Efficient Block Approximate Matrix Multiplication

Chuhan Yang
Division of Engineering, New York University Abu Dhabi, UAE
Tandon School of Engineering, New York University, NY, USA

Christopher Musco
Tandon School of Engineering, New York University, NY, USA

Abstract

Randomized matrix algorithms have had significant recent impact on numerical linear algebra. One especially powerful class of methods are algorithms for approximate matrix multiplication based on sampling. Such methods typically sample individual matrix rows and columns using carefully chosen importance sampling probabilities. However, due to practical considerations like memory locality and the preservation of matrix structure, it is often preferable to sample contiguous blocks of rows and columns all together. Recently, (Wu, 2018) addressed this setting by developing an approximate matrix multiplication method based on block sampling. However, the method is inefficient, as it requires knowledge of optimal importance sampling probabilities that are expensive to compute.

We address this issue by showing that the method of Wu can be accelerated through the use of a randomized implicit trace estimation method. Doing so allows us to provably reduce the cost of sampling to near-linear in the size of the matrices being multiplied, without impacting the accuracy of the final approximate matrix multiplication. Overall, this yields a fast practical algorithm, which we test on a number of synthetic and real-world data sets. We complement our algorithmic contribution with the first extensive empirical comparison of block algorithms for randomized matrix multiplication. Our method offers a significant runtime advantage over the method of (Wu, 2018) and also outperforms basic uniform sampling of blocks. However, we find another recent method of (Charalambides, 2021) which uses sub-optimal but efficiently computable sampling probabilities often (but not always) offers the best trade-off between speed and accuracy.

2012 ACM Subject Classification Theory of computation → Sketching and sampling

Keywords and phrases Approximate matrix multiplication, randomized numerical linear algebra, trace estimation

Digital Object Identifier 10.4230/LIPIcs.ESA.2023.103

Supplementary Material *Software (Source Code)*: `https://github.com/ChuhanYang/Hutch-AMM`

Funding This work was partially supported by National Science Foundation Award #2045590.

1 Introduction

Matrix computations are central across computing, with applications in optimization, scientific computing, data science, and more. In recent years, in an effort to tackle the challenge of scaling computations to larger and larger matrices, randomized methods have taken center stage [14, 27]. Collectively known as Randomized Numerical Linear Algebra or "RandNLA", the study of randomized matrix algorithms has led to faster algorithms for a number of central problems in linear algebra, including least squares regression [5, 24, 25], low-rank approximation [10, 13, 17, 23], trace estimation [18, 21], and more. Many of these algorithms are based on relatively simple sampling and sketching routines (like Johnson-Lindenstrauss random projection hashing-based methods) which are easily parallelized and adapted to modern computational environments, including distributed and streaming architectures [4].

1.1 Randomized Matrix Multiplication

Beyond the problems listed above, another topic of central interest in RandNLA is, of course, matrix multiplication. There has been significant work on developing randomized algorithms for matrix multiplication that return an approximate solution in a fraction of the time it would take to perform exact multiplication [6, 7]. Roughly, randomized matrix-multiplication methods can be split into two categories – random projection methods [25] and random sampling methods [12]. In both cases, the idea is to quickly compress given input matrices $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{B} \in \mathbb{R}^{n\times p}$ to form matrix "sketches" $\mathbf{C} \in \mathbb{R}^{m\times c}$ and $\mathbf{D} \in \mathbb{R}^{c\times p}$, where $c \ll n$. We then return $\mathbf{CD}$ as an approximation to $\mathbf{AB}$. For many standard methods, including Johnson-Lindenstrauss random projection [25] and norm-based column and row sampling [12], a choice of $c = O(1/\epsilon^2)$ for $\epsilon \in (0, 1)$ ensures that with high probability:

$$\|\mathbf{AB} - \mathbf{CD}\|_F \leq \epsilon \|\mathbf{A}\|_F \|\mathbf{B}\|_F. \tag{1}$$

The approximation $\mathbf{CD}$ can be computed in $O(cmp) = O(mp/\epsilon^2)$ time, significantly improving on the naive cost of $O(nmp)$ when n is large in comparison to $1/\epsilon^2$. So, we get an algorithmic speedup as long as $\mathbf{C}$ and $\mathbf{D}$ can be quickly obtained from $\mathbf{A}$ and $\mathbf{B}$.

1.1.1 The BasicMatrixMultiplication Method

In this paper, we are specifically interested in random sampling methods for computing $\mathbf{C}$ and $\mathbf{D}$. The most well-known method is the BasicMatrixMultiplication algorithm of Drineas, Kannan, and Mahoney [11, 12]. The idea behind the algorithm is elegant: the matrix multiplication $\mathbf{AB}$ can be cast as the sum of rank-one components, each an outer product of a column in $\mathbf{A}$ and the corresponding row in $\mathbf{B}$. In particular, $\mathbf{AB} = \sum_{i=1}^{n} \mathbf{A}^{(i)}\mathbf{B}_{(i)}$, where $\mathbf{A}^{(i)}$ denotes the i^{th} column of $\mathbf{A}$ and $\mathbf{B}_{(i)}$ denotes the i^{th} row of $\mathbf{B}$. BasicMatrixMultiplication approximates this sum by sampling the rank-one components using non-uniform sampling probabilities. To be more specific, we sample $\mathbf{A}^{(i)}\mathbf{B}_{(i)}$ with probability $p_i = \frac{\|\mathbf{A}^{(i)}\|_2\|\mathbf{B}_{(i)}\|_2}{\sum_{j=1}^{n}\|\mathbf{A}^{(j)}\|_2\|\mathbf{B}_{(j)}\|_2}$, so "heavy" columns and rows (which contribute more to the matrix product) are sampled with high probability. Equivalently, for the BasicMatrixMultiplication method, $\mathbf{C}$ and $\mathbf{D}$ consist of a subset of columns and rows from $\mathbf{A}$ and $\mathbf{B}$, sampled according to their ℓ_2 norms. More details are provided in Section 2.

The BasicMatrixMultiplication algorithm is a powerful method with a number of desirable properties. First, the method achieves the strong bound of Equation (1), matching random projection methods in the worst case. However, on top of this:

1. When constructing $\mathbf{C}$ and $\mathbf{D}$, the BasicMatrixMultiplication method preserves sparsity and structure originally present in $\mathbf{A}$ and $\mathbf{B}$. This can lead to more compact sketches in practice. The same is not true of random projection based methods, which e.g. produce dense sketches $\mathbf{C}$ and $\mathbf{D}$, even if $\mathbf{A}$ and $\mathbf{B}$ are sparse.
2. The sampling probabilities $p_1, \ldots, p_n$, and thus the sketches $\mathbf{C}$ and $\mathbf{D}$ can be computed in just $O(mn + pn)$ time, which is linear in the size of the input. We simply need to compute the norm of all columns (resp., rows) in $\mathbf{A}$ (resp., $\mathbf{B}$).
3. The sampling probabilities used are *provably optimal* in the sense that they minimize the expected squared error $\mathbb{E}\left[\|\mathbf{AB} - \mathbf{CD}\|_F^2\right]$ amongst all choices of probabilities. As a result, BasicMatrixMultiplication often outperforms the worst-case bound of Equation (1).

Thanks to the advantages above, the BasicMatrixMultiplication algorithm has been widely adapted and applied to problems in information retrieval [15], image processing [20], and distributed computation [19].

1.2 Our Contributions

Despite its many virtues, an important practical issue with the BasicMatrixMultiplication method is that it samples columns and rows from $\mathbf{A}$ and $\mathbf{B}$ *independently at random.* In many applications, it would actually be preferable to sample contiguous *blocks* of columns from $\mathbf{A}$, and respective blocks of rows from $\mathbf{B}$, i.e., to sample every column $\mathbf{A}^{(i)}, \mathbf{A}^{(i+1)}, \ldots, \mathbf{A}^{(i+q-1)}$ for some starting index i and block size q. The need for block sampling arises for a few reasons. First, in some settings it is desirable to keep nearby rows together to preserve structure in $\mathbf{A}$. As an example, [28] and [29] consider a problem where samples are obtained from block-structured finite element stiffness matrices. Moreover, even when $\mathbf{A}$ and $\mathbf{B}$ are unstructured, sampling multiple blocks of columns is typically more efficient on modern architectures where memory access costs are a major factor in determining final runtimes. When $\mathbf{A}$ and $\mathbf{B}$ are large enough that they must be stored in slow memory (e.g. on disk instead of in RAM) accessing a block of q adjacent columns (which are stored adjacent on disk) will often be far cheaper than accessing a set of q columns with non-adjacent indices.

1.2.1 The BlockBasicMatrixMultiplication Method

To address these concerns, recent work by Wu introduces a block-wise version of the BasicMatrixMultiplication method, which we call BlockBasicMatrixMultiplication [28]. The method maintains many of the nice properties of the BasicMatrixMultiplication method, and for any block size q, it can be shown that sampling $O(1/\epsilon^2)$ blocks results in the same error guarantee of Equation (1).[1] Moreover, it can be shown that the sampling probabilities used by Wu's method are *optimal for block sampling*, just as those used by BasicMatrixMultiplication were optimal for single column sampling. However, a major disadvantage of BlockBasicMatrixMultiplication is that these probabilities are no longer efficient to compute. Concretely, consider $\Theta_1, \ldots, \Theta_\ell$ which are disjoint subsets of adjacency indices that partition $\{1, \ldots, n\}$, i.e. for all j, $\Theta_i = \{k_i, k_i + 1, \ldots, k_i + q_i\}$ for some starting index k_i and block size q_i, and $\Theta_1 \cup \ldots \cup \Theta_\ell = \{1, \ldots, n\}$. If we want to sample a subset of blocks from $\Theta_1, \ldots, \Theta_\ell$, [28] shows that the optimal probability to sample the i^{th} block is equal to:

$$\hat{p}_i = \frac{\|\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}\|_F}{\sum_{j=1}^{\ell} \|\mathbf{A}^{(\Theta_j)}\mathbf{B}_{(\Theta_j)}\|_F} \tag{2}$$

This strictly generalizes the probabilities from BasicMatrixMultiplication, since for two vectors $\mathbf{A}^{(i)}$ and $\mathbf{B}_{(i)}$, $\|\mathbf{A}^{(i)}\mathbf{B}_{(i)}\|_F = \|\mathbf{A}^{(i)}\|_2 \|\mathbf{B}_{(i)}\|_2$. However, in the general block case, the probability is more expensive to compute. In particular, the naive cost of computing $\hat{p}_i$ equals $O(q_i mp)$ (the cost of multiplying $\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}$) so the total cost of sampling is $\sum_{i=1}^{\ell} O(q_i mp) = O(nmp)$. This is as slow as if we had computed $\mathbf{AB}$ exactly! Assuming for simplicity that $q_1 = \ldots = q_\ell = q$ (typically blocks have the same size), this runtime can be improved to $O(nmq + npq)$ when $q \leq p, m$ (see Equation (4) for more details). But nevertheless, the complexity is still greater than the linear run time of $O(nm + np)$ achieved by BasicMatrixMultiplication by a multiplicative factor of q. In practice, we might want q in the 100s or 1000s to take sufficient advantage of memory locality.

[1] We might have hoped to obtain a *better* guarantee with block sampling. E.g. that when sampling blocks of size q, we would only need $O(\frac{1}{q\epsilon^2})$ block samples, which is a total of $O(1/\epsilon^2)$ columns. This is not possible, however, as can be seen by considering adversarial matrices with e.g. all but columns $1, q+1, 2q+1, \ldots$ set to zero. That said, in practice, sampling c blocks from $\mathbf{A}$ and $\mathbf{B}$ often performs much better in terms of accuracy than sampling c individual columns.

1.3 Faster Probabilities via Stochastic Trace Estimation

Our main contribution is to present a simple method for efficiently implementing the optimal sampling scheme of [28]. In particular, we show how to approximate the required sampling probabilities from Equation (2) up to a multiplicative constant factor using the classic Hutchinson's stochastic traces estimation method [18]. Another central technique in randomized numerical linear algebra, Hutchinson's traces estimator can approximate the $\|\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}\|_F$ term from Equation (2) with just a small number of matrix-vector multiplies with $\mathbf{A}^{(\Theta_i)}$ and $\mathbf{B}_{(\Theta_i)}$ involving randomly chosen vectors. The end result is a method running in $O((nm+np)\log(n))$ time for computing all probabilities, which is near linear in the size of the total size of the input matrices. At the same time, an analysis of our method shows that using approximate sampling probabilities yields the same theoretical guarantees as [28], up to constant factors. We call our method Hutchinson-estimated Block Approximate Matrix Multiplication (Hutch AMM for short).

1.3.1 Empirical Evaluation

We perform an extensive experimental evaluation of our Hutch AMM method, showing that, empirically, the probability computation can be performed to sufficient accuracy using just $5(nm + np)$ floating-point operations, even for very large values of m. The cost of then computing $\mathbf{CD}$ once the probabilities are computed and used for sampling is then a lower order computational. As a result, our method outperforms an efficient implementation of the approach from [28] in terms of runtime, without sacrificing accuracy in approximating $\mathbf{AB}$.

We also compare Hutch AMM to other baselines. For example, one natural approach is simply to uniformly sample from the rows and columns of $\mathbf{A}$ and $\mathbf{B}$ (either blocks or individual rows and columns). Uniform sampling typically requires more samples in comparison to using optimal importance sampling probabilities to obtain a given level of accuracy. The trade-off, however, is that uniform sampling has no computational overhead to compute probabilities – a sample of c columns from $\mathbf{A}$ can be chosen in $O(c)$ time. The only major runtime cost is computing $\mathbf{CD}$. Nevertheless, we show that for block sampling, our method typically outperforms uniform sampling in terms of runtime to achieve a given level of approximation accuracy, showing the value of importance sampling.

Finally, we compare against a recent method of [3], which also studies block based sampling methods for approximate matrix multiplication. Their "Block CR Method" introduces an alternative approach that samples blocks with probabilities equal to:

$$\tilde{p}_i = \frac{\|\mathbf{A}^{(\Theta_i)}\|_F \|\mathbf{B}_{(\Theta_i)}\|_F}{\sum_{j=1}^{\ell} \|\mathbf{A}^{(\Theta_j)}\|_F \|\mathbf{B}_{(\Theta_j)}\|_F} \tag{3}$$

While not optimal for minimizing $\mathbb{E}\left[\|\mathbf{AB} - \mathbf{CD}\|_F^2\right]$, these probabilities minimize a natural upper bound[2] on the expected squared error, and can be shown to achieve the bound of Equation (1) with $q = O(1/\epsilon^2)$ samples [3]. So, they match the result of Wu in the worst case. At the same time, $\tilde{p}_1, \ldots, \tilde{p}_\ell$ can be computed in linear time, $O(nm + np)$, as they do not require multiplying the blocks $\mathbf{A}^{(\Theta_i)}$ and $\mathbf{B}_{(\Theta_i)}$.

[2] The text of [3] implies that the probabilities are optimal, but there is a small error in the derivation of the variance of the estimator considered, where an upper bound is mistakenly considered to be an equality. Nevertheless, the authors conclusion, that $O(1/\epsilon^2)$ block samples suffice to achieve error $\|\mathbf{AB} - \mathbf{CD}\|_F \leq \epsilon \|\mathbf{A}\|_F \|\mathbf{B}\|_F$, still holds.

Overall, we find the Block CR method of [3] extremely effective. While it does not yield as good an accuracy for a given number of samples as our method and that of Wu, the difference is relatively small. So, when comparing overall runtime to approximate the matrix product $\mathbf{AB}$, the method of [3] usually offers the best accuracy vs. runtime trade-off.

1.4 Paper Organization

The organization of this paper is as follows: In Section 2, we introduce the BlockBasicMatrixMultiplication and our proposed modification. Section 3 presents a detailed analysis and comparison of the expected squared error of the methods. In Section 4, we provide the experiment details, followed by a brief discussion of the results. Finally, we conclude the paper and discuss future research directions in Section 5.

2 Methodology

2.1 Notation

We denote matrices and vectors using bold Roman letters. For a vector $\mathbf{v} \in \mathbb{R}^n$, $\|\mathbf{v}\|_2 = \sqrt{\sum_{i=1}^n \mathbf{v}_i}$ denotes the standard Euclidean norm. For a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n \mathbf{A}_{ij}}$ denotes the Frobenius norm. We use superscript $\mathbf{A}^{(i)}$ to denote the i^{th} column of $\mathbf{A}$ and subscript $\mathbf{A}_{(j)}$ to denote the j^{th} row. For a set of c indices S, we let $\mathbf{A}^{(S)}$ denote the $m \times c$ matrix containing $\mathbf{A}^{(j)}$ for $j \in \mathcal{S}$. Similarly, $\mathbf{A}_{(S)}$ denote the $c \times n$ matrix containing $\mathbf{A}_{(j)}$ for $j \in \mathcal{S}$. For a square $\mathbf{M} \in \mathbb{R}^{n\times n}$, $\mathrm{tr}(\mathbf{M}) = \sum_{i=1}^n \mathbf{M}_{ii}$ denotes the trace. We use $\Pr[B]$ to denote the probability of a random event B and $\mathbb{E}[X]$ to denote the expectation of a random variable X.

2.2 Hutchinson-estimated block Approximate Matrix Multiplication

Given matrices $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{B} \in \mathbb{R}^{n\times p}$, BasicMatrixMultiplication returns an unbiased estimator for the matrix product $\mathbf{AB}$ by sampling rank-one components (each is an outer product of a column from $\mathbf{A}$ and corresponding row from $\mathbf{B}$). Specifically, the algorithm selects and re-weights a subset of c columns from $\mathbf{A}$ to form a matrix $\mathbf{C} \in \mathbb{R}^{m\times c}$ and the corresponding c rows from $\mathbf{B}$ to form a matrix $\mathbf{D} \in \mathbb{R}^{c\times p}$ so that $\mathbf{CD} \approx \mathbf{AB}$. To reduce the approximation error, rows and columns are sampled with non-uniform probabilities, and appropriately scaled after sampling by the inverse probability to ensure the $\mathbb{E}[\mathbf{CD}] = \mathbf{AB}$. The algorithm is summarized in Algorithm 1:

Algorithm 1 BasicMatrixMultiplication w/ Optimal Sampling Probability [12].

Input: $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{B} \in \mathbb{R}^{n\times p}$, number of samples c.
Output: Estimate of matrix product $\mathbf{AB}$.

For all $i \in 1, \ldots, n$, compute optimal sampling probability $p_i = \frac{\|\mathbf{A}^{(i)}\|_2 \|\mathbf{B}_{(i)}\|_2}{\sum_{j=1}^n \|\mathbf{A}^{(j)}\|_2 \|\mathbf{B}_{(j)}\|_2}$.
for $t = 1 \ldots, c$ **do**
  Pick $j \in \{1, \ldots, n\}$ with probability $\Pr(j = k) = p_k$.
  Set $\mathbf{C}^{(t)} = \frac{\mathbf{A}^{(j)}}{\sqrt{cp_j}}$ and $\mathbf{D}_{(t)} = \frac{\mathbf{B}_{(j)}}{\sqrt{cp_j}}$
end
Return $\mathbf{CD}$

As discussed in Section 1, the sampling probabilities $p_1, \ldots, p_n$ used in Algorithm 1 are optimal in that they minimize $\mathbb{E}\left[\|\mathbf{AB} - \mathbf{CD}\|_F^2\right]$ amongst all possible choices of probabilities [12] (when using an unbiased estimator of the same form as Algorithm 1).

It is possible to extend the BasicMatrixMultiplication method to sampling blocks of rows and columns. To do so, we consider a partition of the indices $\{1, \ldots, n\}$ into ℓ disjoint sets $\Theta_1, \ldots, \Theta_\ell$. [28] derived optimal probabilities $\hat{p}_1, \ldots, \hat{p}_\ell$ for sampling in the block setting, which are shown below in Algorithm 2. Again, these probabilities minimize $\mathbb{E}\left[\|\mathbf{AB} - \mathbf{CD}\|_F^2\right]$ amongst all possible choices of sampling probabilities.

Algorithm 2 Block BasicMatrixMultiplication w/ Optimal Sampling Probability [28].

Input: $\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{B} \in \mathbb{R}^{n \times p}$, number of samples c, disjoint set of column indices $\{\Theta_1, \ldots, \Theta_\ell\}$ with $\Theta_1 \cup \ldots \cup \Theta_\ell = \{1, \ldots, n\}$.
Output: Estimate of matrix product $\mathbf{AB}$.

For all $i \in 1, \ldots, \ell$, compute optimal sampling probability $\hat{p}_i = \frac{\|\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}\|_F}{\sum_{j=1}^{\ell} \|\mathbf{A}^{(\Theta_j)}.\mathbf{B}_{(\Theta_j)}\|_F}$
Initialize $\mathbf{C}$ and $\mathbf{D}$ as empty matrices.
for $t = 1, \ldots, c$ **do**
  Pick $j \in \{1, \ldots, \ell\}$ with probability $\Pr(j = k) = \hat{p}_k$.
  Append $\frac{\mathbf{A}^{(\Theta_j)}}{\sqrt{c\hat{p}_j}}$ onto $\mathbf{C}$'s columns and $\frac{\mathbf{B}_{(\Theta_j)}}{\sqrt{c\hat{p}_j}}$ onto $\mathbf{D}$'s rows.
end
Return $\mathbf{CD}$

▷ Claim 1. Assume $\Theta_1, \ldots, \Theta_\ell$ are equally sized, so each contains $q = \frac{n}{l}$ indices. Then the BlockBasicMatrixMultiplication method of Algorithm 2 can be implemented in time $O(mnp)$, or in time $O(n(m+p)q + cqmp)$, which is faster when $q \leq m, p$.

Proof. Once sampling probabilities are computed and sampling performed, the cost of multiplying $\mathbf{CD}$ is equal to $O(cqmp)$. Specifically, $\mathbf{C}$ contains c blocks of q columns, so has dimension $m \times cq$ and $\mathbf{D}$ has dimension $cq \times p$. We focus on the cost of computing the optimal probabilities, $\hat{p}_1, \ldots, \hat{p}_\ell$. Doing so requires computing $\|\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}\|_F$ all i. Since $\mathbf{A}^{(\Theta_i)} \in \mathbb{R}^{m \times q}$ and $\mathbf{B}_{(\Theta_i)} \in \mathbb{R}^{q \times p}$, this can be done directly in $O(qmp)$ time. Summing over all ℓ blocks and using that $q = \frac{n}{\ell}$, the total runtime is $O(nmp)$ to compute $p_1, \ldots, p_\ell$. Alternatively, if $q \leq m + p$, we can use the cyclic property of the trace:

$$\|\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}\|_F^2 = \mathrm{tr}(\mathbf{B}_{(\Theta_i)}^T(\mathbf{A}^{(\Theta_i)})^T\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}) = \mathrm{tr}((\mathbf{A}^{(\Theta_i)})^T\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}\mathbf{B}_{(\Theta_i)}^T). \quad (4)$$

If we first compute $(\mathbf{A}^{(\Theta_i)})^T\mathbf{A}^{(\Theta_i)}$ and $\mathbf{B}_{(\Theta_i)}\mathbf{B}_{(\Theta_i)}^T$ and then multiply the resulting $q \times q$ matrices, the above trace can be computed in time $O(q^2m + q^2p + q^3) = O((m+p)q^2)$. Again, summing over all ℓ blocks and using that $q = \frac{n}{\ell}$, the total runtime is $O(n(m+p)q)$. ◁

As illustrated by Claim 1, the computational cost of BlockBasicMatrixMultiplication can be prohibitively expensive. When q is larger than either m or p, the cost is as large as the cost of computing the product $\mathbf{AB}$ exactly.

Our proposed approach is to speed up the BlockBasicMatrixMultiplication method by using Hutchinson's stochastic trace estimator [16, 18] to efficiently approximate $\|\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}\|_F^2 = \mathrm{tr}(\mathbf{B}_{(\Theta_i)}^T(\mathbf{A}^{(\Theta_i)})^T\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)})$. Given an $p \times p$ matrix $\mathbf{M}$, this estimator estimates $\mathrm{tr}(\mathbf{M})$ via repeated multiplication of the matrix by random vectors. In particular, for a sampling parameter h and vectors $\mathbf{g}_1, \ldots, \mathbf{g}_h$ each chosen to have independent random entries with mean 0 and variance 1, the estimator takes the form:

$$H_h(\mathbf{M}) = \frac{1}{h}\sum_{j=1}^{h} g_j^\intercal \mathbf{M} g_j. \tag{5}$$

Typically, random $\{-1, +1\}$ Rademacher random variables are chosen for the entries of each $\mathbf{g}_j$. It is easily checked that $\mathbb{E}\left[H_h(\mathbf{M})\right] = \mathrm{tr}(\mathbf{M})$ and for sufficiently large h, the estimator concentrates around its expectation [1, 8]. The value of the estimation in Equation (5) is that, for $\mathbf{M} = \mathbf{B}_{(\Theta_i)}^T(\mathbf{A}^{(\Theta_i)})^T\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}$, each term in the sum $\sum_{i=1}^{h} g_j^\intercal \mathbf{M} g_j$ can be computed in just $O(pq + mq + pq + mq) = O(q(m+p))$ time by multiplying $\mathbf{B}_{(\Theta_i)}^T(\mathbf{A}^{(\Theta_i)})^T\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}\mathbf{g}_j$ from right to left. When h is a small constant, the cost is thus linear in the total size of the blocks $\mathbf{A}^{(\Theta_i)}$ and $\mathbf{B}_{(\Theta_i)}$. Overall, our proposed algorithm is summarized in Algorithm 3:

Algorithm 3 Hutchinson-estimated Block Approx. Matrix Multiplication (Hutch AMM).

Input: $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{B} \in \mathbb{R}^{n\times p}$, number of samples c, disjoint sets of column indices $\{\Theta_1, \ldots, \Theta_\ell\}$ with $\Theta_1 \cup \ldots \cup \Theta_\ell = \{1, \ldots, n\}$, number of Hutchinson's samples h.

Output: Estimate of matrix product $\mathbf{AB}$.

For all $i \in 1, \ldots, \ell$, compute approximate optimal sampling probability

$$\bar{p}_i = \frac{\sqrt{H_h(\mathbf{B}_{(\Theta_i)}^T(\mathbf{A}^{(\Theta_i)})^T\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)})}}{\sum_{j=1}^{\ell}\sqrt{H_h(\mathbf{B}_{(\Theta_j)}^T(\mathbf{A}^{(\Theta_j)})^T\mathbf{A}^{(\Theta_j)}\mathbf{B}_{(\Theta_j)})}}$$

Initialize $\mathbf{C}$ and $\mathbf{D}$ as empty matrices.
for $t = 1, \ldots, c$ **do**
  Pick $j \in \{1, \ldots, \ell\}$ with probability $\Pr(j = k) = \bar{p}_k$.
  Append $\frac{\mathbf{A}^{(\Theta_j)}}{\sqrt{c\bar{p}_j}}$ onto $\mathbf{C}$'s columns and $\frac{\mathbf{B}_{(\Theta_j)}}{\sqrt{c\bar{p}_j}}$ onto $\mathbf{D}$'s rows.
end
Return $\mathbf{CD}$

We can bound the runtime of Algorithm 3 as follows:

▷ Claim 2. Assume $\Theta_1, \ldots, \Theta_\ell$ are equally sized, so each contains $q = \frac{n}{l}$ indices. Then the Hutch AMM method of Algorithm 3 can be implemented in time $O(nh(m+p) + cqmp)$.

Proof. The proof follows from the discussion above. For each block, computing the Hutchinson's estimate $H_h(\mathbf{B}_{(\Theta_i)}^T(\mathbf{A}^{(\Theta_i)})^T\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)})$ takes time $O(qh(m+p))$ for a total runtime of $\ell \cdot O(qh(m+p)) = O(nh(m+p))$. We then have an additional cost of $O(cqmp)$ to compute $\mathbf{CD}$, which will typically be a lower order term when the number of samples c is small. ◁

From Claim 1 and Claim 2, we can check that Hutch AMM improves on the BlockBasicMatrixMultiplication algorithm whenever the number of Hutchinson's iterations h is less than the block size q. As we will prove in the next section, if suffices to set $h = O(\log n)$ to match the accuracy guarantees of BlockBasicMatrixMultiplication up to a multiplicative constant.

3 Error Analysis

In this section we provide an analysis of our Hutch AMM method. We start by stating the main result from [28], which bounds the expected squared error of the BlockBasicMatrixMultiplication method.

▶ **Proposition 3** (Proposition 2.2 [28]). *For input matrices* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *and* $\mathbf{B} \in \mathbb{R}^{n\times p}$*, let* $\mathbf{CD}$ *be the approximate matrix product returned by Algorithm 2. We have that* $\mathbb{E}[\mathbf{CD}] = \mathbf{AB}$ *and the expected squared approximation error in Frobenius norm is:*

$$\mathbb{E}[\|\mathbf{AB} - \mathbf{CD}\|_F^2] = \frac{1}{c}\sum_{i=1}^{\ell}\frac{1}{\hat{p}_i}\|\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}\|_F^2 - \frac{\|\mathbf{AB}\|_F^2}{c}. \tag{6}$$

Setting $\hat{p}_i := \frac{\|\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}\|_F}{\sum_{i=1}^{l}\|\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}\|_F}$ *for* $i = 1, \ldots, \ell$*, we obtain minimum expected error, which can be upper bounded by:*

$$\mathbb{E}[\|\mathbf{AB} - \mathbf{CD}\|_F^2] \leq \frac{1}{c}\left(\sum_{i=1}^{\ell}\|\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}\|_F\right)^2 \tag{7}$$

Note that, applying submultiplicativity of the Frobenius norm and Cauchy-Schwarz inequality we can (loosely) bound:

$$\begin{aligned}\frac{1}{c}\left(\sum_{i=1}^{\ell}\|\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}\|_F\right)^2 &\leq \frac{1}{c}\left(\sum_{i=1}^{\ell}\|\mathbf{A}^{(\Theta_i)}\|_F\|\mathbf{B}_{(\Theta_i)}\|_F\right)^2 \\ &\leq \frac{1}{c}\left(\sum_{i=1}^{\ell}\|\mathbf{A}^{(\Theta_i)}\|_F^2\right)\left(\sum_{i=1}^{\ell}\|\mathbf{B}^{(\Theta_i)}\|_F^2\right) = \frac{1}{c}\|\mathbf{A}\|_F^2\|\mathbf{B}\|_F^2\end{aligned}$$

Using the resulting bound on $\mathbb{E}[\|\mathbf{AB} - \mathbf{CD}\|_F^2]$ we can apply Markov's inequality to the non-negative random variable $\|\mathbf{AB} - \mathbf{CD}\|_F^2$ to prove that, with probability $(1 - \delta)$, BlockBasicMatrixMultiplication achieves error $\|\mathbf{AB} - \mathbf{CD}\|_F \leq \frac{1}{\sqrt{\delta c}}\|\mathbf{A}\|_F\|\mathbf{B}\|_F$. In other words, with $c = O(1/\delta\epsilon^2)$ samples, we obtain error $\epsilon\|\mathbf{A}\|_F\|\mathbf{B}\|_F$ with probability $(1 - \delta)$, as desired.

What happens if we sample with approximately optimal probabilities? From Equation (6), it can be checked directly that, if instead $\mathbf{C}$ and $\mathbf{D}$ are sampled with probabilities $\bar{p}_1, \ldots, \bar{p}_\ell$ satisfying $\bar{p}_i \geq \frac{1}{\beta}\hat{p}_i$ for some constant $\beta \geq 1$, then the expected squared error can be bounded by:

$$\mathbb{E}[\|\mathbf{AB} - \mathbf{CD}\|_F^2] \leq \frac{\beta}{c}\left(\sum_{i=1}^{\ell}\|\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}\|_F\right)^2. \tag{8}$$

In other words, as long as $\bar{p}_1, \ldots, \bar{p}_\ell$ approximate the optimal sampling probabilities $\hat{p}_1, \ldots, \hat{p}_\ell$ up to a constant factor, we obtain expected squared error that is within a multiplicative constant of the bound from Equation (8) achieved by Wu's optimal method.

With this in mind, we focus on showing that the approximate probabilities used in our Hutch AMM method satisfy this multiplicative bound. In particular, we prove:

▶ **Lemma 4.** *For inputs* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *and* $\mathbf{B} \in \mathbb{R}^{n\times p}$*, let* $\bar{p}_1, \ldots, \bar{p}_\ell$ *be computed as in Algorithm 3 with parameter* h*, and let* $\hat{p}_1, \ldots, \hat{p}_\ell$ *be the optimal sampling probabilities from Algorithm 2. As long as* $h \geq C\log(\ell/\delta)$ *for a fixed constant* C*, then with probability* $1 - \delta$*,*

$$\bar{p}_i \geq \frac{1}{2}\hat{p}_i \qquad \text{for all } i \in 1, \ldots \ell.$$

In other words, as long as $h = O(\log \ell) \leq O(\log n)$, then with high probability we obtain sampling probabilities with a constant factor of the optimal probabilities, and thus expected squared error with a constant of Equation (7), as desired.

To prove 4, we use the following standard bound on the accuracy of Hutchinson's estimator:

▶ **Lemma 5** (See [8] or [21]). *Let $\mathbf{M} \in \mathbb{R}^{d\times d}$ and $\delta \in (0, 1/2]$. Let $H_h(\mathbf{M})$ be Hutchinson's estimator run for h repetition, as in Equation (5). For fixed constants C_1, C_2, if $h > C_1 \log(1/\delta)$, then with probability $\geq 1-\delta$*

$$|H_h(\mathbf{M}) - \mathrm{tr}(\mathbf{M})| \leq C_2\sqrt{\frac{\log(1/\delta)}{h}}\|\mathbf{M}\|_F. \tag{9}$$

Proof of Lemma 4. Let $\mathbf{X}_i$ denote $\mathbf{X}_i = \mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}$. Let $h = 4\max(C_1, C_2^2)\log(\ell/\delta)$. Then by Lemma 5 it follows that, for any $i \in 1, \ldots, \ell$,

$$\left|H_h(\mathbf{X}_i^T\mathbf{X}_i) - \mathrm{tr}(\mathbf{X}_i^T\mathbf{X}_i)\right| \leq \frac{1}{2}\|\mathbf{X}_i^T\mathbf{X}_i\|_F,$$

with probability $1 - \delta/\ell$. By a union bound, the statement holds simultaneously for all $i \in 1, \ldots, \ell$ with probability $1-\delta$. Since $\mathbf{X}_i^T\mathbf{X}_i$ is a positive semidefinite matrix, we have that $\|\mathbf{X}_i^T\mathbf{X}_i\|_F \leq \mathrm{tr}(\mathbf{X}_i^T\mathbf{X}_i)$. So we can conclude that for all $i \in 1, \ldots, \ell$,

$$.5 \cdot \mathrm{tr}(\mathbf{X}_i^T\mathbf{X}_i) \leq H_h(\mathbf{X}_i^T\mathbf{X}_i) \leq 1.5 \cdot \mathrm{tr}(\mathbf{X}_i^T\mathbf{X}_i).$$

We have that $\bar{p}_i = \frac{\sqrt{H_h(\mathbf{X}_i^T\mathbf{X}_i)}}{\sum_{j=1}^{\ell}\sqrt{H_h(\mathbf{X}_j^T\mathbf{X}_j)}}$, so we conclude that, as desired,

$$\bar{p}_i \geq \frac{\sqrt{.5}\cdot\mathrm{tr}(\mathbf{X}_i^T\mathbf{X}_i)}{\sqrt{1.5}\sum_{j=1}^{\ell}\mathrm{tr}(\mathbf{X}_j^T\mathbf{X}_j)} \geq \frac{1}{2}\hat{p}_i. \qquad \blacktriangleleft$$

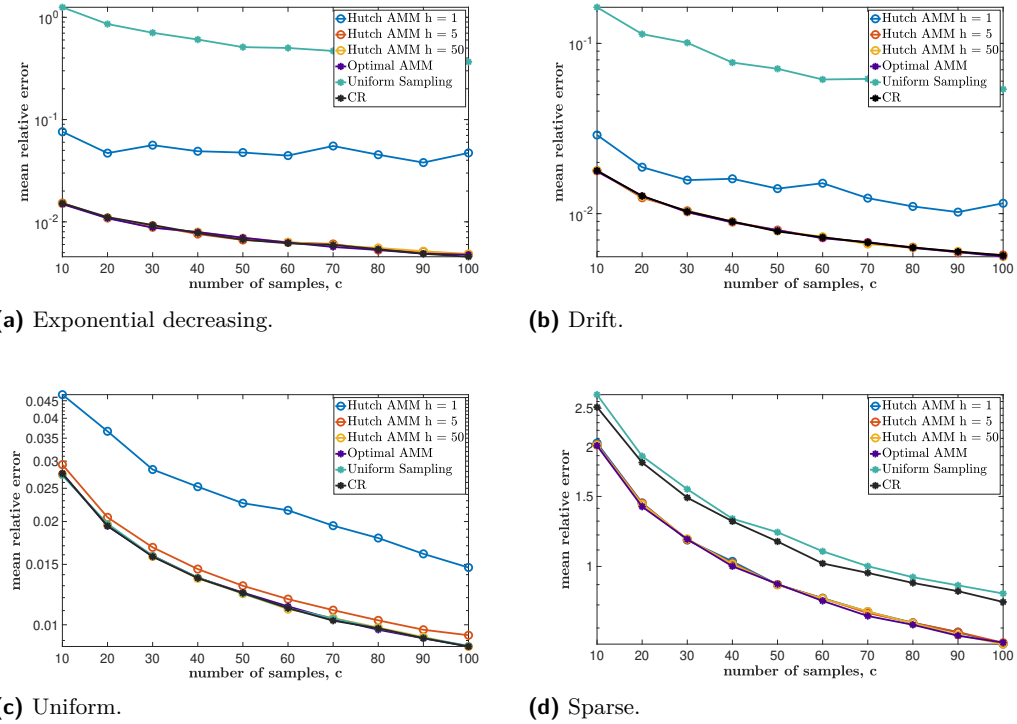
Combined with the bound in Equation (8) and Markov's inequality, Lemma 4 implies that as long as $h = O(\log(\ell/\delta)$, then with probability $1-\delta$, Hutch AMM returns an estimate $\mathbf{CD}$ satisfying $\|\mathbf{AB} - \mathbf{CD}\|_F \leq \frac{1}{\sqrt{\delta c}}\sum_{i=1}^{\ell}\|\mathbf{A}^{(\Theta_i)}\mathbf{B}_{(\Theta_i)}\|_F \leq \frac{1}{\sqrt{\delta c}}\|\mathbf{A}\|_F\|\mathbf{B}\|_F$. Accordingly, we match the guarantees of the BlockBasicMatrixMultiplication method, but with a total computational cost of just $O(n\log(n)(m+p))$ to compute sampling probabilities, which is near linear in the size of the inputs $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{B} \in \mathbb{R}^{n\times p}$.

4 Experiments

In this section, we test our algorithm on both synthetic matrices and real-world datasets. We demonstrate its performance by comparing it with the optimal BlockBasicMatrixMultiplication method [28], which in this section we call "Optimal Block AMM" for conciseness. As discussed in the introduction, we also compare with two baseline method: sampling blocks using uniform probabilities $\tilde{p}_i = \frac{1}{\ell}$ and the CR method from [3], which samples with the probabilities shown in Equation (3).[3]

To compare methods, we use mean relative error $\frac{\|\mathbf{CD}-\mathbf{AB}\|_F}{\|\mathbf{AB}\|_F}$ and the total computational time as two performance metrics in estimation accuracy and efficiency. All experiments were conducted in Matlab R2022b on a laptop with a 2.7 GHz Quad-Core Intel Core i7 and 16 GB memory.

[3] We note that there has been some recent follow-up work in [22] on modifications of the BlockBasicMatrixMultiplication method. However, the algorithms in that work ultimately perform individual row/column sampling within each block, so they are not directly comparable to the other methods studied in this paper, which always sampled contiguous blocks of rows or columns together.



(a) Exponential decreasing.

(b) Drift.

(c) Uniform.

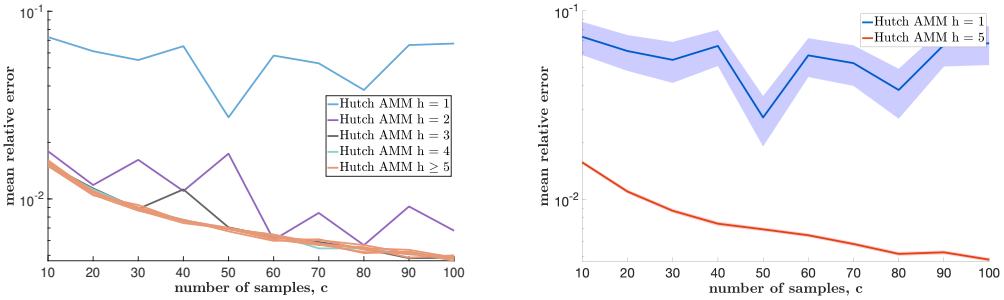
(d) Sparse.

Figure 1 Performance comparison of different block approximate matrix multiplication methods on matrices generated by different procedures. The matrices for plots (a-c) have size of 100×10000, with a sampling block size $q = 100$, (d) involves matrices of size of 10×100000, with sampling block size $q = 1000$. Our results show that, when comparing the number of samples c to average relative error for 200 trials, the Hutch AMM method with $h \geq 5$ consistently performs as well as the optimal Optimal Block AMM method for matrices with significant differences in row/column norms (a,b), while the Hutch AMM with $h = 1$ and uniform sampling methods perform worse. The CR methods perform similarly to the optimal block AMM, except for the sparse case (d).

4.1 Data sets

For our synthetic data, we generate $\mathbf{B} \in \mathbb{R}^{10000\times 100}$ with all entries uniform random in $[0, 1]$. For two data sets, we generate $\mathbf{A} \in R^{100\times 10000}$ with random Gaussian entries with variance 1, and mean depending on the column index, i. For (a) Exponential decreasing, the means are uniformly spaced on an exponential grid from $\exp(50)$ down to 1. For (b) Drift, the means follow two trends. For the first 5000 columns they are exponentially decreasing as in (a), and for the next 5000 columns the means are linearly decreasing. A similar matrix was used in [9]. These first two data sets have columns and blocks of columns with widely varying magnitudes. For our third dataset, (c) Uniform, $\mathbf{A}$ is simply generated with entries uniform random in $[0, 1]$ like $\mathbf{B}$. Finally, for (d) Sparse, we construct a sparse matrix $\mathbf{A} \in \mathbb{R}^{10\times 100000}$ with 0.1% non-zero entries selecting uniformly at random in $[0, 1]$. We construct $\mathbf{B} \in \mathbb{R}^{100000\times 10}$ with entries drawn from a standard normal distribution, but with 1% entries randomly replaced with larger values uniformly distributed between 0 and an arbitrary positive number (we used $\exp(4)$ in this paper).

For real-world data, we consider two different application scenarios: (1) Natural language processing: We extracted TF-IDF matrices from a subset of the TDT2 corpus (Nist Topic Detection and Tracking corpus)[2]. (2) Time series: We formed the whole-month (February



(a) Mean relative error comparison of Hutch AMM from h=1 to h=10.

(b) Estimation error for selected h plotted in shaded region, shaded region ranges between plus and minus one standard deviation from mean.

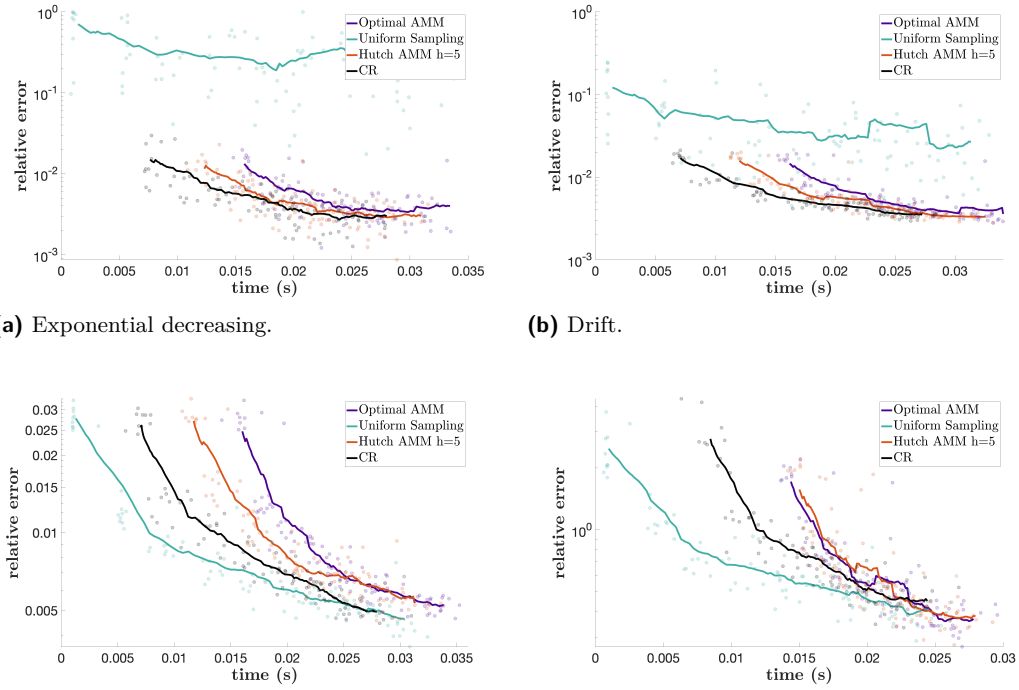
Figure 2 This figure investigates the impact of h on the estimation accuracy of Hutch AMM methods. The graph shows the average relative estimation error against sampling number c for 200 trials for the Exponential Decreasing synthetic data. Higher choices of h generally lead to better average error, but there is little to no benefit of choosing a value of h larger than 5. Similarly, low choices of h lead to higher variance, as shown in (b). The shaded region represents the plus and minus one standard deviation from the mean relative estimation error among the 200 trials. As we can see, this region is much narrower for $h = 5$ in comparison to $h = 1$.

2022) yellow taxi trip records as a matrix using NYC Taxi and Limousine Commission (TLC) Trip Record Data[26], the columns are sorted by pick-up time. We include numerical features such as trip distances, itemized fares, and driver-reported passenger counts, as well as one-hot-encoded categorical features such as VendorID and payment type. For the TF-IDF experiments, we construct a matrix $\mathbf{A}$ of size 1000×36763; for the trip record experiments, $\mathbf{A}$ has size 24×2877693. In both cases we set $\mathbf{B} = \mathbf{A}^T$ and use block size $q = 100$.

4.2 Results

We first compare the accuracy of block approximate matrix multiplication methods as a function of the number of samples c. As shown in Figure 1, our Hutch AMM method run with h as small as 5 consistently matches the Optimal Block AMM method from [28]. Both methods outperform uniform sampling, except when $\mathbf{A}$ and $\mathbf{B}$ have uniform column norms, in which case there is limited value in importance sampling. The CR method also performs quite well, although is worse than our Hutch AMM method for the sparse synthetic data. Running Hutch AMM with $h = 1$ shows worse performance. In Figure 2, we take a closer look at the impact of varying h, and the choice of $h = 5$ seems to be a sweet spot – lower values lead to higher error (due to worse approximation of the optimal sampling probabilities) whereas high values offer little improvement, and lead to higher computation cost.

Having settled on $h = 5$ as a default parameter for our Hutch AMM method, we proceed to compare the runtime against all baselines. We do so in Figure 3 for the synthetic data and Figure 4 for the real-world data sets. In both cases, we perform repeated trials with various choices of c. Since runtime and accuracy are not deterministic functions of c, this leads to a scatter plot comparing running time vs. accuracy, which we visualize plot plotting a moving average trend for each method tested. Overall, the plots show that Hutch AMM offers improved computation time over Optimal Block AMM, and typically much better accuracy than uniform sampling. However, while it showed better performance than the CR method in terms of accuracy for a given number of samples for some problems (e.g. the Sparse data), the slightly lower computational complexity of the CR method comes through



(a) Exponential decreasing.

(b) Drift.

(c) Uniform.

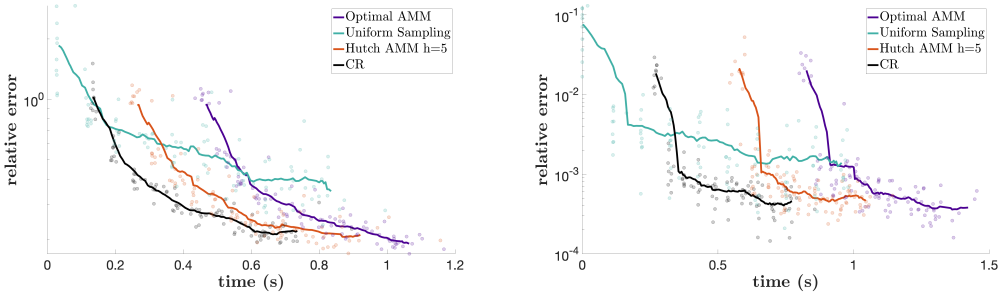
(d) Sparse.

Figure 3 Relative estimation error against computation time, with a moving average line generated using window length 20. Matrices are generated using the same procedures as in Figure 1. For interpretation: A line to the left indicates the corresponding method's superior computational efficiency; a line towards the bottom indicates the corresponding method's superior estimation accuracy. Our observations indicate that both our Hutch AMM method and the CR method improve on the Optimal Block Matrix multiplication method in terms of efficiency/accuracy tradeoff. They also beat uniform sampling in all cases except for (c). Overall, the CR method shows the best overall performance, despite its use of non-uniform sampling probabilities.

in the plots. In terms of overall computational complexity, in our experiments, it was more efficient to run the CR method. The lack of optimal sampling probabilities is made up for by including a slightly larger number of samples c in $\mathbf{C}$ in $\mathbf{D}$.

5 Summary and Conclusion

In this study, we present the Hutchinson-estimated block Approximate Matrix Multiplication (Hutch AMM) method, which is a more computationally efficient variant of the existing BlockBasicMatrixMultiplication method. Our proposed method utilizes Hutchinson's estimator to calculate a near-optimal sampling probability, which reduces computation cost while maintaining a high level of estimation accuracy when an appropriate h parameter is chosen. We validate our method through a detailed complexity analysis and proof of bounded squared error. Additionally, we perform experiments on both synthetic and practical data. Our results demonstrate that the Hutch AMM method is more computationally efficient and provides accurate estimation on par with the optimal BlockBasicMatrixMultiplication in all cases. Furthermore, we show that our proposed method can offer better performance in terms of estimation accuracy than the similarly efficient CR block matrix multiplication method.



(a) TDT2 corpus. **(b)** NYC Yellow Taxi Trip.

Figure 4 The figure shows scatter points of relative estimation error plotted against computation time, with a moving average line generated using smoothing window length 20. The matrices used are practical datasets: (a) TF-IDF matrices from a subset of the TDT2 corpus, and (b) the February 2022 Trip Record Data from the NYC Taxi and Limousine Commission (TLC). Despite the different characteristics of these datasets, we still observe that Hutch AMM is more computationally efficient and achieves close estimation accuracy to optimal block AMM.

We acknowledge that our proposed Hutch AMM method has potential for improvement. Although our current partitioning strategy splits matrix columns into blocks by a natural sequence, it may be possible to optimize this process by implementing alternative partitioning strategies tailored for specific matrix factors. Moreover, we can explore alternative estimators for near-optimal sampling probabilities or even consider biased estimators to further enhance the computational efficiency of AMM. These are promising avenues for future research in this area.

References

1 Haim Avron and Sivan Toledo. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *J. ACM*, 58(2), 2011.

2 Deng Cai, Xuanhui Wang, and Xiaofei He. Probabilistic dyadic data analysis with local and global consistency. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML'09)*, pages 105–112, 2009.

3 Neophytos Charalambides, Mert Pilanci, and Alfred O Hero. Approximate weighted CR coded matrix multiplication. In *Proceedings of the 2021 International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 5095–5099. IEEE, 2021.

4 Kenneth Clarkson and David Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 205–214, 2009.

5 Kenneth L. Clarkson and David P. Woodruff. Low rank approximation and regression in input sparsity time. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 81–90, 2013.

6 Edith Cohen and David D Lewis. Approximating matrix multiplication for pattern recognition tasks. *Journal of Algorithms*, 30(2):211–252, 1999.

7 Michael B. Cohen, Jelani Nelson, and David P. Woodruff. Optimal approximate matrix product in terms of stable rank. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP)*, 2016.

8 Alice Cortinovis and Daniel Kressner. On randomized trace estimates for indefinite matrices with an application to determinants. *Foundations of Computational Mathematics*, 22(3):875–903, 2022.

9 Amey Desai, Mina Ghashami, and Jeff M Phillips. Improved practical matrix sketching with guarantees. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1678–1690, 2016.

10 Amit Deshpande and Santosh Vempala. Adaptive sampling and fast low-rank matrix approximation. In *Proceedings of the 10th International Workshop on Randomization and Computation (RANDOM)*, pages 292–303, 2006.

11 Petros Drineas and Ravi Kannan. Fast Monte-Carlo algorithms for approximate matrix multiplication. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 452–459. IEEE, 2001.

12 Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1):132–157, 2006.

13 Petros Drineas, Ravi Kannan, and Michael W. Mahoney. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. *SIAM Journal on Computing*, 36(1):158–183, 2006.

14 Petros Drineas and Michael W. Mahoney. RandNLA: Randomized numerical linear algebra. *Commun. ACM*, 59(6), 2016.

15 Sylvester Eriksson-Bique, Mary Solbrig, Michael Stefanelli, Sarah Warkentin, Ralph Abbey, and Ilse CF Ipsen. Importance sampling for a Monte Carlo matrix multiplication algorithm, with application to information retrieval. *SIAM Journal on Scientific Computing*, 33(4):1689–1706, 2011.

16 Didier Girard. Un algorithme simple et rapide pour la validation croisee géenéralisée sur des problémes de grande taille. Technical report, École nationale supérieure d'informatique et de mathématiques appliquées de Grenoble, 1987.

17 Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

18 Michael F. Hutchinson. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

19 Tayyebeh Jahani-Nezhad and Mohammad Ali Maddah-Ali. CodedSketch: A coding scheme for distributed computation of approximated matrix multiplication. *IEEE Transactions on Information Theory*, 67(6):4185–4196, 2021.

20 Humberto Madrid, Valia Guerra, and Marielba Rojas. Sampling techniques for Monte Carlo matrix multiplication with applications to image processing. In *Mexican Conference on Pattern Recognition*, pages 45–54. Springer, 2012.

21 Raphael A Meyer, Cameron Musco, Christopher Musco, and David P Woodruff. Hutch++: Optimal stochastic trace estimation. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 142–155. SIAM, 2021.

22 Chengmei Niu and Hanyu Li. Optimal sampling algorithms for block matrix multiplication. *Journal of Computational and Applied Mathematics*, page 115063, 2023.

23 Christos H Papadimitriou, Prabhakar Raghavan, Hisao Tamaki, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. *Journal of Computer and System Sciences*, 61(2):217–235, 2000.

24 Vladimir Rokhlin and Mark Tygert. A fast randomized algorithm for overdetermined linear least-squares regression. *Proceedings of the National Academy of Sciences*, 105(36):13212–13217, 2008.

25 Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 143–152, 2006.

26 NYC Taxi and Limousine Commission (TLC). Tlc trip record data. `https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page`, 2022.

27 David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1–2):1–157, 2014.

28 Yue Wu. A note on random sampling for matrix multiplication. *arXiv preprint*, 2018. arXiv:1811.11237.

29 Yue Wu, Dimitris Kamilis, and Nick Polydorides. A randomised finite element method for elliptic partial differential equations. *arXiv preprint*, 2019. arXiv:1903.07696.