions of the integrand. The sampling distribution greatly impacts the efficiency of the technique; it can be constructed iteratively, as in the VEGAS algorithm [2]. The idea of stratified sampling is to subdivide the integration domain and then to apply the Monte-Carlo method on each subdivision. Here the subdivision scheme is important; it can be constructed recursively as in the MISER algorithm [10].

Finally, the Bayesian Quadrature method [11], [12] is a model-based approach for computing multidimensional integration. Here, the integrand is modeled as a Gaussian Process; a posterior distribution is inferred from prior samples and the integral is approximated as the mean of this distribution.

In this paper, we propose an improvement of the stratified sampling technique for the Monte-Carlo method. Our method uses a reinforcement learning technique to successively select the most interesting subdivision to sample (i.e. a multi-armed bandit problem). We compare our algorithm to other classic algorithms on numerous testing integrands, including the classic TESTPACK integrands [13].

II. Background

A. Monte-Carlo integration

The Monte-Carlo method is commonly used for computing complex integration problems [9], [14]. Such problems can be defined by:

$$I = \int_{\Omega} f(\mathbf{x}) \mathrm{d}\mu(\mathbf{x}) \tag{1}$$

where Ω is the integration domain (monodimensional or multidimensional), $f : \Omega \to \mathbb{R}$ is the function to integrate and μ is a measure function on Ω.

The Monte-Carlo method uses random sampling to estimate the value of the integral. In its simpliest form (where the domain Ω is the unit hypercube and the random samples $\mathbf{X}_k$ are chosen uniformly and independently), the Monte-Carlo estimator can be written as:

$$F = \frac{1}{K} \sum_{k=1}^{K} f(\mathbf{X}_k). \tag{2}$$

The estimator F converges to I at a rate of $\mathcal{O}(K^{-1/2})$; this can be shown by computing the expected value of F and the evolution of the standard deviation with K.

B. Stratified sampling

Stratified sampling is a variance reduction technique which can improve the convergence rate of the Monte-Carlo method. The basic idea is to partition the integration domain in *strata* Ω_n:

$$\Omega = \bigcup_{n=1}^{N} \Omega_n, \tag{3}$$

and then to consider a given number of samples K_n in each stratum Ω_n, with:

$$K = \sum_{n=1}^{N} K_n. \tag{4}$$

The corresponding estimator is:

$$F' = \sum_{n=1}^{N} v_n \frac{1}{K_n} \sum_{k=1}^{K_n} f(\mathbf{X}_{n,k}), \tag{5}$$

where $v_n = \mu(\Omega_n)$ is the volume of the stratum Ω_n. The variance of F' is then:

$$\mathrm{Var}(F') = \sum_{n=1}^{N} \frac{v_n^2 \sigma_n^2}{K_n}, \tag{6}$$

where $\sigma_n^2 = \mathrm{Var}(f(\mathbf{X}_{n,k}))$. Assuming (for simplification) that $K_n = v_n K$, then the variance of the stratified estimator is:

$$\mathrm{Var}(F') = \frac{1}{K} \sum_{n=1}^{N} v_n \sigma_n^2. \tag{7}$$

It can be shown that the variance of the unstratified estimator is [14]:

$$\mathrm{Var}(F) = \frac{1}{K} \left[\sum_{n=1}^{N} v_n \sigma_n^2 + \sum_{n=1}^{N} v_n (\mu_n - I)^2 \right], \tag{8}$$

where μ_n is the mean value of f in the stratum Ω_n. This implies that the variance of the stratified estimator is always less than or equal to the variance of the unstratified estimator, i.e. $\mathrm{Var}(F') \leq \mathrm{Var}(F)$. However, the effective reduction of variance depends on the partition of the integration domain and on the number of samples chosen in each stratum.

C. Multi-Armed Bandit

One historical paradigm for online learning is defined as the multi-armed bandit problem [15]. This problem involves N bandit arms with unknown reward probabilities p_n. At each step a player or a program selects an arm and receives a new reward r_k where $r_k = 1$ with probability p_n and 0 otherwise. The goal is to maximize the cumulated reward gathered over all time steps K. This is equivalent to minimize the regret, defined as the loss incured compared to the best arm. The regret is defined as:

$$K p^* - \sum_{k=1}^{K} r_k, \tag{9}$$

where p^* is the maximal reward probability among $p_1, \ldots, p_N$.

One recent way for dealing with this problem is the Upper Bound Confidence (UCB) algorithm [16]. This algorithm provides an optimal asymptotic bound on the regret in $\mathcal{O}(\ln(K))$. At each step k, the algorithm selects an arm n which maximizes the formula:

$$\hat{p}_{j,k} + \sqrt{\frac{2 \ln \left(\sum_{n=1}^{N} K_{n,k} \right)}{K_{j,k}}}, \tag{10}$$

where $\hat{p}_{j,k}$ is the average reward for the arm j and $K_{n,k}$ is the number of times the arm n has already been selected.

This formula is a trade-off between *exploitation* and *exploration*. The first part of the formula corresponds to the exploitation part as it tends to select the arm with the optimal average reward. The second part of the formula corresponds to the exploration part as it tends to select the arm which has been selected the most rarely.

III. Proposed method

In this paper, we propose a multidimensional integration method, called UCBATURE, which combines the stratified sampling technique and the UCB algorithm, introduced in section II. The UCBATURE function is shown in Algorithm 1 and explained in this section.

A. Preliminary description

As the stratified sampling technique, our algorithm first divides the integration domain Ω into *regions* (the strata partitioning defined in equation 3). If the integrand is sufficiently "complex", some regions of the integration domain may be more interesting to sample. This can be seen as a multi-armed bandit problem: each region Ω_n is an arm and we have to decide which arm to select (i.e. which region to sample).

Thus, rather than taking the same number of samples in every partitions (as stratified sampling does), we can implement an UCB strategy to distribute the samples in a cleverer way, using an online learning process. The algorithm will then tend to concentrate function evaluations in interesting regions, where the integrand varies the most or has the largest values.

B. Initializing with stratified sampling

The first step of the proposed method is the initialization of the regions (lines 1 to 10 of Algorithm 1). Any set of non-overlapping regions satisfying equation 3 can be used to partition the integration domain. To simplify implementation, we use a regular grid (i.e. all cells have the same volume), but this is not a restriction of the method. If we split the integration domain into N_0 parts for each of the D dimensions, we get $N = N_0^D$ regions.

Then, the algorithm computes $K_{n,0}$ initial random samples in each region n to approximate the mean value and the variance of the regions. This can be seen as a stratified sampling step, giving a rough estimation of the relevance of every regions, in order to initialize the next step of the algorithm (online learning).

Algorithm 1: Ucbature

Input:
N: number of regions
Ω_n: region of index n
v_n: volume of the region Ω_n
$K_{n,0}$: number of initial samples for Ω_n
K: total number of samples
f: integrand
Output:
F': estimation of the integral of f

1 **begin**
2 {initialization using stratified sampling}
3 **for** $n \leftarrow 1$ **to** N **do**
4 **for** $k \leftarrow 1$ **to** $K_{n,0}$ **do**
5 sample $\mathbf{x}$ in Ω_n
6 $\mathbf{y} \leftarrow f(\mathbf{x})$
7 $S_n \leftarrow S_n + \mathbf{y}$
8 $S'_n \leftarrow S'_n + \mathbf{y}^2$
9 $k_n \leftarrow K_{n,0}$
10 $V_n \leftarrow \frac{S'_n}{k_n} - \left(\frac{S_n}{k_n}\right)^2$
11 {multi-armed bandit}
12 **for** $k \leftarrow \sum_n^N K_{n,0}$ **to** K **do**
13 $n \leftarrow \arg\max_n \left[\mathrm{Ucb}(k, k_n, V_n)\right]$
14 sample $\mathbf{x}$ in Ω_n
15 $\mathbf{y} \leftarrow f(\mathbf{x})$
16 $S_n \leftarrow S_n + \mathbf{y}$
17 $S'_n \leftarrow S'_n + \mathbf{y}^2$
18 $k_n \leftarrow k_n + 1$
19 $V_n \leftarrow \frac{S'_n}{k_n} - \left(\frac{S_n}{k_n}\right)^2$
20 {final result}
21 $F' \leftarrow \sum_n^N v_n \frac{S_n}{k_n}$

In classic stratified sampling applications, it is generally recommended to compute at least 10 samples in each region [17]. Here, the stratified sampling step only aims to initialize the following step of the algorithm so this size can be reduced (in our experiments, we used $K_{n,0} = 2, \forall n$). Moreover, if we have a total number of samples K, increasing the number of initial samples for the stratified sampling step implies decreasing the remaining number of samples for the online learning step (to $K - \sum_{n=1}^{N} K_{n,0}$).

With these initial samples, we can compute an initial estimation of the mean value and of the variance, in every regions. Since we have to update these values with new samples, in the online learning step, we compute the variances (without storing the samples) thanks to the König-Huygens formula [18], defined by:

$$\mathrm{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2. \tag{11}$$

Indeed, if we store, for each region n, the sum of the integrand evaluations (S_n), the sum of the squared integrand evaluations (S'_n) and the number of samples already taken in the region (k_n), then the current variance of the region is:

$$V_n = \frac{S'_n}{k_n} - \left(\frac{S_n}{k_n}\right)^2. \tag{12}$$

C. Sampling with UCB

Once regions are initialized (with their corresponding variances, sums and number of samples), we can easily apply an UCB strategy to iteratively select the most interesting region to sample (lines 11 to 19 of Algorithm 1).

To define the *reward* process of this multi-armed bandit problem (see equation 10), we specify the score of an arm (i.e. region) as:

$$\mathrm{Ucb}(k, k_n, V_n) = \frac{V_n}{\sqrt{k_n}} + R\sqrt{\frac{\ln k}{k_n}}, \tag{13}$$

where k is the total number of function evaluations, k_n is the number of function evaluations for the arm n (which is roughly the number of times the arm has already been selected), V_n is the variance of the respective region (interpreted as the reward of the arm), and R is an exploration parameter.

The left term in equation 13 is the exploitation part: it tends to encourage regions that have a high reward (variance). The right term is the exploration part: it tends to encourage the arms that have not been selected many times. The parameter R defines the importance granted to exploration: if R has a high value the algorithm tends to sample equally in every region (similarly to the stratified sampling technique). Thus, equation 13 ensures the trade-off discussed in section II-C.

The main loop of our algorithm consists in determining the current best region to sample (according to UCB scores), computing a sample in this region, and then updating the region (variance, sums, number of samples). This process is repeated until we reach the number of function evaluations (K). In this way, we expect that the algorithm will preferably sample inside interesting regions, while not totally ignoring other regions that appear to be less attractive.

D. Computing final result

When all samples are computed, the last step consists in computing the final value of the integral. Since the algorithm is based on stratified sampling, equation 5 can be used. In Algorithm 1, this means computing:

$$F' = \sum_{n=1}^{N} v_n \frac{S_n}{k_n}, \tag{14}$$

where v_n is the volume of the region n.

IV. Experiments

In this section, we present the integrand functions we use for evaluating the performance of the proposed method. We also detail the testing procedure and we present and discuss some of our results.

A. Integrand functions

We use the classic TESTPACK suite of multidimensional integrand functions [13]. We also use more complex functions, defined as follow:

$$f_{\text{sphere}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \|\mathbf{x}\|_{L_2} < 1 \\ 0 & \text{otherwise} \end{cases}, \tag{15}$$

$$f_{\text{sinc}}(\mathbf{x}) = 0.2 + \frac{0.8}{D} \sum_{i=1}^{D} \frac{\sin(\pi \alpha_i x_i)}{\pi \alpha_i x_i} \tag{16}$$

and

$$f_{\text{sinc2cos}}(\mathbf{x}) = \max\left[0 \ , \ f_{\text{sinc2cos'}}(\mathbf{T}(\mathbf{x}))\right] \tag{17}$$

where $\mathbf{T}$ is a rotation group and

$$\begin{aligned} f_{\text{sinc2cos'}}(\mathbf{y}) = & \left[0.5 + f_{\text{sinc}}(0.5 \times \mathbf{y})\right] \\ & \times \frac{0.5}{D} \sum_{i=1}^{D} \cos(\alpha_i y_i + \beta_i) \\ & \times \left[1 + \frac{0.1}{D} \sum_{i=1}^{D} \cos(50 \times y_i)\right]. \end{aligned}$$

In these definitions, D is the dimension of the integration domain, and α and β are two parameters that affect the shape of the integrand functions: α is a scaling vector (e.g. peak width) and β is a displacement vector (e.g. peak location). Thus, the previous equations define several families of functions and the two parameters enable us to define different functions and difficulties, from these families.

Some of the test integrand functions used in our experiments are depicted Fig. 1.

B. Testing procedure

We use the classic procedure described in [13] for testing multidimensional integration algorithms. This procedure consists in applying the integration algorithm to various integrand functions using the function families described previously. To this end, we randomly choose two parameter vectors α and β and normalize α to a given difficulty (the greater the norm, the more difficult the problem is). Then, we estimate the ground-truth of the corresponding integral using a Monte-Carlo simulation performed with a very high number of samples (10 millions). Finally, we run the integration algorithms we want to compare on the integrand function and repeat the whole procedure, for another random integrand.

To evaluate the results of an algorithm, we compute the mean absolute error, defined by:

$$\frac{1}{M} \sum_{m=1}^{M} \left| I_m - \hat{I}_m \right| \tag{18}$$

where M is the number of random integrands (in our experiments: one thousand for each family), I_m is the estimated ground-truth for the integrand i and $\hat{I}_m$ is the resulting integral returned by the algorithm.

C. Results and discussion

We compare the UCBATURE algorithm with two classic numerical integration algorithms: MONTE-CARLO and STRATIFIED-SAMPLING. For UCBATURE and STRATIFIED-SAMPLING, we partition each dimension into $N_0 = 3$ parts, which means the total number of regions is $N = 3^D$. For the exploration parameter R of UCBATURE, we tried several values and selected the best one, for each integrand family. These parameters may be tuned more finely, since the dimension, the number of regions and the bounds of the integration domain may affect the performance of the algorithms. We do not investigate deeply the tuning of the common parameters because this affect similarly all the algorithms, and our goal is to study their differences.

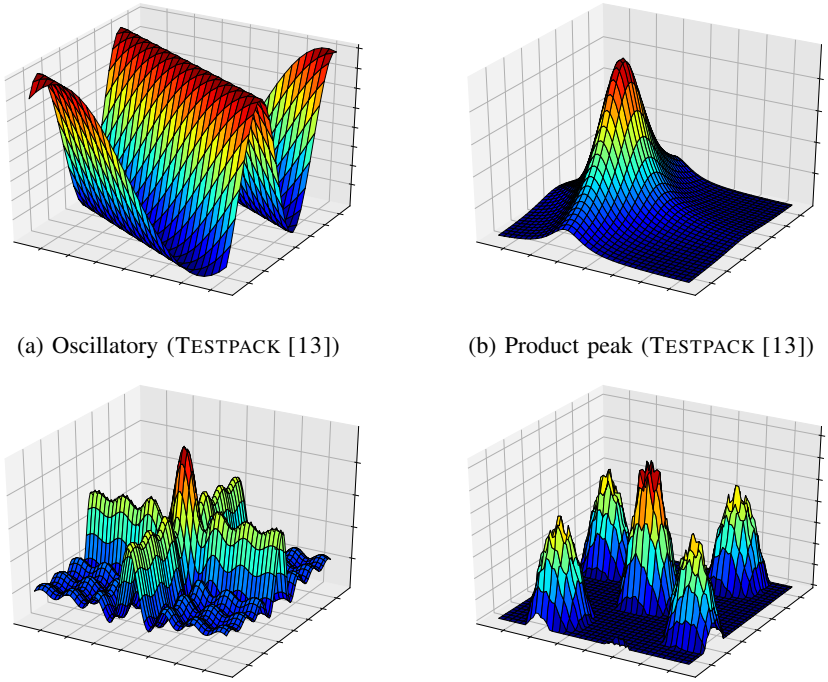
Our results show that UCBATURE generally converges faster than MONTE-CARLO and STRATIFIED-SAMPLING (see Table I and Fig. 2): for a given number of integrand evaluations, our algorithm has a lower average error than the two other algorithms. This benefit is clearly noticeable on single-peak integrands (e.g. *gaussian* and *discontinuous*), however it is less noticeable on more global integrands (e.g. *oscillatory* and *sinc2cos*). This can be explained by the fact that "global integrands" have a large but homogeneous variance across all regions, therefore no region is significantly more interesting than the other ones. Thus, UCBATURE mainly performs exploration hence performances are similar to STRATIFIED SAMPLING. A thinner partitioning should make the regions more heterogeneous and increase UCBATURE efficiency.

In higher dimensions, we observe a slightly slower learning process (see Table II and Fig. 3): with few integrand evaluations UCBATURE has generally greater errors than STRATIFIED-SAMPLING but with more integrand evaluations UCBATURE becomes better. This can be explained by the higher number of regions: since the algorithm privileges exploitation during the first integrand evaluations, it needs more samples to learn the interesting regions to sample.

V. CONCLUSION

In this paper, we propose an online learning approach to compute multidimensional numerical integration. Our method is based on the Monte-Carlo method with the stratified sampling technique, and uses the Upper Counfidence Bound technique to iteratively select the most interesting region to sample. Our experiments indicate that this approach brings noticeable benefits and, more generally, that machine learning methods seem promising for solving numerical integration problems.

Since we use a naive partitioning scheme, the proposed algorithm is limited to moderate dimensions [19]. In future works, we would investigate using an adaptive stratification scheme or a post-stratification method [20] to overcome this problem. Finally, we would simplify the use of the algorithm by automatically tuning the exploration parameter.



(a) Oscillatory (TESTPACK [13])

(b) Product peak (TESTPACK [13])

(c) f_{sinc} (equation 16)

(d) f_{sinc2cos} (equation 17)

Fig. 1: Some of the integrand functions used in our experiments (for a 2D integration domain).

TABLE I: Integration error of the three algorithms for the tested integrand families, in 2D.

integrand	$\mathbf{10^3}$ **evaluations**			$\mathbf{10^6}$ **evaluations**		
	MONTE-CARLO	**STRATIFIED**	**UCBATURE**	**MONTE-CARLO**	**STRATIFIED**	**UCBATURE**
oscillatory	0.070	0.068	0.052	0.0024	0.0023	0.0017
product peak	39.89	31.83	19.04	1.27	1.04	0.70
gaussian	0.0096	0.0086	0.0053	0.00030	0.00028	0.00019
C^0-continuous	0.0071	0.0062	0.0043	0.00024	0.00022	0.00016
discontinuous	0.0039	0.0040	0.0030	0.00013	0.00012	0.00010
sphere	0.041	0.037	0.03	0.0014	0.0011	0.0009
sinusc	1.74	1.26	1.14	0.057	0.042	0.037
sinc2cos	0.36	0.30	0.22	0.012	0.01	0.008

TABLE II: Integration error of the three algorithms for the tested integrand families, in 5D.

integrand	$\mathbf{10^3}$ **evaluations**			$\mathbf{10^6}$ **evaluations**		
	MONTE-CARLO	**STRATIFIED**	**UCBATURE**	**MONTE-CARLO**	**STRATIFIED**	**UCBATURE**
oscillatory	0.54	0.32	0.30	0.017	0.011	0.011
product peak	1342	1032	1058	41	34	28
gaussian	0.020	0.017	0.019	0.00065	0.00057	0.00043
C^0-continuous	0.010	0.0084	0.0087	0.00034	0.00029	0.00024
discontinuous	0.0020	0.0018	0.0013	6.40e−5	6.27e−5	5.18e−5
sphere	0.31	0.25	0.31	0.01	0.008	0.007
sinusc	8422	6545	8025	392	291	280
sinc2cos	175	163	165	5.65	4.98	4.63

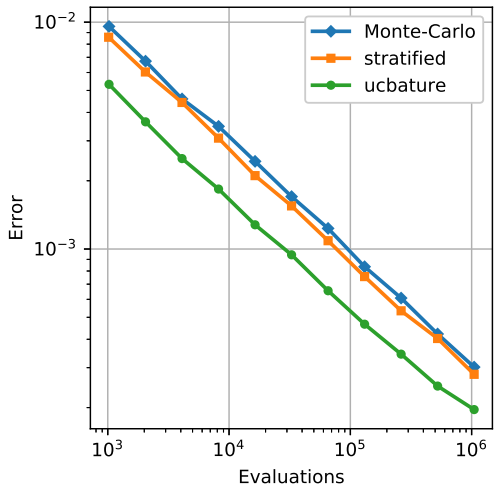


Fig. 2: Evolution of the integration error for the 2D gaussian integrand family, according to the number of integrand evaluations ($R = 0.01$, $N_0 = 3$).

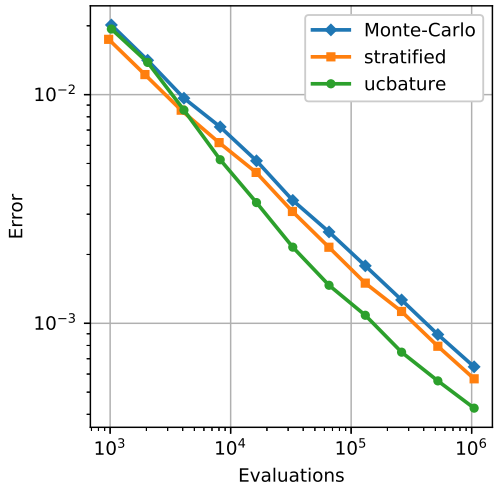


Fig. 3: Evolution of the integration error for the 5D gaussian integrand family, according to the number of integrand evaluations ($R = 0.0125$, $N_0 = 3$).

Acknowledgment

Experiments presented in this paper were carried out using the CALCULCO computing platform, supported by SCOSI/ULCO (Service COmmun du Système d'Information de l'Université du Littoral Côte d'Opale).

References

[1] R. Piessens, E. d. Kapenga, C. Uberhuber, and D. Kahaner, *QUADPACK: A Subroutine Package for Automatic Integration*. Springer–Verlag, 1983.

[2] G. P. Lepage, "A new algorithm for adaptive multidimensional integration," *Journal of Computational Physics*, vol. 27, no. 2, pp. 192 – 203, 1978.

[3] T. Hahn, "Cuba - a library for multidimensional numerical integration," *Computer Physics Communications*, vol. 168, no. 2, pp. 78 – 95, 2005.

[4] J. Berntsen, T. O. Espelid, and A. Genz, "An adaptive algorithm for the approximate calculation of multiple integrals," *ACM Trans. Math. Softw.*, vol. 17, no. 4, pp. 437–451, 1991.

[5] T. Gerstner and M. Griebel, "Numerical integration using sparse grids," *Numerical Algorithms*, vol. 18, no. 3, p. 209, 1998.

[6] K. Petras, "Fast calculation of coefficients in the smolyak algorithm," *Numerical Algorithms*, vol. 26, no. 2, pp. 93–109, 2001.

[7] S. A. Smolyak, "Quadrature and interpolation formulas for tensor products of certain class of functions," *Dokl. Akad. Nauk SSSR*, vol. 148, no. 5, pp. 1042–1053, 1963.

[8] N. Metropolis and S. Ulam, "The monte carlo method," *Journal of the American Statistical Association*, vol. 44, no. 247, pp. 335–341, 1949.

[9] G. S. Fishman, *Monte Carlo: Concepts, algorithms, and applications*, ser. Springer Series in Operations Research. New York: Springer-Verlag, 1996.

[10] W. H. Press and G. R. Farrar, "Recursive stratified sampling for multidimensional monte carlo integration," *Comput. Phys.*, vol. 4, no. 2, pp. 190–195, 1990.

[11] C. Rasmussen and Z. Ghahramani, "Bayesian monte carlo," in *Advances in Neural Information Processing Systems 15*, Max-Planck-Gesellschaft. Cambridge, MA, USA: MIT Press, 2003, pp. 489–496.

[12] F.-X. Briol, C. Oates, M. Girolami, and M. A. Osborne, "Frank-wolfe bayesian quadrature: Probabilistic integration with theoretical guarantees," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 1162–1170.

[13] A. Genz, *A Package for Testing Multiple Integration Subroutines*. Dordrecht: Springer Netherlands, 1987, pp. 337–340.

[14] E. Veach, "Robust monte carlo methods for light transport simulation," Ph.D. dissertation, Stanford University, 1997.

[15] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in applied mathematics*, vol. 6, no. 1, pp. 4–22, 1985.

[16] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2-3, pp. 235–256, 2002.

[17] J. A. Westfall, P. L. Patterson, and J. W. Coulston, "Post-stratified estimation: within-strata and total sample size recommendations," *Canadian Journal of Forest Research*, vol. 41, no. 5, pp. 1130–1139, 2011.

[18] A. Mood, F. Graybill, and D. Boes, *Introduction to the Theory of Statistics*, ser. International Student edition. McGraw-Hill, 1974.

[19] D. P. Mitchell, "Consequences of stratified sampling in graphics," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '96. New York, NY, USA: ACM, 1996, pp. 277–280.

[20] D. Holt and T. M. F. Smith, "Post stratification," *Journal of the Royal Statistical Society. Series A (General)*, vol. 142, no. 1, pp. 33–46, 1979.