# A Multilevel Monte Carlo Estimator for Matrix Multiplication

# A MULTILEVEL MONTE CARLO ESTIMATOR FOR MATRIX MULTIPLICATION*

YUE WU † AND NICK POLYDORIDES ‡

**Abstract.** Inspired by recent developments in multilevel Monte Carlo (MLMC) methods and randomised sketching for linear algebra problems we propose a MLMC estimator for real-time processing of matrix structured random data. Our algorithm is particularly effective in handling high-dimensional inner products and matrix multiplication, and finds applications in computer vision and large-scale supervised learning.

**Key words.** Sketching, inner product, matrix multiplication, real-time computing.

**AMS subject classifications.** 68W20.

**1. Introduction.** Randomised algorithms for matrix operations are in general 'pass-efficient', and are primarily aimed at problems involving massive data sets that are otherwise cumbersome to process with deterministic algorithms. Pass-efficient implies that the algorithm necessitates only a very small number of passes through the complete data set, but for the cases we consider here such a pass maybe turn out to be impractical due to memory or time restrictions. In matrix multiplication for example, the BasicMatrixMultiplication algorithm [3] is considered to be the gold standard. Based on a probability assigned to the columns of a matrix $A$, and respectively the rows of a matrix $B$, it approximates the product $AB$ through re-scaling the outer products of some sampled columns of $A$ with the corresponding rows of $B$ via a sampling-and-rescaling matrix operator. Variants of the BasicMatrixMultiplication algorithm were published in [4], [8], [13], exploiting different types of information available on the elements of the matrices involved. In particular, the algorithm in [4] addresses the case where the probability distributions of the elements are known a priori to devise an importance sampling strategy based on BasicMatrixMultiplication that minimizes the expected value of the variance. The algorithm was shown to be effective when implemented with the optimized sampling probabilities, particularly so in comparison to the estimators resulting from uniform sampling. This result indeed extends BasicMatrixMultiplication to a random variable setting and can be applied to many query matching with information retrieval applications [4]. However, designing the optimized probabilities relies exclusively on the knowledge of the probability distributions of the matrix elements, which limits its applicability to the cases where such information is a priori available. Conversely, it can be argued that BasicMatrixMultiplication with uniform probabilities becomes more appealing when dealing with real-time random matrix multiplication tasks, where distributions change dynamically. In batch processing for instance, the task at hand is to evaluate the expectation of the multipli-

*Submitted to the editors August 14, 2020.

**Funding:** NP and YW are grateful to EPSRC for funding this work through the project EP/R041431/1: 'Randomness: a resource for real-time analytics'; YW is also funded by The Alan Turing Institute under the EPSRC grant EP/N510129/1 and by EPSRC through the project EP/S026347/1:' Unparameterised multi-modal data, high order signatures, and the mathematics of data science'.

†Mathematical Institute, University of Oxford, Oxford, UK; The Alan Turing Institute, London, UK; (yue.wu@maths.ox.ac,uk).

‡School of Engineering, University of Edinburgh, Edinburgh, UK; The Alan Turing Institute, London, UK; (n.polydorides@ed.ac.uk).

cation or indeed a functional of a matrix product at any given time, a formidable task in terms of the required speed and accuracy. To accelerate the time-dependent training of large-scale kernel machines for example, the evaluation of a kernel function is identified as and approximated through the expectation of a random inner product via some randomised feature map [10], [11]. In this case, coupling a standard Monte Carlo method (MC) and BasicMatrixMultiplication with uniform probabilities may satisfy the speed specifications but compromise the accuracy of the result. A more prudent alternative is to employ a multilevel Monte Carlo method, similar to the one developed in [5] instead of MC.

MLMC was initially conceived for reducing the cost of computing the expected value of a financial derivative whose payoff depends upon the solution of a stochastic differential equation (SDE). The framework in [5] generalizes Kebaier's approach in [9] to multiple levels, using a geometric sequence of different time step sizes. In doing so it reduces substantially the computational cost of MC by taking most of the samples on coarse grids at low cost and accuracy, and only a few samples on finer computationally expensive grids that lead to solutions of high accuracy. Over time, MLMC has grown in scope and found a wide range of applications in the broad area of SDEs, SPDEs, for stochastic reaction networks and inverse problems [12], while further variants have been developed in the form of multilevel quasi-Monte Carlo estimators [6] and multilevel sequential Monte Carlo samplers [1]. For an overview on MLMC we refer the reader to the excellent survey [7]. Therein the author emphasizes that the multilevel theorem allows to use other estimators as long as they satisfy some specific conditions. This theorem lays the foundation for the algorithm proposed in this paper. A closely related work [2] considers the MLMC estimate for approximating the mean field of a nonlinear PDE, providing a theoretical framework in separable Hilbert spaces. Although there is clearly no actual time stepsize in the matrix multiplication context, we can draw an analogy between the term *time stepsize* in numerical analysis for differential equations and the term *the size of the sampled index set* in randomised linear algebra. As anticipated in a convergent MLMC scheme, the numerical estimation error shrinks with decreasing time stepsize. Similarly, due to the law of large numbers, increasing the size of index samples will decrease the expected squared Frobenius approximation error as shown in Lemma 4 of the seminal work [3]. Therefore we claim that a random strategy for matrix multiplication with fewer index samples is analogous to using a "coarser grid" in the PDE setting. This observation is crucial to our construction of MLMC estimators for matrix multiplication.

In Section 2 below we begin by discussing the simpler case of calculating 'on the fly' the expectation of the inner product between large random vectors. We first consider the BasicMatrixMultiplication algorithm with uniform probability and proceed to review the main results for the inner product from [4]. We then introduce the important quantities *base number* $M \in \mathbb{N}$ and *level size* $L \in \mathbb{N}$ based on which the MLMC estimator (c.f. (2.9) and (2.10)) is constructed via inner product approximations with index sample sizes $M^0, M^1, \ldots, M^L$. In this context, the approximation on the 'finest grid' corresponds to the inner product realization with $M^L$ samples. Here we note the distinction between samples and indices, in that since we are sampling with replacement, taking $M^L > n$ samples does not imply sampling all $n$ indices. Given that the variance of the approximated inner product is proportional to $M^{-l}$ for $l \in \{1, \ldots, L\}$ (c.f. Theorems 2.1 and Theorem 2.2), the complexity of the proposed MLMC estimator for a functional of the inner product conditioned on certain features of the underlying approximation can be treated similarly as the case $\beta = 1$ of Theorem 3.1 in [5]. This result is revisited in Theorem 2.2 where a comparison with

standard MC is attempted. Corollary 2.4 discusses the computational complexity of our MLMC estimator using Theorem 2.2. At the end of Section 2, we comment on the optimal choice of base number $M$ following the reasoning in [5].

In Section 3 we extend our approach to matrix multiplication, adapting Theorems 2.1 and 2.2 accordingly. It is worth mentioning that, because the approximation error (c.f. Theorems 3.1 and 3.2) is measured in expectation as a Frobenius norm, for the analysis the matrices are considered transformed in vector form prompting a new definition of 'variance' for the vectorized matrices denoted as $\mathbb{V}_{\|}$. Further, Theorem 3.3 discusses the complexity and Corollary 3.4 validates the complexity of the MLMC estimator for matrix multiplication. The implementation of our method is presented as Algorithm 3.2. Finally, in Section 4 we present two simple numerical experiments to illustrate the performance of the MLMC estimator in comparison with the standard MC one. By making appropriate choices for $M$ and $L$ parameters, the proposed MLMC estimator outperforms the MC estimator in terms of accuracy as well as speed and computational efficiency.

**2. Inner product.** We define $\mathbf{T}$ as a countable collection of discrete time points and set $t \in \mathbf{T}$. Let $\mathbf{a}(t)$ and $\mathbf{b}(t)$ be two random vectors of length $n$, whose elements are drawn from some unknown, perhaps different, probability distributions, say $\mathbf{a}(t) \sim \mathcal{L}_{\mathbf{a}(t)}$ and $\mathbf{b}(t) \sim \mathcal{L}_{\mathbf{b}(t)}$. Here and throughout this paper, $n$ is assumed to be extremely large such that evaluating the inner product of $\mathbf{a}(t)^T\mathbf{b}(t)$ is deemed impractical if at all possible. Consider that there is a need to compute $\mathbb{E}_{\mathbf{a}(t),\mathbf{b}(t)}[f(\mathbf{a}(t)^T\mathbf{b}(t))]$ on demand, at different times, where $f$ is a Lipchitz function with Lipchitz constant $C_f$ and $\mathbb{E}_{\mathbf{a}(t),\mathbf{b}(t)}$ is the expectation under $\mathcal{L}_{\mathbf{a}(t)}$ and $\mathcal{L}_{\mathbf{b}(t)}$. For the sake of notational simplicity, the argument $(t)$ is suppressed in the notation but assumed implicitly in all of the quantities introduced above.

Indeed the task at hand consists of two main parts: approximating $\mathbf{a}^T\mathbf{b}$ in an efficient and accurate manner and approximating its expected value in the spirit of Monte Carlo methods. To tackle the first issue, the random sampling method for inner product estimation presents a viable option. Suppose there is a sampling distribution $\xi := \{\xi_j\}_{j=1}^n$ with $\sum_{j=1}^n \xi_j = 1$ such that each index $j \in [n]$, where $[n] := \{1, 2, \ldots, n\}$, can be drawn with an assigned positive probability $\xi_j$. Further suppose we *fix* a 'base' number $M \in \mathbb{N}$ and collect $M^L$, $L \in \mathbb{N}$, independent and identically distributed index samples as an index sequence $(r_1, \ldots, r_{M^L})$ according to $\xi$. We shall refer to these collected $M^L$ indices, or equivalently, the sequence $(r_1, \ldots, r_{M^L})$, as a sample *realisation*. Then denote by $S_L$ the *sampling-and-rescaling matrix* of size $n \times M^L$ such that elements of $\mathbf{a}$ and $\mathbf{b}$ at the $M^L$ index samples will be used for approximating the inner product of $\mathbf{a}^T\mathbf{b}$. That is,

$$\widehat{\mathbf{a}^T\mathbf{b}} = \mathbf{a}^T S_L S_L^T \mathbf{b} = \frac{1}{M^L} \sum_{i=1}^{M^L} \frac{1}{\xi_{r_i}} \mathbf{a}_{r_i} \mathbf{b}_{r_i} := X_L(\xi), \tag{2.1}$$

where $X_L(\xi)$ denotes a scalar random variable that approximates the target $\mathbf{a}^T\mathbf{b}$ using $M^L$ samples from $\xi$, emphasizing its dependence on $\xi$. Previous research has shown that $X_L(\xi)$ is an unbiased estimator of $\mathbf{a}^T\mathbf{b}$ under the sampling distribution $\xi$. The performance of the approximation when the vector elements $\mathbf{a}$ and $\mathbf{b}$ are known only up to their distributions can be assessed through quantifying the variance of the estimator. The minimum variance is attained when sampling according to the distribution given by the following theorem from [4].

THEOREM 2.1. *If the vector elements* $\mathbf{a}_j$ *and* $\mathbf{b}_j$ *are independent random variables,* $j \in [n]$*, with finite and nonzero moments* $\mathbb{E}_{\mathbf{a},\mathbf{b}}[\mathbf{a}_j^2\mathbf{b}_j^2]$*, then the probability* $\xi^*$ *with elements*

$$\xi_j^* = \frac{\sqrt{\mathbb{E}_{\mathbf{a},\mathbf{b}}[\mathbf{a}_j^2\mathbf{b}_j^2]}}{\sum_{i=1}^n \sqrt{\mathbb{E}_{\mathbf{a},\mathbf{b}}[\mathbf{a}_i^2\mathbf{b}_i^2]}}, \tag{2.2}$$

*minimizes the expected value of the variance in* (2.1)*, that is,*

$$\min_{\xi} \mathbb{E}_{\mathbf{a},\mathbf{b}}[\,\boldsymbol{Var}[X_L(\xi)]] = \mathbb{E}_{\mathbf{a},\mathbf{b}}[\,\boldsymbol{Var}[X_L(\xi^*)]] := \frac{\mu}{M^L}, \tag{2.3}$$

*where* $\boldsymbol{Var}$ *is the variance under* $\xi$ *and* $\mu = \mathbb{E}_{\mathbf{a},\mathbf{b}}\Big[\sum_{i=1}^n \frac{\mathbf{a}_i^2\mathbf{b}_i^2}{\xi_i^*} - (\mathbf{a}^T\mathbf{b})^2\Big]$.

Sampling with $\xi^*$ is clearly not practical when we have no knowledge about the distributions of $\mathbf{a}$ and $\mathbf{b}$ in advance, hence a plausible convenient alternative is to use a uniform probability over the index set

$$\xi_j^u = \frac{1}{n}, \quad j \in [n], \tag{2.4}$$

with variance as follows.

THEOREM 2.2. *[4] Assume the same setting as in Theorem 2.1 but with probability* $\xi^u$ *defined in* (2.4)*, then the variance is*

$$\mathbb{E}_{\mathbf{a},\mathbf{b}}[\,\boldsymbol{Var}[X_L(\xi^u)]] = \mathbb{E}_{\mathbf{a},\mathbf{b}}[\,\boldsymbol{Var}[X_L(\xi^*)]] + \frac{n\nu}{M^L} = \frac{n\nu+\mu}{M^L}, \tag{2.5}$$

*where*

$$\nu = \sum_{i=1}^n \Big(\sqrt{\mathbb{E}_{\mathbf{a},\mathbf{b}}[\mathbf{a}_i^2\mathbf{b}_i^2]} - \frac{1}{n}\sum_{j=1}^n \sqrt{\mathbb{E}_{\mathbf{a},\mathbf{b}}[\mathbf{a}_j^2\mathbf{b}_j^2]}\Big)^2.$$

Typically one may consider approximating the expectation using a standard MC method that simulates $\mathbb{E}_{\mathbf{a},\mathbf{b}}[f(\mathbf{a}^T\mathbf{b})]$. In this instance, the quantity of interest, say $P$, can then be estimated by (2.1) with a uniform probability (2.4) and MC as

$$\mathbb{E}_{\mathbf{a},\mathbf{b}}[P] := \mathbb{E}_{\mathbf{a},\mathbf{b}}[f(\mathbf{a}^T\mathbf{b})] \approx \frac{1}{N}\sum_{k=1}^N f\big((\mathbf{a}^{(k)})^T\mathbf{b}^{(k)}\big) = \frac{1}{N}\sum_{k=1}^N f\big(X_L^{(k)}(\xi^u)\big) := \hat{P}, \tag{2.6}$$

where $N$ is the number of realisations for $M^L$ many index samples or equivalently, an index sequence of length $M^L$. In this case the mean square error (MSE) for the estimate $\hat{P}$ turns out to be

$$\begin{aligned}\mathbb{E}\big[\big(\hat{P} - \mathbb{E}[P]\big)^2\big] &= \mathbb{E}\big[\big(\hat{P} - \mathbb{E}[\hat{P}]\big)^2\big] + \big(\mathbb{E}[P] - \mathbb{E}[\hat{P}]\big)^2 \\ &= \mathbb{E}\big[\big(\hat{P} - \mathbb{E}[\hat{P}]\big)^2\big] + \big(\mathbb{E}[f(\mathbf{a}^T\mathbf{b}) - f(X_L(\xi^u))]\big)^2,\end{aligned} \tag{2.7}$$

where $\mathbb{E}$, and also $\mathbb{V}$ that appears in the sequel, (without subscripts) denote respectively the expectation and the variance under $\mathcal{L}_\mathbf{a}$, $\mathcal{L}_\mathbf{b}$ and $\xi^u$. The last term in (2.7), for a fixed $L$, characterizes the bias and can be bounded by

$$\begin{aligned}&\big(\mathbb{E}[f(\mathbf{a}^T\mathbf{b}) - f(X_L(\xi^u))]\big)^2 \le \mathbb{E}\big[\big(f(\mathbf{a}^T\mathbf{b}) - f(X_L(\xi^u))\big)^2\big] \\ &= \mathbb{E}\big[\big(f(\mathbf{a}^T\mathbf{b}) - f(\mathbf{a}^T S_L S_L^T\mathbf{b})\big)^2\big] \le C_f^2\mathbb{E}[|\mathbf{a}^T(I - S_L S_L^T)\mathbf{b}|^2] \sim \mathcal{O}(M^{-L}),\end{aligned}$$

where the first inequality comes from Jensen's inequality, that is $\mathbb{E}[X]^2 \leq \mathbb{E}[X^2]$ for arbitrary random variable $X$, the second inequality is due to the Lipchitz continuity of $f$ and the last one due to (2.5). The first term in (2.7) is simply the variance from the MC simulation and can be bounded in terms of $N$ as

$$\begin{aligned}
&\mathbb{E}\big[\big(\hat{P} - \mathbb{E}[\hat{P}]\big)^2\big] = \mathbb{V}[\hat{P}] = \frac{1}{N}\mathbb{V}[f(X_L(\xi^u))] \\
&\leq \frac{1}{N}\Big(\mathbb{V}[f(X_L(\xi^u)) - f(\mathbf{a}^T\mathbf{b})]^{\frac{1}{2}} + \mathbb{V}[f(\mathbf{a}^T\mathbf{b})]^{\frac{1}{2}}\Big)^2 \qquad (2.8)\\
&\leq \frac{1}{N}\Big(\frac{C_f}{M^{\frac{L}{2}}}(n\nu+\mu)^{\frac{1}{2}} + \mathbb{V}_{\mathbf{a},\mathbf{b}}[f(\mathbf{a}^T\mathbf{b})]^{\frac{1}{2}}\Big)^2 \sim \mathcal{O}(N^{-1}).
\end{aligned}$$

Overall, as in [5], the MSE varies in terms of $\frac{1}{M^L}$ and $\frac{1}{N}$. This is still true even if we sample based on the optimal sampling probability (2.2). Meanwhile, the complexity is in terms of $NM^L$, for an integer $N$ to be determined.

Alternatively, it may be possible to obtain the same accuracy at a reduced computational cost, by considering a multilevel MC simulation [5]. For $l \in [L]\bigcup\{0\}$ define as $\hat{P}_l$ the approximation to $f(\mathbf{a}^T\mathbf{b})$ from $M^l$ sampled indices. Further define $\hat{Y}_l$ as an estimator of $\mathbb{E}[\hat{P}_l - \hat{P}_{l-1}]$ using $N_l$ realizations with $l > 0$ and similarly $\hat{Y}_0$ the estimator of $\mathbb{E}[\hat{P}_0]$ using $N_0$ samples, that is

$$\hat{Y}_l := \frac{1}{N_l}\sum_{k=1}^{N_l}(\hat{P}_l^{(k)} - \hat{P}_{l-1}^{(k)}). \qquad (2.9)$$

A key point to note is that both $\hat{P}_l^{(k)}$ and $\hat{P}_{l-1}^{(k)}$ emerge from the *same* realization, as we discuss in more detail when we describe our Algorithm 3.1. By the linear property of the expectation it follows immediately that

$$\mathbb{E}[\hat{P}_L] = \mathbb{E}[\hat{P}_0] + \sum_{l=1}^{L}\mathbb{E}[\hat{P}_l - \hat{P}_{l-1}] \approx \hat{Y}_0 + \sum_{l=1}^{L}\hat{Y}_l := \hat{Y}, \qquad (2.10)$$

where clearly $\mathbb{E}[\hat{P}_L] = \mathbb{E}[\hat{Y}]$. To investigate the performance of the proposed MLMC estimator $\hat{Y}$ in (2.10) we compare the complexity of two estimators $\hat{Y}$ and $\hat{P}$ at the same accuracy level.

THEOREM 2.3. *Let* $\mathbf{a}$ *and* $\mathbf{b}$ *be two random vectors with length* $n$ *drawn from different unknown distributions, that is* $\mathbf{a} \sim \mathcal{L}_\mathbf{a}$ *and* $\mathbf{b} \sim \mathcal{L}_\mathbf{b}$*, and let* $f : \mathbb{R} \to \mathbb{R}$ *be a Lipschitz function with Lipschitz number* $C_f$*. Denote by* $P$ *the term of interest as in* (2.6)*, and define* $\hat{P}_l$ *the corresponding approximation to* $f(\mathbf{a}^\mathbf{T}\mathbf{b})$ *based on the sketched version of matrix multiplication via* $M^l$ *many index samples like in* (2.1)*.*

1. *If there exist independent estimators* $\hat{Y}_l$ *as in* (2.9) *based on* $N_l$ *Monte Carlo samples, and positive constants* $c_1$*,* $c_2$*,* $c_3$ *such that*
   (a) $\mathbb{E}[\hat{P}_l - P] \leq c_1 M^{-\frac{l}{2}}$,
   (b) $\mathbb{V}[\hat{Y}_l] \leq c_2 N_l^{-1} M^{-l}$,
   (c) *the complexity of* $\hat{Y}_l$*, denoted by* $C_l$*, is bounded by* $C_l \leq c_3 N_l M^l$,

   *then there exists a positive constant* $c_4$ *such that for* $\epsilon < e^{-1}$*, there are values* $L$ *and* $N_l$ *for which the multilevel estimator* $\hat{Y} = \sum_{l=0}^{L}\hat{Y}_l$ *has an MSE* $\mathbb{E}[(\hat{Y} - P)^2]$ *with bound* $\epsilon^2$*, and computational complexity*

$$C(\hat{Y}) := \sum_{l=0}^{L} C_l \leq c_4\epsilon^{-2}(\log\epsilon)^2.$$

2. *Furthermore, define the estimator based on the finest level $L$ and $N$ realisations as in (2.6) with either the optimal sampling probability (2.2) (if tractable) or the uniform probability (2.4), and suppose*
   (a) *the variance for $\hat{P}$ is bounded by the same constant $c_2$, i.e., $\mathbb{V}[\hat{P}] \leq c_2 N^{-1}$,*
   (b) *the complexity for $\hat{P}$ is bounded by the same constant $c_3$, i.e., $C(\hat{P}) \leq c_3 N M^L$,*

   *then at the same accuracy $\epsilon^2$, $C(\hat{P}) \leq c_6 \epsilon^{-4}$, which is much larger than the upper bound of $C(\hat{Y})$ when $\epsilon$ is sufficiently small.*

*Proof.* 1. The proof is based on [5]. Accordingly, the MSE for $\hat{Y}$ is

$$\begin{aligned}\mathbb{E}\big[(\mathbb{E}[P]-\hat{Y})^2\big] &= (\mathbb{E}[P]-\mathbb{E}[\hat{Y}])^2 + \mathbb{E}\big[\big(\hat{Y}-\mathbb{E}\big[\hat{Y}\big]\big)^2\big] \\ &= (\mathbb{E}[P]-\mathbb{E}[\hat{P}_L])^2 + \mathbb{V}[\hat{Y}],\end{aligned}$$

where $L$ is to be determined. If choosing the ceiling

$$L = \lceil \frac{\log(2c_1^2\epsilon^{-2})}{\log M} \rceil, \tag{2.11}$$

then its bias component can be bounded via condition 1.(a)-(b) as

$$\big(\mathbb{E}[P]-\mathbb{E}[\hat{P}_L]\big)^2 \leq c_1^2 M^{-L} \leq \frac{1}{2}\epsilon^2.$$

On the other hand, choosing

$$N_l = \lceil 2(L+1)c_2\epsilon^{-2}M^{-l} \rceil \tag{2.12}$$

together with condition 1.(b) gives that

$$\begin{aligned}\mathbb{V}[\hat{Y}] &\leq \sum_{l=0}^{L} \mathbb{V}[\hat{Y}_l] \leq c_2 \sum_{l=0}^{L} N_l^{-1} M^{-l} \\ &\leq c_2 \sum_{l=0}^{L} \big(2(L+1)c_2\epsilon^{-2}M^{-l}\big)^{-1} M^{-l} \\ &= c_2 \sum_{l=0}^{L} \frac{\epsilon^2}{2(L+1)c_2} = \frac{1}{2}\epsilon^2.\end{aligned}$$

To bound the complexity $C$, let us first find the bound for $L$ in terms of $\log \epsilon^{-1}$. Indeed, $L+1$, defined in (2.11) is bounded by

$$L+1 \leq \frac{2\log(\epsilon^{-1})}{\log M} + \frac{\log(2c_1^2)}{\log M} + 2 \leq c_5 \log \epsilon^{-1}, \tag{2.13}$$

where $c_5 = \frac{1+\big(0\vee\log(2c_1^2)\big)}{\log M} + 2$ given that $\log \epsilon^{-1} > 1$ ($\epsilon \leq e^{-1}$). Besides, from (2.11) we can get an upper bound for $M^{L-1}$ as

$$M^{L-1} \leq M^{\frac{\log(2c_1^2\epsilon^{-2})}{\log M}} = e^{\log M \frac{\log(2c_1^2\epsilon^{-2})}{\log M}} = 2c_1^2\epsilon^{-2}. \tag{2.14}$$

Therefore the computational complexity $C$ is bounded through

$$C \le c_3 \sum_{l=0}^{L} N_l M^l \le c_3 \sum_{l=0}^{L} \big(2(L+1)c_2\epsilon^{-2}M^{-l}+1\big)M^l$$
$$= c_3\Big(2(L+1)^2 c_2\epsilon^{-2} + \frac{M^2M^{L-1}-1}{M-1}\Big) \le c_4\epsilon^{-2}(\log\epsilon)^2,$$

where $c_4 = 2c_2c_3c_5^2 + \frac{2c_3c_1^2M^2}{M-1}$.

2. For both estimators $\hat{Y}$ and $\hat{P}$, the bias is fixed for the same choice of $L$ in (2.11). Now let us choose an appropriate $N$ such that $\mathbb{V}[\hat{P}] \le \frac{1}{2}\epsilon^2$. Let $N = \lceil 2c_2\epsilon^{-2}\rceil$ to meet the accuracy specification, and recall the upper bound for $M^{L-1}$ in (2.14). Then the complexity $C(\hat{P})$ is

$$C(\hat{P}) \le c_3 N M^L \le c_3(2c_2\epsilon^{-2}+1)M^2 2c_1^2\epsilon^{-2} \le c_6\epsilon^{-4},$$

where $c_6 = 2c_1^2c_3M^2(2c_2+e^{-2})$. ∎

The application of Theorem 2.3 relies on its conditions being verified. This is explored in the form of the following corollary.

COROLLARY 2.4. *Assume the setting in Theorem 2.3 and choose a uniform sampling distribution $\xi^u$ as in (2.4). Then we have*

1. $c_1 = C_f^2(n\nu+\mu)$,
2. $c_2 = 2C_f^2(M+1)(n\nu+\mu) + 2\mathbb{V}_{\mathbf{a},\mathbf{b}}[P]$,
3. $c_3 = 1 + M^{-1}$.

*Proof.* 1. For any $l \in \mathbb{N}\bigcup\{0\}$ we have that

$$\big(\mathbb{E}[f(\mathbf{a}^T\mathbf{b})] - \mathbb{E}[f(X_l(\xi^u))]\big)^2 \le \mathbb{E}[\big(f(\mathbf{a}^T\mathbf{b}) - f(X_l(\xi^u))\big)^2]$$
$$\le C_f^2\mathbb{E}[|\mathbf{a}^T(I - S_lS_l^T)\mathbf{b}|^2] \le C_f^2M^{-l}(n\nu+\mu),$$

where the last inequality holds because of (2.5).

2. For any $l > 0$ we have that

$$\mathbb{V}[\hat{P}_l - \hat{P}_{l-1}] \le \big(\mathbb{V}[\hat{P}_l - P]^{\frac{1}{2}} + \mathbb{V}[P - \hat{P}_{l-1}]^{\frac{1}{2}}\big)^2$$
$$\le \big(\mathbb{E}[(\hat{P}_l - P)^2]^{\frac{1}{2}} + \mathbb{E}[(\hat{P}_{l-1} - P)^2]^{\frac{1}{2}}\big)^2$$
$$\le C_f^2\big(\mathbb{E}[|\mathbf{a}^T(I - S_lS_l^T)\mathbf{b}|^2]^{\frac{1}{2}} + \mathbb{E}[|\mathbf{a}^T(I - S_{l-1}S_{l-1}^T)\mathbf{b}|^2]^{\frac{1}{2}}\big)^2$$
$$\le 2C_f^2(M^{-l} + M^{-l+1})(n\nu+\mu) \le 2C_f^2(M+1)(n\nu+\mu)M^{-l}.$$

For $l = 0$ we have that

$$\mathbb{V}[\hat{P}_0] = \mathbb{V}[f(X_0(\xi))] \le \big(\mathbb{V}[f(X_0(\xi)) - P]^{\frac{1}{2}} + \mathbb{V}[P]^{\frac{1}{2}}\big)^2$$
$$\le \big(C_f\mathbb{E}[|\mathbf{a}^T(I - S_0S_0^T)\mathbf{b}|^2]^{\frac{1}{2}} + \mathbb{V}_{\mathbf{a},\mathbf{b}}[P]^{\frac{1}{2}}\big)^2$$
$$\le \big(C_f(n\nu+\mu)^{\frac{1}{2}} + \mathbb{V}_{\mathbf{a},\mathbf{b}}[P]^{\frac{1}{2}}\big)^2$$
$$\le 2C_f^2(n\nu+\mu) + 2\mathbb{V}_{\mathbf{a},\mathbf{b}}[P].$$

Besides, from (2.8) we can see that $\mathbb{V}[\hat{P}]$ is bounded by the same $c_2$.

3. For any $l > 0$ we can see easily the complexity is roughly

$$C_l \leq N^l(M^l + M^{l-1}) = (1 + M^{-1})N^1 M^l,$$

while

$$C_0 \leq N^0 M^0 \leq (1 + M^{-1})N^0 M^0.$$

Besides, we have for the complexity of $\hat{P}$ that

$$C(\hat{P}) \leq NM^L \leq (1 + M^{-1})NM^L.$$

Thus $c_3$ can be set as $1 + M^{-1}$. ∎

*Remark* 2.5. Asymptotically as $l \to \infty$, we have that $\mathbb{E}[P - \hat{P}_l] \approx c_1 M^{-\frac{l}{2}}$, and hence

$$\mathbb{E}[\hat{P}_l - \hat{P}_{l-1}] \approx (\sqrt{M} - 1)c_1 M^{-\frac{l}{2}} \approx (\sqrt{M} - 1)\mathbb{E}[P - \hat{P}_l].$$

Similarly to the analysis in Section 4.2 of [5], this information can be used as an approximate bound: $L$ can be set as the smallest integer such that

$$|\hat{Y}_L| < \frac{1}{\sqrt{2}}(\sqrt{M} - 1)\epsilon. \tag{2.15}$$

By doing this, we might achieve a bias bounded by $\frac{\epsilon^2}{2}$ without evaluating $c_1$.

*Remark* 2.6 (Optimal $N_l$). To achieve a fixed variance, i.e., $\mathbb{V}[\hat{Y}] < \frac{1}{2}\epsilon$, the optimal $N_l$ can be chosen as

$$N_l \approx \left\lceil 2\epsilon^{-2}\sqrt{V_l M^{-l}}\Big(\sum_{j=0}^{L}\sqrt{V_l M^l}\Big)\right\rceil, \tag{2.16}$$

where $V_l$ denotes the variance of a single sample $\hat{P}_l - \hat{P}_{l-1}$. This result is simply an application of Section 1.3 of [7] or Eqn. (12) in [5] to the 'stepsize' $M^{-l}$. The estimation for $N_l$ in (2.16) is conservative and may induce oversampling. In practice, some scaling factor might be introduced to avoid oversampling (see Section 4.1).

**2.1. Optimal $M$.** This part explores the methods in [5] in order to find an optimal $M$ that reduces the computational complexity of the estimator even further. With $c_2$ given by Corollary 2.4, $L$ and $N_l$ given in the proof of Theorem 2.3, we can express the complexity of $\hat{Y}$ in terms of $M$ as

$$\begin{aligned} C(\hat{Y}) &\leq \sum_{l=0}^{L} C_l \approx \sum_{l=0}^{L} N_l(M^l + M^{l-1}) \overset{(2.12)}{\approx} \sum_{l=0}^{L} c_2(L+1)(M^l + M^{l-1})\epsilon^{-2}M^{-l} \\ &\overset{c_2}{\approx} \sum_{l=0}^{L}(L+1)(M+1)^2 M^{-1}\epsilon^{-2} = (L+1)^2(M+1)^2 M^{-1}\epsilon^{-2} \\ &\overset{(2.11)}{\approx} M^{-1}(M+1)^2 \log(M)^{-2}\log(\epsilon)^2\epsilon^{-2} = g(M)\log(\epsilon)^2\epsilon^{-2}, \end{aligned}$$

where

$$g(M) := M^{-1}(M+1)^2 \log(M)^{-2}. \tag{2.17}$$

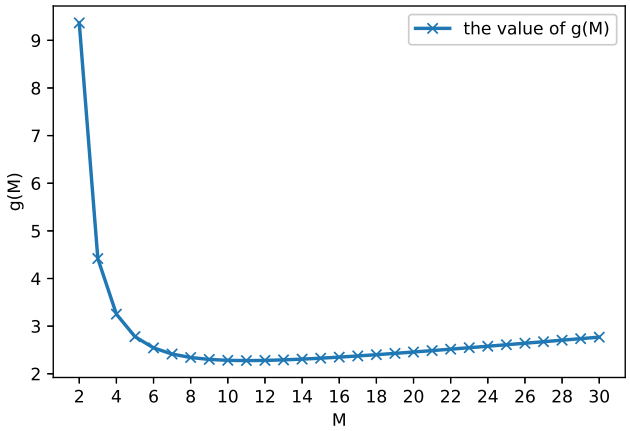


FIG. 1. *The plot of the dominant complexity term $g(M)$ against the base number $M$, indicating the existence of an optimal $M$ at the minimum point.*

As illustrated in figure 1 where we plot $g(M)$ against $M$, $g(M)$ drops sharply for $M < 6$ and then starts growing slightly again after $M$ going beyond 12. The minimum (optimum) is attained at $M = 11$, however from our experience using either $M = 10$ or $M = 12$ does not make a significant difference. We remark that our definition of $g(M)$ in (2.17) differs somewhat from that used in [5], i.e. in the term $(M+1)^2$, but this does not affect the general trend of $g(M)$ as described above. In the numerical experiments of Section 4.1, a choice of $M = 10$ is used as it was deemed appealing in terms of both the performance and time cost.

**3. Matrix multiplication.** We now extend our approach to matrix multiplication and thus we consider $A(t)$ and $B(t)$ to be two random matrices of size $m \times n$ and $n \times d$ respectively, $m, n, d \in \mathbb{N}$, drawn from different distributions, elementwise, in the sense $A(t) \sim \mathcal{L}_{A(t)}$ and $B(t) \sim \mathcal{L}_{B(t)}$, and again we suppress $t$ in the notation as in Section 2 and assume that $n$ is extremely large such that computing directly $AB$ is prohibitively expensive. Recall that $f$ is a Lipchitz function with Lipschitz constant $C_f$, and define $f^{\odot}(AB)$ the elementwise operator on $AB$, that is,

$$\big(f^{\odot}(AB)\big)_{ik} = f\big((AB)_{ik}\big), \quad \text{for } i \in [m], \text{ and } k \in [d].$$

Once again consider that there is a need to compute $\mathbb{E}_{A,B}[f^{\odot}(AB)]$ where $\mathbb{E}_{A,B}$ is the expectation under $\mathcal{L}_A$ and $\mathcal{L}_B$.

As in the inner product case, in order to simulate $\mathbb{E}_{A,B}[f^{\odot}(AB)]$ we first approximate $AB$ by random sampling (sketching) for matrix multiplication and then approximate the expectation through a Monte Carlo method. Recall that $\xi := \{\xi_j\}_{j=1}^n$ with $\sum_{j=1}^n \xi_j = 1$ is a sampling probability such that an index $j \in [n]$ can be drawn with positive probability $\xi_j$ and $S_L$ a sampling-and-rescaling matrix of size $n \times M^L$ such that

$$\widehat{AB} = AS_L S_L^T B = \frac{1}{M^L} \sum_{i=1}^{M^L} \frac{1}{\xi_{r_i}} A_{:,r_i} B_{r_i,:} := Z_L(\xi), \tag{3.1}$$

where $A_{:,j}$ indicates the $j$th column of $A$ and $B_{j,:}$ indicates the $j$th row of $B$, and $Z_L(\xi)$ denotes the matrix-valued random variable that approximates $AB$ based on $M^L$ indices sampled from $\xi$. It is easy to verify that $Z_L(\xi)$ is an unbiased estimator under the sampling distribution $\xi$. Besides, following arguments similar to those in the proof of Theorem 2.1 in [4] and Lemma 4 in [3], we can conclude that the minimum of the expected squared Frobenius error can be achieved by the following result.

THEOREM 3.1. *If the matrix elements $A_{ij}$ and $B_{jk}$ are independent random variables, $i \in [m]$, $j \in [n]$ and $k \in [d]$, with finite and nonzero moments $\mathbb{E}_A[\|A_{:,j}\|_2^2]$ and $\mathbb{E}_B[\|B_{j,:}\|_2^2]$. Then the probability $\xi^{**}$, which is defined as*

$$\xi_j^{**} = \frac{\sqrt{\mathbb{E}_A[\|A_{:,j}\|_2^2]\mathbb{E}_B[\|B_{j,:}\|_2^2]}}{\sum_{i=1}^n \sqrt{\mathbb{E}_A[\|A_{:,i}\|_2^2]\mathbb{E}_B[\|B_{i,:}\|_2^2]}}, \tag{3.2}$$

*minimizes the expected value of the variance in (2.1), that is,*

$$\begin{aligned} &\min_{\xi} \mathbb{E}_{A,B}\big[\boldsymbol{E}[\|AB - Z_L(\xi)\|_F^2]\big] = \mathbb{E}_{A,B}\big[\boldsymbol{E}[\|AB - Z_L(\xi^{**})\|_F^2]\big] \\ &= \frac{1}{M^L}\Big(\Big(\sum_{j=1}^n \sqrt{\mathbb{E}_A[\|A_{:,j}\|_2^2]\mathbb{E}_B[\|B_{j,:}\|_2^2]}\Big)^2 - \mathbb{E}_{A.B}[\|AB\|_F^2]\Big) := \frac{\bar{\mu}}{M^L}, \end{aligned} \tag{3.3}$$

*where $\bar{\mu} = \Big(\sum_{j=1}^n \sqrt{\mathbb{E}_A[\|A_{:,j}\|_2^2]\mathbb{E}_B[\|B_{j,:}\|_2^2]}\Big)^2 - \mathbb{E}_{A,B}[\|AB\|_F^2]$, $\boldsymbol{E}[\cdot]$ denotes the expectation under the distribution $\xi$, and $\mathbb{E}_{A,B}[\cdot]$ is the expectation with respect to the (element-wise) probabilities of $A$ and $B$.*

The proof is omitted here as it is quite similar to the proof of Theorem 2.1 in [4]. Besides, as discussed in Section 2, it is impractical to use $\xi^*$ for random sampling. A simpler option would be to use a uniform probability $\xi^u$ as defined in (2.4).

THEOREM 3.2. *Assume the same setting as in Theorem 3.1 but with probability $\xi^u$ as defined in (2.4), then the expected squared Frobenius error is*

$$\mathbb{E}_{A,B}\big[\boldsymbol{E}[\|Z_L(\xi^u) - AB\|_F^2]\big] = \mathbb{E}_{A,B}\big[\boldsymbol{E}[\|Z_L(\xi^{**}) - AB\|_F^2]\big] + \frac{n\bar{\nu}}{M^l} = \frac{n\bar{\nu} + \bar{\mu}}{M^l}, \tag{3.4}$$

*where*

$$\bar{\nu} = \sum_{i=1}^n \Big(\sqrt{\mathbb{E}_A[\|A_{:,i}\|_2^2]\mathbb{E}_B[\|B_{i,:}\|_2^2]} - \frac{1}{n}\sum_{j=1}^n \sqrt{\mathbb{E}_A[\|A_{:,j}\|_2^2]\mathbb{E}_B[\|B_{j,:}\|_2^2]}\Big)^2.$$

The proof is omitted here as it is very similar to that of Theorem 2.3.

In this context, a quantity of interest $P$ can be approximated with standard MC coupled to a random sampling method for matrix multiplication via either uniform probability (2.4) or the optimal probability (3.2) (if tractable)

$$\mathbb{E}_{A,B}[P] := \mathbb{E}_{A,B}[f^{\odot}(AB)] \approx \frac{1}{N}\sum_{j=1}^N f(AS_L^{(j)}(S_L^{(j)})^T B) = \frac{1}{N}\sum_{j=1}^N f\big(Z_L^{(j)}(\xi)\big) := \hat{P}, \tag{3.5}$$

where $N$ is the number of realisations for $M^L$ many index samples. To consider the MSE for the estimate $\hat{P}$, we apply a matrix *vectorization*: for instance, if $A \in \mathbb{R}^{m\times n}$,

$$\mathrm{vec}(A) = \mathrm{vec}([A_{:,1} \quad \cdots \quad A_{:,n}]) = \begin{bmatrix} A_{:,1} \\ \vdots \\ A_{:,n} \end{bmatrix} \in \mathbb{R}^{mn}, \tag{3.6}$$

is the column concatenation of $A$ into a vector. Then the MSE would be

$$\begin{aligned}\mathbb{E}\big[\|\mathrm{vec}(\hat{P}-\mathbb{E}[P])\|_2^2\big] &= \mathbb{E}\big[\|\mathrm{vec}(\mathbb{E}[\hat{P}]-\mathbb{E}[P])\|_2^2\big] + \mathbb{E}\big[\|\mathrm{vec}(\hat{P}-\mathbb{E}[\hat{P}])\|_2^2\big] \\ &= \|\mathrm{vec}\big(\mathbb{E}[A(I-S_LS_L^T)B]\big)\|_2^2 + \mathbb{E}\big[\|\mathrm{vec}\big(\hat{P}-\mathbb{E}[\hat{P}]\big)\|_2^2\big] \\ &= \|\mathrm{vec}\big(\mathbb{E}[A(I-S_LS_L^T)B]\big)\|_2^2 + \mathbb{V}_{\|}\big[\mathrm{vec}(\hat{P})\big],\end{aligned} \tag{3.7}$$

where $\mathbb{E}$ is short for $\mathbb{E}_{A,B,\xi}$ and $\mathbb{V}_{\|}\big[\mathrm{vec}(X)\big] := \mathbb{E}\big[\|\mathrm{vec}\big(X-\mathbb{E}[X]\big)\|_2^2\big]$ for any random matrix $X$. Besides, it is easy to verify that

$$\mathbb{V}_{\|}[X+Y]^{\frac{1}{2}} \le \mathbb{V}_{\|}[X]^{\frac{1}{2}} + \mathbb{V}_{\|}[Y]^{\frac{1}{2}}, \tag{3.8}$$

for any random vectors $X$ and $Y$. Note that the variance of a vectorized random matrix is indeed the variance of the random matrix in Frobenius norm. For example,

$$\begin{aligned}\mathbb{V}_{\|}\big[\mathrm{vec}(\hat{P})\big] &= \mathbb{E}\big[\|\mathrm{vec}\big(\hat{P}-\mathbb{E}[\hat{P}]\big)\|_2^2\big] = \mathbb{E}\big[\sum_{h=1}^{md}\mathrm{vec}\big(\hat{P}-\mathbb{E}[\hat{P}]\big)_h^2\big] \\ &= \mathbb{E}\big[\sum_{i=1}^{m}\sum_{k=1}^{d}\big(\hat{P}-\mathbb{E}[\hat{P}]\big)_{ik}^2\big] = \mathbb{E}[\big\|\hat{P}-\mathbb{E}[\hat{P}]\big\|_F^2].\end{aligned}$$

With these preliminaries let us now extend the approach of Section 3.1 to matrix multiplication. For $l \in [L]\bigcup\{0\}$, define $\hat{P}_l$ as the approximation to $f^{\odot}(AB)$ with $M^l$ many index samples. Recall that $\hat{Y}_l$ is an estimator of $\mathbb{E}[\hat{P}_l-\hat{P}_{l-1}]$ using $N_l$ realizations with $l>0$ and $\hat{Y}_0$ the respective estimator of $\mathbb{E}[\hat{P}_0]$ using $N_0$ samples, as defined in (2.9). Eqn. (2.10) remains unchanged, from where we have that $\mathbb{E}[\hat{P}_L] = \mathbb{E}[\hat{Y}]$.

THEOREM 3.3. *Let $A$ and $B$ be two random matrices with sizes $m\times n$ and $n\times d$ respectively, drawn from different distributions, namely $A\sim\mathcal{L}_A$ and $B\sim\mathcal{L}_B$. Let $f:\mathbb{R}\to\mathbb{R}$ be a Lipchitz function with Lipchitz number $C_f$. Denote by $P$ the term of interest as in (3.5). Define $\hat{P}_\ell$ the corresponding approximation to $f^{\odot}(AB)$ based on the sketched version of matrix multiplication via $M^\ell$ many index samples like in (3.1).*

1. *If there exist independent estimators $\hat{Y}_l$ as in (2.9) based on $N_l$ Monte Carlo samples, and positive constants $c_1$, $c_2$, $c_3$ such that*
   (a) $\big\|\mathrm{vec}(\mathbb{E}[\hat{P}_l-P])\big\|_2^2 \le c_1^2M^{-l}$,
   (b) $\mathbb{V}_{\|}[\mathrm{vec}(\hat{Y}_l)] \le c_2N_l^{-1}M^{-l}$,
   (c) *the complexity of $\hat{Y}_l$, denoted by $C_l$, is bounded by $C_l \le c_3N_lM^l$,*

   *then there exists a positive constant $c_4$ such that for $\epsilon < e^{-1}$, there are values $L$ and $N_l$ for which the multilevel estimator $\hat{Y}=\sum_{l=0}^{L}\hat{Y}_l$ has an MSE $\mathbb{E}[\|\mathrm{vec}(\hat{Y}-\mathbb{E}[P])\|_2^2]$ with bound $\epsilon^2$, with computational complexity*

$$C(\hat{Y}) := \sum_{l=0}^{L} C_l \le c_4\epsilon^{-2}(\log\epsilon)^2.$$

2. *Furthermore, define the estimator based on the finest level $L$ and $N$ realizations as in (2.6) with either the uniform probability (2.4) or the optimal probability (3.2) (if approachable). Suppose*
   (a) *the variance for $\hat{P}$ is bounded by the same constant $c_2$, i.e., $\mathbb{V}_{\|}[\hat{P}] \le c_2N^{-1}$,*

(b) *the complexity for* $\hat{P}$ *is bounded by the same constant* $c_3$*, i.e.,* $C(\hat{P}) \leq c_3 N M^L$,

*then with the same accuracy* $\epsilon^2$, $C(\hat{P}) \leq c_6 \epsilon^{-4}$ *which is much larger than the bound of* $C(\hat{Y})$.

The proof is similar to that of Theorem 2.3, expect from the decomposition of MSE,

$$\mathbb{E}\big[\|\mathrm{vec}(\hat{Y} - \mathbb{E}[P])\|_2^2\big] = \mathbb{E}\big[\|\mathrm{vec}(\mathbb{E}[\hat{Y} - P])\|_2^2\big] + \mathbb{V}_{\|}\big[\mathrm{vec}(\hat{Y})\big], \tag{3.9}$$

so we omit the proof. A more important issue is to verify our proposed MLMC estimator satisfies the conditions of Theorem 3.3.

COROLLARY 3.4. *Assume the same setting in Theorem 3.3 via the sampling distribution* $\xi^u$ *in (2.4). Then we have*

1. $c_1 = C_f^2(n\bar{\nu} + \mu)$,
2. $c_2 = 2C_f^2(M+1)(n\bar{\nu} + \mu) + 2\mathbb{V}_{\|}[f^{\odot}(AB)]$,
3. $c_3 = md(1 + M^{-1})$.

*Proof.* 1. For any $l \in \mathbb{N}$ we have that

$$\begin{aligned}
\big\|\mathrm{vec}\big(\mathbb{E}[f^{\odot}(AB) - f^{\odot}(Z_l(\xi^u))]\big)\big\|_2^2 &\leq \mathbb{E}\big[\big\|\mathrm{vec}\big(f^{\odot}(AB) - f^{\odot}(Z_l(\xi^u))\big)\big\|_2^2\big] \\
&= \mathbb{E}\big[\big\|f^{\odot}(AB) - f^{\odot}(Z_l(\xi^u))\big\|_F^2\big] \\
&\leq C_f^2 \mathbb{E}[\|AB - Z_l(\xi^u)\|_F^2] \\
&\leq C_f^2 M^{-l}(n\bar{\nu} + \mu),
\end{aligned}$$

where the last inequality comes from Theorem 3.2.

2. For any $l > 0$ we have that

$$\begin{aligned}
&\mathbb{V}_{\|}[\mathrm{vec}(\hat{P}_l - \hat{P}_{l-1})] \\
&\leq \Big(\mathbb{V}_{\|}\big[\mathrm{vec}\big(\hat{P}_l - f^{\odot}(AB)\big)\big]^{\frac{1}{2}} + \mathbb{V}_{\|}\big[\mathrm{vec}\big(\hat{P}_{l-1} - f^{\odot}(AB)\big)\big]^{\frac{1}{2}}\Big)^2 \\
&\leq \Big(\mathbb{E}\big[\big\|\mathrm{vec}\big(f^{\odot}(AB) - f^{\odot}(Z_l(\xi^u))\big)\big\|_2^2\big]^{\frac{1}{2}} \\
&\quad + \mathbb{E}\big[\big\|\mathrm{vec}\big(f^{\odot}(AB) - f^{\odot}(Z_{l-1}(\xi^u))\big)\big\|_2^2\big]^{\frac{1}{2}}\Big)^2 \\
&\leq 2C_f^2\big(\mathbb{E}[\|AB - Z_l(\xi^u)\|_F^2]^{\frac{1}{2}} + \mathbb{E}[\|AB - Z_{l-1}(\xi^u)\|_F^2]^{\frac{1}{2}}\big)^2 \\
&\leq 2C_f^2(M^{-l} + M^{-l+1})(n\bar{\nu} + \mu) \\
&\leq 2C_f^2(M+1)(n\bar{\nu} + \mu)M^{-l}.
\end{aligned}$$

For $l = 0$ we have that

$$\begin{aligned}
&\mathbb{V}_{\|}[\mathrm{vec}(\hat{P}_0)] = \mathbb{V}_{\|}\big[\mathrm{vec}\big(f^{\odot}(Z_0(\xi^u))\big)\big] \\
&\leq \Big(\mathbb{V}_{\|}\big[\mathrm{vec}\big(f^{\odot}(Z_0(\xi^u)) - f^{\odot}(AB)\big)\big]^{\frac{1}{2}} + \mathbb{V}_{\|}\big[\mathrm{vec}\big(f^{\odot}(AB)\big)\big]^{\frac{1}{2}}\Big)^2 \\
&\leq 2C_f^2 \mathbb{E}[\|A(I - S_0 S_0^T)B\|_F^2] + 2\mathbb{V}_{\|}\big[\mathrm{vec}\big(f^{\odot}(AB)\big)\big] \\
&\leq 2C_f^2(n\bar{\nu} + \mu) + 2\mathbb{V}_{\|}\big[\mathrm{vec}\big(f^{\odot}(AB)\big)\big].
\end{aligned}$$

Besides, it is easy to see that $\mathbb{V}_{\|}[\mathrm{vec}(\hat{P})]$ can be bounded by the same $c_2$ together with $N^{-1}$.

3. For any $l > 0$ we can see easily the complexity is roughly

$$C_l \leq mdN^l(M^l + M^{l-1}) = md(1 + M^{-1})N^1M^l,$$

while

$$C_0 \leq mdN^0M^0 \leq md(1 + M^{-1})N^0M^0.$$

Besides, we have for the complexity of $\hat{P}$ that

$$C(\hat{P}) \leq mdNM^L \leq md(1 + M^{-1})NM^L.$$

Thus $c_3$ can be set as $md(1 + M^{-1})$. ∎

*Remark* 3.5. Asymptotically as $l \to \infty$, we have that $\|\text{vec}(\mathbb{E}[P - \hat{P}_l])\|_2 \approx c_1 M^{-\frac{l}{2}}$, and hence

$$\|\text{vec}(\mathbb{E}[\hat{P}_l - \hat{P}_{l-1}])\|_2 \approx (\sqrt{M} + 1)c_1 M^{-\frac{l}{2}} \approx (\sqrt{M} + 1)\|\text{vec}(\mathbb{E}[P - \hat{P}_l])\|_2.$$

Similarly as in Section 4.2 of [5], this information can be used as an approximate bound: $L$ can be set as the smallest integer such that

$$\|\text{vec}(\hat{Y}_L)\|_2 < \frac{1}{\sqrt{2}}(\sqrt{M} + 1)\epsilon. \tag{3.10}$$

By doing this, we might achieve a bias bounded by $\frac{\epsilon^2}{2}$ without evaluating $c_1$.

*Remark* 3.6 (Optimal $N_l$). To achieve a fixed variance, i.e., $\mathbb{V}_{\|}[\text{vec}(\hat{Y})] < \frac{1}{2}\epsilon$, the optimal $N_l$ can be chosen as

$$N_l \approx \left\lceil 2\epsilon^{-2}\sqrt{V_l M^{-l}}\Big(\sum_{j=0}^{L}\sqrt{V_l M^l}\Big)\right\rceil, \tag{3.11}$$

where $V_l$ is the variance of the vectorized form of a single sample $\hat{P}_l - \hat{P}_{l-1}$ (recall the definition of the vectorized matrix variance right before (3.8)). This result is simply an application of Section 1.3 of [7] or Eqn. (12) in [5] to the 'stepsize' $M^{-l}$.

To choose the optimal value of $M$ we argue as in Section 2.1, that is, $M = 11$ leads to the least computational complexity among between all choices of $M$. In the numerical experiments of Section 4.2, it turns out that $M = 10$ and $M = 6$ both yield acceptable approximations.

**3.1. MLMC sketching algorithm.** Based on the general discussion in the beginning of Section 2, Remark 3.5 and Remark 3.6, we propose an algorithm for estimating the matrix product based on MLMC method in Algorithm 3.2. The inner product case discussed in Section 2 can be treated as a special case. Algorithm 3.2 approximates $\mathbb{E}[f^{\odot}(AB)]$ through (2.10) under uniform probability (2.4), where the evaluation for each $\hat{Y}_l$ in (2.10) is performed through function *level_estimation* described in Algorithm 3.1. To ensure the convergence, the choices of $L$ and $N_l$ with $l \in [L]\bigcup\{0\}$ are determined within Algorithm 3.2 using a *while* loop with one of the conditions given by Eqn. (3.10) in Remark 3.5[1]. It is worth noticing that, while the

[1]The condition for inner product is slightly different, see Eqn. (2.15) in Remark 2.5.

value $L$ and therefore $N_l$ for $l \in [L] \bigcup \{0\}$ are updated in the while loop (see Line 16 and Line 9), previous evaluations for $\hat{Y}_l$ are reused in Line 13 for efficiency.

Although the outline in Algorithm 3.1 is simple to follow we draw the reader's attention to Line 17 describing how $\hat{P}_l^{(k)}$ and $\hat{P}_{l-1}^{(k)}$ are computed through the common realization of $M^l$ indices. Indeed, the procedure for getting $\hat{P}_l^{(k)}$ is by random sampling as in (3.1) via the indices of a sample realization of size $M^l$ under uniform probability, and likewise $\hat{P}_{l-1}^{(k)}$ via $M^{l-1}$ of those $M^l$ indices. That is, taking $(r_1, \ldots, r_{M^l})$ as a realization, then

$$\hat{P}_l^{(k)} = f^{\odot}\Big(\frac{n}{M^l}\sum_{j=1}^{M^l} A^{(k)}_{:,r_j} B^{(k)}_{r_j,:}\Big), \quad \text{and} \quad \hat{P}_{l-1}^{(k)} = f^{\odot}\Big(\frac{n}{M^{l-1}}\sum_{j=1}^{M^{l-1}} A^{(k)}_{:,r_{jM}} B^{(k)}_{r_{jM},:}\Big).$$

Note that in practice the above computation can be further simplified by mapping $(r_1, \ldots, r_{M^l})$ into a set with non-repeated elements.

**Algorithm 3.1** function *level_estimation*.

1: **Pre-defined:** $\mathcal{L}_{\mathbf{A}}$ and $\mathcal{L}_{\mathbf{B}}$, the distributions of the targeted random matrices.
2: $f$, the targeted function; $\xi^u$, the uniform sampling distribution defined in (2.4).
3: **input:** $l$, the level size;
4: $M$, the base number;
5: $N_l$, the number of iterations.
6: **output:** $\hat{Y}_l$, the approximated version of $\mathbb{E}[\hat{P}_l - \hat{P}_{l-1}]$ for $l \neq 0$ or $\mathbb{E}[\hat{P}_l]$ for $l = 0$.
7: **initialization**: $\hat{Y}_l = 0$.
8: **if** $l = 0$ **then**
9: **for** $\ell = 1 \cdots N_l$ **do**
10: get a pair of samples $A^{(\ell)}$ and $B^{(\ell)}$ from $\mathcal{L}_{\mathbf{A}}$ and $\mathcal{L}_{\mathbf{B}}$;
11: sample one index $r$ from 1 to $n$ according to $\xi^u$;
12: set $\hat{Y}_l = \hat{Y}_l + \frac{1}{N_0} f^{\odot}\big(nA^{(\ell)}_{:,r} B^{(\ell)}_{r,:}\big)$;
13: **else**
14: **for** $k = 1 \cdots N_l$ **do**
15: get a pair of samples $A^{(k)}$ and $B^{(k)}$ from $\mathcal{L}_{\mathbf{A}}$ and $\mathcal{L}_{\mathbf{B}}$;
16: sample $M^l$ many indices $(r_j)_{j=1}^{M^l}$ from 1 to $n$ according to $\xi^u$;
17: set

$$\hat{Y}_l = \hat{Y}_l + \frac{1}{N_l}\Big(f^{\odot}\Big(\frac{n}{M^l}\sum_{j=1}^{M^l} A^{(k)}_{:,r_j} B^{(k)}_{r_j,:}\Big) - f^{\odot}\Big(\frac{n}{M^{l-1}}\sum_{j=1}^{M^{l-1}} A^{(k)}_{:,r_{jM}} B^{(k)}_{r_{jM},:}\Big)\Big);$$

18: **return**: $\hat{Y}_l$.

**4. Numerical experiments.** In this part, we present some numerical experiments designed to test the performance of the Algorithm 3.2 in comparison with a standard MC method embedded with the optimal sampling distribution (see Theorem 2.1 and Theorem 3.1). Our experiments are implemented in Python (version 3.6.9) with Numpy-based calculations being optimized under openBLAS [14] and executed on a Linux cluster with two 14-core E5-2690 v4 Intel Xeon CPUs at 2.60GHz and non-uniform memory allocation.

**Algorithm 3.2** The MLMC estimator for $\mathbb{E}[f^{\odot}(AB)]$.

**Pre-defined:** $\mathcal{L}_{\mathbf{A}}$ and $\mathcal{L}_{\mathbf{B}}$, the distributions of the targeted random vectors.
  $f$: the targeted function; $\xi^u$: the uniform distribution in (2.4).
**input:** $M$, the base number;
  $\epsilon$, the error tolerance.
**output:** $\hat{Y}$, the approximated version of $\mathbb{E}[f^{\odot}(AB)]$.
**initialization**: set $L = 0$, $t = 0$.
**while** $L < 3$ or $\frac{\|\hat{Y}_{L-1}\|_F}{N_{L-1}^{(t-1)}} \geq \frac{1}{\sqrt{2}}(\sqrt{M}+1)\epsilon$ **do**
  initialize $\hat{Y}_L = 0$;
  update $V_l$ (defined in Remark 3.6) for all $l \in [L]\bigcup\{0\}$;
  calculate the optimal $N_l^{(t)}$ for all $l \in [L]\bigcup\{0\}$ through Eqn.(3.11);
  update $\hat{Y}_L = \hat{Y}_L + N_L^{(t)} \mathit{level\_estimation}(L, M, N_L^{(t)})$;
  **if** $L > 0$ **then**
    **for** $l = 0 \cdots L-1$ **do**
      update $\hat{Y}_l = \hat{Y}_l + (N_l^{(t)} - N_l^{(t-1)})\mathit{level\_estimation}(l, M, N_l^{(t)} - N_l^{(t-1)})$;
  set $L = L+1$ and $t = t+1$;
update $\hat{Y}_l = \hat{Y}_l / N_l^{(t-1)}$ for all $l \in [L-1]\bigcup\{0\}$;
**return**: $\sum_{l=0}^{L-1} \hat{Y}_l$.

**4.1. Example for the inner product.** Set $n = 10^4$ with $\mathbf{a}_j \sim \frac{j}{50}(0.4 - N(0,1))$ and $\mathbf{b}_j \sim \cos\big(\mathrm{Poi}(10)+2\mathrm{Exp}(1)\big)\mathrm{Bern}(0.05)$, $j \in [n]$, where $\mathrm{Poi}(\lambda)$ is a Poisson random variable with parameter $\lambda$, $\mathrm{Exp}(\alpha)$ is an exponential random variable with parameter $\alpha$, and $\mathrm{Bern}(\beta)$ is a Bernoulli random variable with success rate $\beta$. As the Bernoulli random variable has low success rate we expect $\mathbf{b}$ to be a sparse vector. In this example we targeted function is set to $f(x) := \frac{1}{|x|H(x+0.4)+0.01}$, where $H(\cdot)$ is an Heaviside step function. It is easy to see in this case $f$ is highly nonlinear.

We test the Algorithm 3.2 for the inner product case with a parameter $M = 10$ and error tolerance $\epsilon = 0.1$, where the reference solution is obtained through direct MC computation:

$$\mathbb{E}[\mathbf{a}^T\mathbf{b}] \approx \frac{1}{\mathcal{N}_1}\sum_{j=1}^{\mathcal{N}_1}(\mathbf{a}^{(j)})^T\mathbf{b}^{(j)}, \text{ with } \mathcal{N}_1 = 10^5. \tag{4.1}$$

The value of $L$ and the number of realizations at each level $l \leq L$, i.e., $N_l$ with $l \in [L]\bigcup\{0\}$ are tuned automatically by the algorithm itself. In our case $L = 3$ and $N_l$ is obtained through scaling (2.16) by $\frac{1}{20}$. This scaling factor $\frac{1}{20}$ is introduced to prevent oversampling. Note that the scaling factor does not affect the trend of $N_l$. Figure 2 illustrates the trend of the variance of each single path sample $\hat{P}_l - \hat{P}_{l-1}$ together with its corresponding $N_l$. From there it is easy to see that there is a clear decay in variance with respect to $l$ from $l = 1$, which results in the nearly polynomial decay in the number of realizations $N_l$. For comparison we also perform a standard MC simulation of the same $M$ and $L$ under *optimal sampling* (Theorem 2.1) with a number of repetitions chosen to maintain roughly the same accuracy level (convergence). The results obtained are tabulated in Table 1.

From Table 1, the MLMC estimator using $\xi^u$ in general outperforms the MC one using $\xi^*$ in terms of the elapsed time. Though MC using $\xi^u$ provides an approximation that doubles the accuracy of MLMC, its computational time is about six times

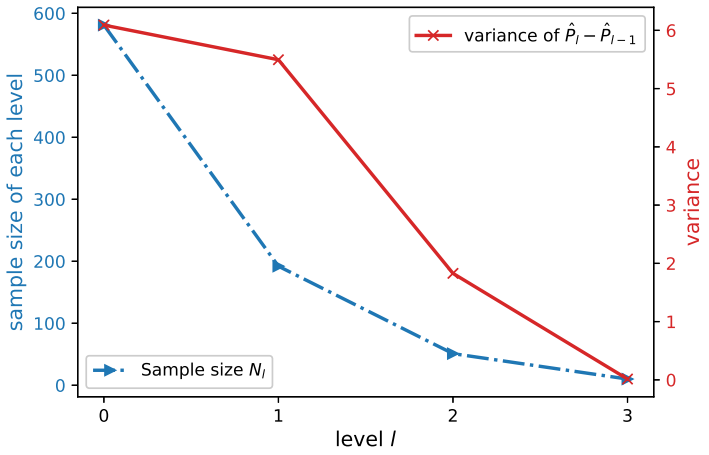


FIG. 2. *Inner product case: plot of the variance of a single realization $\hat{P}_l - \hat{P}_{l-1}$ up to $l = L$ (red solid line), and its corresponding number of realizations for each $l$ up to $l = L$ (blue dashed line): for $l = 0$, the variance of a single realization $\hat{P}_l - \hat{P}_{l-1}$ is indeed the variance of a single realization $\hat{P}_0$.*

| | MLMC using $\xi^u$ | | | MC using $\xi^*$ | | | directMC |
|---|---|---|---|---|---|---|---|
| $(M,L)$ | AE | RE | time cost | AE | RE | time cost | time cost |
| $(10,3)$ | 0.002 | 0.079 | 0.047 s | 0.001 | 0.041 | 0.274 s | 0.423 s |

TABLE 1

*Numerical results from the implementation of our method on approximating the inner product. These include records of the relative errors (RE), absolute errors (AE) and computational times for Algorithm 3.2 under $M = 10$ and its corresponding $L$. For comparison we provide also the results from standard MC (2.6) with optimal sampling distribution $\xi^*$ (2.2) based on the finest level $L$, and time cost for getting reference solution through direct MC (4.1).*

longer. The computation times of both estimators are less than that of the directMC for getting the reference solution, which illustrates the advantage of our proposed estimator in practice.

**4.2. Example for the matrix multiplication.** In the matrix multiplication case we consider a setup with $n = 10^4$, $m = d = 10^3$ using $A_{ij} \sim g_1\big(\frac{j}{10^4}(0.5 - N(0,1))\big)$, where $g_1(x) := \sin(x) + N(0,1)x$, and $B_{jk} \sim g_2(\text{Poi}(2))\text{Bern}(0.2)$, where $g_2(x) := \cos(x)H(5-x)$ for $i \in [m], j \in [n]$ and $k \in [d]$. Like before, $\text{Poi}(\lambda)$ denotes a Poisson random variable with parameter $\lambda$ and $\text{Bern}(\beta)$ is a Bernoulli random variable with success rate $\beta$. The targeted function is chosen to be $f(x) := |x|H(2-x)$, where $H(\cdot)$ is an Heaviside step function.

Similar to the inner product example, we run Algorithm 3.2 for the matrix product with base number $M = 10$ and the error tolerance $\epsilon = 0.1$. The reference solution is computed as

$$\mathbb{E}[AB] \approx \frac{1}{\mathcal{N}_2}\sum_{j=1}^{\mathcal{N}_2} A^{(j)}B^{(j)}, \text{ with } \mathcal{N}_2 = 10^5. \tag{4.2}$$

Algorithm 3.2 automatically chooses $L = 5$. Though $M^L$ is now larger than $n$, this

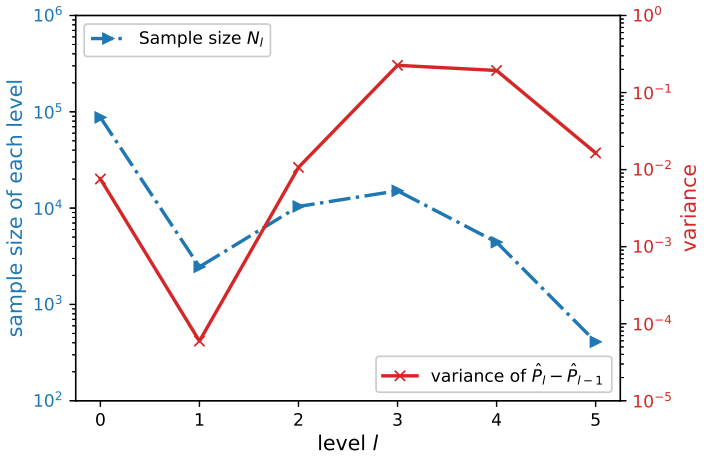


FIG. 3. *Matrix multiplication case: Plot of the variance of a single realization* $\hat{P}_l - \hat{P}_{l-1}$ *up to* $l = L$ *(red solid line), and its corresponding number of realizations for each* $l$ *up to* $l = L$ *(blue dashed line): for* $l = 0$*, the variance of a single realization* $\hat{P}_l - \hat{P}_{l-1}$ *is indeed the variance of a single realization* $\hat{P}_0$*.*

does not imply sampling all the columns of $A$. Besides, the number of realizations generated at the finest level $L$ is very small, which does not affect the total performance. Meanwhile the variance of each single path sample $\hat{P}_l - \hat{P}_{l-1}$ together with its corresponding $N_l$ is directly obtained through (3.11). Figure 3 illustrates the trend of the variance of $\hat{P}_l - \hat{P}_{l-1}$ and $N_l$ up to $l = L$. It can be noted from Figure 3 that from $l = 3$ the variance curve begins to decay, while the apparent low values in variance from $l = 0$ to $l = 2$ are due to the nature of randomised sketching of matrix multiplication. For example, to get an approximation $Z_l(\xi^u)$ of $AB$ through (3.1) with $l = 0$, only one column of A and the corresponding row of B are selected and multiplied. This is guaranteed to decrease the variance of $Z_0(\xi^u)$. We can also observe this phenomenon from the inner product case as depicted in Figure 2, where the variance of $l = 0$ is only slightly bigger than the one of $l = 1$. The curve in Figure 3 also indicates that our proposed estimator will be more efficient in super-large-scale matrix application, where the variance decay speeds up for higher level $l >> 5$.

To compare performance, a standard MC simulation of the same $M$ and $L$, formulated in (3.5), is implemented with optimal sampling distribution $\xi^{**}$ (Theorem 3.1) with the number of repetitions chosen to maintain roughly the same accuracy level. As matrix $B$ is a very sparse matrix, optimal probability defined in Eqn. 3.3 might have sparse or very small entries. Therefore even $M^L$ is now larger than $n$, the probability that all the columns of $A$ are sampled to obtain an approximation is pretty small. The results obtained are recorded in Table 2. The MLMC estimator using $\xi^u$ in general outperforms the MC one using $\xi^{**}$ in terms of the elapsed time. Meanwhile, the computational times for MC using $\xi^{**}$ are admittedly very large, taking three times longer than the directMC. This is mainly due to the choice of the high level $L = 5$ compared to the matrix size. On the other hand, it is reasonable to anticipate that the MLMC method under the approximated optimal probability instead of the uniform one, would lead to a drastic improvement of the efficiency of the approximation beyond what has been demonstrated in this work.

| | MLMC using $\xi^u$ | | | MC using $\xi^{**}$ | | | directMC |
|---|---|---|---|---|---|---|---|
| $(M,L)$ | AE | RE | time cost | AE | RE | time cost | time cost |
| $(10,5)$ | 0.088 | 0.006 | 2.240 s | 0.069 | 0.005 | 75.173 s | 25.561 s |

TABLE 2

*Numerical results from the implementation of our method on approximating the matrix product. These include records of the absolute errors in Frobenius norm (AE), the relative errors (RE) and computational times for Algorithm 3.2 under $M = 10$ and its corresponding $L$. For the sake of comparison we provide also the results from a standard MC simulation (3.5) based on the finest level $L$ and sampling distribution $\xi^{**}$ (3.2), and the time cost for getting the reference solution through direct MC (4.2)*

.

**5. Conclusions.** We presented a new approach for computing arbitrary vector and matrix products 'on-the-fly' that combines ideas from sketching in randomized linear algebra and multilevel Monte Carlo approaches for estimating high-dimensional integrals. Our approach is simple to implement and, subject to optimizing some algorithmic parameters, it outperforms the standard Monte Carlo in both in terms of the accuracy and the time required for computing the estimator.

**Acknowledgements.** The authors are grateful to EPSRC for funding this work through the project EP/R041431/1, titled 'Randomness: a resource for real-time analytics'; YW is also funded by The Alan Turing Institute under the EPSRC grant EP/N510129/1 and by EPSRC through the project EP/S026347/1, titled 'Unparameterised multi-modal data, high order signatures, and the mathematics of data science'.

REFERENCES

[1] Beskos, A., Jasra, A., Law, K., Tempone, R., and Zhou, Y. (2017). *Multilevel sequential Monte Carlo samplers.* Stochastic Processes and their Applications, 127(5), 1417-1440.
[2] Bierig, C. and Chernov, A. (2015). *Convergence analysis of multilevel Monte Carlo variance estimators and application for random obstacle problems.* Numerische Mathematik, 130(4), 579-613.
[3] Drineas, P., Kannan R, and Mahoney, W. M. (2006). *Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication*, SIAM J. Comput. 36(1), 132-157.
[4] Eriksson-Bique, S., Solbrig, M., Stefanelli, M., Warkentin, S., Abbey, R., Ipsen, I.C.F. (2011). *Importance sampling for a Monte Carlo matrix multiplication algorithm, with application to information retrieval*, SIAM J. Comput.,33, 16891706.
[5] Giles, M. B. (2008). *Multilevel monte carlo path simulation.* Operations Research 56.3, 607-617.
[6] Giles, M. B. and Waterhouse, B. J. (2009). *Multilevel quasi-Monte Carlo path simulation.* Advanced Financial Modelling, Radon Series on Computational and Applied Mathematics 8, 165-181.
[7] Giles, M. B. (2015) *Multilevel monte carlo methods.* Acta Numerica 24, 259-328.
[8] Holodnak, J., and Ipsen, I. (2015). *Randomized approximation of the gram matrix: Exact computation and probabilistic bounds.* SIAM Journal on Matrix Analysis and Applications 36.1: 110-137.
[9] Kebaier, A. (2005). *Statistical Romberg extrapolation: a new variance reduction method and applications to option pricing.* The Annals of Applied Probability 15.4 (2005): 2681-2705.
[10] Kar, P., and Karnick, H. (2012). *Random feature maps for dot product kernels.* In Artificial Intelligence and Statistics, 583-591.
[11] Rahimi, A., and Recht, B. (2008). *Random features for large-scale kernel machines.* In Advances in neural information processing systems, 1177-1184.
[12] Teckentrup, A.L., Scheichl, R., Giles, M. B., and Ullmann E. (2013). *Further analysis of multilevel Monte Carlo methods for elliptic PDEs with random coefficients.* Numerische Mathematik, 125(3), 569-600.

[13] Wu, Y. (2018). *A Note on Random Sampling for Matrix Multiplication.* arXiv preprint, arXiv:1811.11237.
[14] Zhang, X., Wang, Q. and Chothia, Z. (2012). *openBLAS.* http://xianyi.github. io/OpenBLAS, 88.