

# Efficient and Near-Optimal Noise Generation for Streaming Differential Privacy

Krishnamurthy (Dj) Dvijotham<sup>\*</sup>   H. Brendan McMahan<sup>†</sup>   Krishna Pillutla<sup>‡</sup>  
Thomas Steinke<sup>§</sup>   Abhradeep Thakurta<sup>¶</sup>

April 30, 2024

## Abstract

In the task of differentially private (DP) continual counting, we receive a stream of increments and our goal is to output an approximate running total of these increments, without revealing too much about any specific increment. Despite its simplicity, differentially private continual counting has attracted significant attention both in theory and in practice. Existing algorithms for differentially private continual counting are either inefficient in terms of their space usage or add an excessive amount of noise, inducing suboptimal utility.

The most practical DP continual counting algorithms add carefully correlated Gaussian noise to the values. The task of choosing the covariance for this noise can be expressed in terms of factoring the lower-triangular matrix of ones (which computes prefix sums). We present two approaches from this class (for different parameter regimes) that achieve near-optimal utility for DP continual counting and only require logarithmic or polylogarithmic space (and time).

Our first approach is based on a space-efficient streaming matrix multiplication algorithm for a class of Toeplitz matrices. We show that to instantiate this algorithm for DP continual counting, it is sufficient to find a low-degree rational function that approximates the square root on a circle in the complex plane. We then apply and extend tools from approximation theory to achieve this. We also derive efficient closed-forms for the objective function for arbitrarily many steps, and show direct numerical optimization yields a highly practical solution to the problem. Our second approach combines our first approach with a recursive construction similar to the binary tree mechanism.

---

<sup>0</sup>Alphabetical author order.

<sup>\*</sup>Google DeepMind ..... dvij@google.com

<sup>†</sup>Google Research ..... mcmahan@google.com

<sup>‡</sup>IIT Madras. Work done at Google. .... pillutla@cs.washington.edu

<sup>§</sup>Google DeepMind ..... steinke@google.com

<sup>¶</sup>Google DeepMind ..... athakurta@google.com

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Our Contributions . . . . .	6
1.2	Our Algorithms & Techniques . . . . .	8
1.3	An Empirical Comparison of Mechanisms . . . . .	12
<b>2</b>	<b>Background</b>	<b>15</b>
2.1	Prior Work on Matrix Factorizations for Continual Counting . . . . .	16
2.2	Lower Triangular Toeplitz Factorizations versus General Factorizations . . . . .	17
2.3	Applications of Matrix Factorizations in Machine Learning . . . . .	24
2.4	Other Related Work . . . . .	25
<b>3</b>	<b>Efficiently-Sampleable Factorization via Rational Generating Functions</b>	<b>27</b>
3.1	Sequences, Lower Triangular Toeplitz Matrices, & Generating Functions . . . . .	28
3.2	Rational Generating Functions and Constant-Recurrent Sequences . . . . .	30
3.3	Efficient Sampling via BLT Multiplication . . . . .	34
<b>4</b>	<b>Factorizations via Rational Function Approximation</b>	<b>35</b>
4.1	Reduction to Approximating the Square Root . . . . .	36
4.2	Rational Approximation of $\sqrt{x}$ . . . . .	40
4.3	Putting Everything Together . . . . .	46
<b>5</b>	<b>Factorizations via Direct Optimization</b>	<b>48</b>
5.1	Fast and Exact Error Computation . . . . .	48
5.2	Direct Optimization of BLTs . . . . .	53
5.3	Derivation of the Inverse BLT Parameterization . . . . .	54
<b>6</b>	<b>Generalizing the Binary Tree Approach</b>	<b>56</b>
6.1	Recursive Matrix Factorization . . . . .	57
6.2	Recursive Algorithm . . . . .	60
6.3	Combining Recursion with BLTs . . . . .	63
<b>7</b>	<b>Numerical Lower Bound on Optimal Performance</b>	<b>65</b>
<b>8</b>	<b>BLT with <math>d = 1</math> Buffers</b>	<b>70</b>
<b>9</b>	<b>Conjecture</b>	<b>71</b>
	<b>Acknowledgements</b>	<b>72</b>
	<b>References</b>	<b>72</b>

# 1 Introduction

The simplest task in differentially private data analysis is *counting*. This task has an equally simple (and optimal) algorithm: we simply add Laplace or Gaussian noise to the final count [DMNS06].

*Continual counting* [DNPR10; CSS11] asks that we not only release the final count, but that we provide a running total, i.e., we release all the partial sums. This task and its variants have proved to be a surprisingly complex problem in the theory of differential privacy [BNS13; BNSV15; BDRS18; ALMM19; KLMNS20; GJK21; CLNSS23].

Continual counting is also critically important in practice. State-of-the-art methods for differentially private machine learning rely on (high-dimensional extensions of) continual counting algorithms to maintain their state in a differentially private manner [KMSTTX21; ZTC22; KMSTTX21; DMRST22; CCMRT22; CCGMMRGTX24; XZACCKMRZ23]. (See Section 2.3 for discussion of this application.) This has motivated significant work on making differentially private continual counting as accurate as possible [Mat93; MNT20; FHU22; HUU23; HUU24; AP24].

Practical algorithms for continual counting add Gaussian noise to the partial sums. However, this noise is not independent: it must be carefully correlated across steps to achieve good utility. Choosing an appropriate multivariate Gaussian can be formulated as a *matrix factorization* problem, as we now describe.

To convey the key ideas in this introduction, we consider summing  $n$  scalars; this discussion readily extends to summing vectors. Let  $x \in \mathbb{R}^n$  contain the  $n$  terms we wish to sum (over  $n$  iterations or steps). Let  $A \in \{0, 1\}^{n \times n}$  be the lower-triangular all-ones matrix so that  $Ax$  is the vector of partial sums – i.e.,  $(Ax)_k = \sum_{i \leq k} x_i$  for each  $k$ . We factor  $A = BC$ , where  $B, C^T \in \mathbb{R}^{n \times n'}$ . The corresponding differentially private continual counting algorithm  $\mathcal{M}$  is given by

$$\mathcal{M}(x) := B(Cx + z) = Ax + Bz = A(x + C^\dagger z), \quad (1.1)$$

where  $z \leftarrow \mathcal{N}(0, \sigma^2 I)$  is a vector of independent Gaussian noise.<sup>1</sup> Equivalently, the output of the mechanism can be written as  $\mathcal{M}(x) = Ax + \hat{z}$  where  $\hat{z} \leftarrow \mathcal{N}(0, \Sigma)$  with  $\Sigma = \sigma^2 BB^T$ .

**Mechanism privacy:** We assume that one person can change only one entry of the input  $x$  by at most 1, which means  $Cx$  can change by at most the maximum column norm of  $C$ , denoted

$$\|C\|_{1 \rightarrow 2} := \max_j \sqrt{\sum_i C_{i,j}^2}.$$

This is the  $L_2$  sensitivity of  $Cx$ . Then, following the standard privacy analysis of Gaussian noise addition [DR14; Ste22], to ensure that  $Cx + z$  is  $(\varepsilon, \delta)$ -differentially private, we scale the noise  $z \leftarrow \mathcal{N}(0, \sigma^2 I)$  to have standard deviation  $\sigma = \zeta \|C\|_{1 \rightarrow 2}$ , where  $\zeta = O(\frac{1}{\varepsilon} \sqrt{\log(1/\delta)})$  is a noise multiplier depending only on the privacy parameters  $(\varepsilon, \delta)$ .<sup>2</sup> By postprocessing, this implies  $\mathcal{M}(x) = B(Cx + z)$  is  $(\varepsilon, \delta)$ -differentially private.

<sup>1</sup> $C^\dagger = A^{-1}B$  is the inverse of  $C$  whenever  $C$  is a square matrix. However, we will also consider settings where  $C$  is not a square matrix and so  $C^\dagger$  is a suitable pseudo-inverse.

<sup>2</sup>Importantly, particularly for machine learning applications, this privacy analysis can be extended to the case where each  $x_k$  is adaptively chosen depending on the prefix sums  $0, \dots, k-1$  already released by the mechanism [DMRST22].

**Mechanism utility:** The error of the mechanism  $\mathcal{M}$  is given by  $\mathcal{M}(x) - Ax = Bz$ . Thus, the root mean squared error of the  $i^{\text{th}}$  partial sum can be calculated as

$$\sqrt{\mathbb{E}_{\mathcal{M}}(\mathcal{M}(x) - Ax)_i^2} = \sqrt{\mathbb{E}_z[(Bz)_i^2]} = \sqrt{\sigma^2 \sum_j B_{i,j}^2},$$

which scales with the norm of the corresponding row of  $B$ . We take our notion of utility to be the maximum<sup>3</sup> such error over all  $n$  partial sums released, which scales with the maximum row norm

$$\|B\|_{2 \rightarrow \infty} := \max_i \sqrt{\sum_j B_{i,j}^2}.$$

Thus, the maximum root mean squared error of the mechanism's answers is

$$\begin{aligned} \max_i \sqrt{\mathbb{E}_{\mathcal{M}}[(\mathcal{M}(x) - Ax)_i^2]} &= \max_i \sqrt{\mathbb{E}_{z \leftarrow \mathcal{N}(0, \sigma^2 I)}[(Bz)_i^2]} = \max_i \sqrt{\sigma^2 \sum_j B_{i,j}^2} \\ &= \sigma \|B\|_{2 \rightarrow \infty} = \zeta \|B\|_{2 \rightarrow \infty} \|C\|_{1 \rightarrow 2} =: \zeta \text{MaxErr}(B, C), \end{aligned} \quad (1.2)$$

where we take  $\sigma = \zeta \|C\|_{1 \rightarrow 2}$  to ensure  $(\epsilon, \delta)$ -differential privacy, and we define

$$\text{MaxErr}(B, C) := \|B\|_{2 \rightarrow \infty} \|C\|_{1 \rightarrow 2} = \sqrt{\max_i \sum_j B_{i,j}^2} \sqrt{\max_j \sum_i C_{i,j}^2}. \quad (1.3)$$

Since the error in Equation 1.2 can be written as a product of the noise multiplier  $\zeta$  (which does not depend on the factorization  $B, C$ ) and  $\text{MaxErr}(B, C)$  (which does not depend on the differential privacy parameters), the same factorization will minimize error for all settings of the privacy parameters. Thus, we suppress  $\zeta$  for the remainder of this paper.

**Matrix factorization:** To summarize, our goal is to solve the matrix factorization problem

$$\text{minimize}_{B,C} \text{MaxErr}(B, C) \quad \text{subject to } BC = A. \quad (1.4)$$

The optimal value of this objective is denoted  $\gamma_2(A)$  and is known as the *gamma-two factorization norm* of  $A$ . Prior work has obtained near-optimal factorizations of  $A$  (and related matrices) [Mat93; MNT20]. The optimal value is

$$\gamma_2(A) := \inf \{ \text{MaxErr}(B, C) : B, C \in \mathbb{R}^{n \times n}, BC = A \} = \frac{\log(n)}{\pi} \pm O(1). \quad (1.5)$$

However, the existing factorizations are either far from optimal in terms of their utility or are impractical due to their high computational cost, which we turn to next.

---

<sup>3</sup>We can also consider other measures of error such as sum of variances (rather than the max). This corresponds to a different norm:  $\mathbb{E}[\|Bz\|_2^2] = \sigma^2 \|B\|_F^2$ .

**Computationally efficient noise generation:** The computational cost of implementing this matrix factorization approach for differentially private continual counting is dominated by the matrix-vector multiplication  $Bz$ , where  $z \in \mathbb{R}^n$  consists of i.i.d. Gaussian noise  $z \leftarrow \mathcal{N}(0, \sigma^2 I)$ . The main computational challenge is that our algorithm cannot store the increments  $x$  or the seed noise  $z$  (or even the matrices  $B$  or  $C$ ) in memory; this would require memory that is linear in the number of steps  $n$ , which can be prohibitively large.

To give a sense of scale, in private machine learning applications, we often need to compute  $n > 10^6$  partial sums (each coming from an iteration of stochastic gradient descent or a related algorithm), over  $m > 10^{10}$  dimensions (the size of the model and its gradients).<sup>4</sup> Thus we need to be able to compute  $\mathcal{M}(x)$  without storing all of the data or noise in memory, as this would take  $nm > 10^{16}$  units of memory – i.e., over 40 petabytes if we use 32-bit floating point numbers.<sup>5</sup>

In this work, we address the computational efficiency of sampling the correlated noise. Specifically, *we provide a matrix factorization mechanism for differentially private continual counting attaining near-optimal error with only polylogarithmic memory overhead.*

**Streaming setting:** Our algorithm receives the coordinates of the input  $x$  and i.i.d. noise  $z$  one at a time and outputs the approximate partial sums  $\mathcal{M}(x)$  one at a time. In particular, we must output each partial sum  $(\mathcal{M}(x))_k \approx \sum_{i \leq k} x_i$  before receiving the next input  $x_{k+1}$ .

Ideally, the memory usage should not grow with the stream length  $n$  at all. If we use the trivial factorization given by  $B = I$  and  $C = A$ , then this corresponds to adding independent noise to each output. This would only require constant memory and constant time per output, but the noise scale  $\text{MaxErr}(B, C) = \sqrt{n}$  is far from optimal. Thus, the challenge is to simultaneously obtain near-optimal error and computational efficiency.

To overcome this challenge we must impose some structure on the factors  $B$  and  $C$  that allows efficient noise generation without significantly increasing the matrix factorization objective  $\text{MaxErr}(B, C)$ . The three structures that have been considered in the literature are (i) lower triangular matrices, (ii) sparse matrices, and (iii) Toeplitz matrices – i.e., constant diagonals,  $\forall i, j \ C_{i,j} = c_{i-j}$  for some vector  $c$  – and combinations of these structures. We focus primarily on lower triangular Toeplitz matrices. The streaming setting naturally corresponds to lower triangular matrices, as it ensures that the  $k$ -th output is only a function of the first  $k$  inputs. Adding the Toeplitz constraint has a minimal impact on the matrix factorization objective (1.4). Furthermore, Toeplitz matrices (and certain specializations we consider) have convenient mathematical and algorithmic properties. See Figure 2 and Section 2.2 for further discussion about lower triangular Toeplitz factorizations versus other factorizations. We also leverage the sparsity in the  $B$  and  $C$  matrices derived from generalizations of the binary tree aggregation mechanism to show additional results.

---

<sup>4</sup>In this introductory discussion, we have considered sums over  $m = 1$  dimension. Extending to  $m > 1$  is essentially a matter of running  $m$  parallel copies of the algorithm. In our technical sections, we take  $x$  to be a matrix whose rows we wish to sum, rather than a vector.

<sup>5</sup>Throughout this paper we treat real computation as atomic; e.g., a real number takes up one unit of memory. In practice, floating point implementations can compromise privacy [Mir12], but techniques exist to ensure differential privacy in discrete settings [CKS20; KLS21].

## 1.1 Our Contributions

Our main theoretical result is a lower triangular Toeplitz matrix factorization that is nearly optimal accompanied by an efficient streaming algorithm for generating the corresponding noise.

**Theorem 1.1** (Main Result – Informal version of Theorem 4.6). *For each integer  $n \geq 1$  and error parameter  $\mu \in (0, 1)$ , there exists a lower triangular Toeplitz matrix factorization  $B, C \in \mathbb{R}^{n \times n}$  with the following properties.*

- **Validity:**  $BC = A$ , where  $A$  is the  $n \times n$  lower-triangular all-ones matrix.
- **Near-optimality:**  $\text{MaxErr}(B, C) \leq \text{OptLTToe}(n) + \mu$ , where

$$\text{OptLTToe}(n) = 1 + \sum_{k=1}^{n-1} \left( 2^{-2k} \binom{2k}{k} \right)^2 \leq 1 + \frac{0.57722 + \log(n)}{\pi}$$

is the optimal value of  $\text{MaxErr}(B, C)$  over all lower triangular Toeplitz factorizations  $A = BC$ .

- **Efficiency:** There exists a streaming algorithm that at each step  $k$  takes as input  $z_k$  and outputs  $(Bz)_k$  (or, equivalently,  $(C^{-1}z)_k$ ) and runs in time and space  $O(\log^2(n/\mu))$ .

In order to ensure that the error term  $\mu$  in our result is  $o(1)$ , we have space and time complexity  $d = \Theta(\log^2 n)$ . It is natural to wonder whether this computational complexity can be improved. We show that it can be improved to  $\tilde{O}(\log n)$  at the expense of a weaker multiplicative near-optimality guarantee for the matrix factorization objective.<sup>6</sup>

**Theorem 1.2** (Secondary Result – Informal version of Theorem 6.6). *For each integer  $n \geq 1$ , there exists an integer  $n' = O(n)$  and a matrix factorization  $B, C^T \in \mathbb{R}^{n \times n'}$  with the following properties.*

- **Validity:**  $BC = A$ , where  $A$  is the  $n \times n$  lower-triangular all-ones matrix.
- **Near-optimality:**  $\text{MaxErr}(B, C) \leq (1 + o(1)) \cdot \text{Opt}(n)$ , where  $\text{Opt}(n) = \frac{\log(n)}{\pi} \pm O(1)$  is the optimal value of  $\text{MaxErr}(B, C)$  over all factorizations.
- **Efficiency:** There exists a streaming algorithm that at each step  $k$  takes as input some coordinates of  $z$  (but never reads the same coordinate more than once) and outputs  $(Bz)_k$  and runs in space (and amortized time per iteration)  $\tilde{O}(\log n)$ .

The optimal matrix factorization objective value over lower-triangular Toeplitz factorizations  $\text{OptLTToe}(n)$  is a small additive constant (specifically, 0.365) away from the optimal over all factorizations  $\text{Opt}(n)$  (cf. Corollary 2.4). Thus, Theorem 1.1's approximation bound on the class of lower triangular Toeplitz matrices can be directly compared to Theorem 1.2's approximation bound over all possible factorizations.

More generally, we can smoothly trade off between the matrix factorization objective and computational efficiency. That is, we can interpolate between Theorems 1.1 and 1.2; see Proposition 6.5 for a general statement.

<sup>6</sup>In this problem, the multiplicative constants in the error have a larger practical impact on the mechanism's utility than an additive error. For instance, the binary tree mechanism is suboptimal by a multiplicative factor of  $\pi / \log 2 \approx 4.5$ , which yields a much worse  $\text{MaxErr}$  than the factorization of Fichtenberger, Henzinger, and Upadhyay [FHU22] (cf. Figure 2, left).

Theorem 1.2 does not generate a lower triangular Toeplitz factorization; in fact it does not even produce a square factorization. It also gives a weaker multiplicative near-optimality guarantee. We leave it as an interesting open problem whether it is possible to improve on  $\log^2 n$  space complexity with a Toeplitz factorization or with  $\text{MaxErr}(B, C) \leq \text{Opt}(n) + O(1)$ .

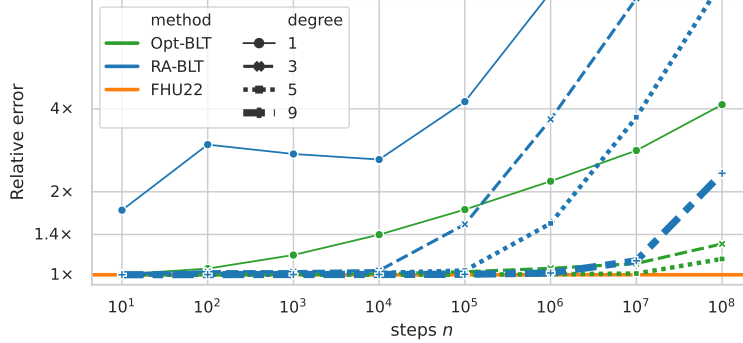


Figure 1: Ratio of  $\text{MaxErr}(B, C)$  of our RA-BLT and Opt-BLT mechanisms for different numbers of steps  $n$  and degrees  $d$  (which corresponds directly to the space complexity) over that of the optimal Toeplitz mechanism of Fichtenberger, Henzinger, and Upadhyay [FHU22]. This illustrates that even with modest degree  $d$ , we obtain very good  $\text{MaxErr}(B, C)$  even for large numbers of steps  $n$ . For example, Opt-BLT with  $d = 5$  is within 1% of optimal for  $n = 10^7$  (we do not plot Opt-BLT for  $d = 9$ ).

**Practical algorithms:** Since our work is directly motivated by practical considerations, we also study the problem numerically. Specifically, we show that the method behind Theorem 1.1 yields a factorization that is indistinguishable from optimal Toeplitz mechanism for practical purposes, but which has a highly efficient procedure for generating the noise. Figure 1 shows the ratio of  $\text{MaxErr}(B, C)$  for our algorithm over the optimal OptLTToe( $n$ ).

To prove Theorem 1.1 we provide closed-form parameters for the matrix factorization. This is already reasonably practical, but we further optimize the parameters numerically to obtain even better algorithms. In order to optimize the matrix factorization we (i) choose an appropriate parameterization for the class of factorizations  $B, C$  that we consider, (ii) give an efficiently computable expression for the objective  $\text{MaxErr}(B, C)$  in terms of this parameterization, and (iii) show that this expression is differentiable. This allows us to numerically optimize our factorization using gradient-based methods. While we do not prove that this optimization procedure converges, in practice it yields significantly better solutions than the closed-form parameters used to prove Theorem 1.1.

To further illustrate the power of our approach, we show in Section 8 that even with constant memory, we can asymptotically attain  $\text{MaxErr}(B, C) = O(n^{1/6})$ . This is already an improvement over  $\text{MaxErr}(I, A) = \sqrt{n}$  obtained by the trivial factorization.

**Lower bounds:** In Section 2.2 we prove a lower bound that exactly characterizes OptLTToe( $n$ ). Specifically, we show that the lower triangular Toeplitz factorization of Fichtenberger, Henzinger, and Upadhyay [FHU22] is precisely optimal for this class.



Further, in Section 7, we develop *numerical* lower bounds on the objective of any matrix factorization for various classes of matrices for any fixed size  $n$ . In particular, we give lower bounds for arbitrary lower triangular matrices, arbitrary Toeplitz matrices, and Toeplitz matrices that correspond to our algorithm with a specific constant memory constraint.

## 1.2 Our Algorithms & Techniques

The starting point for our main result is the lower triangular Toeplitz factorization of Fichtenberger, Henzinger, and Upadhyay [FHU22], which we show is in fact the optimal lower triangular Toeplitz factorization in Section 2.2. This factorization is given by  $B_{i,j} = C_{i,j} = f_{i-j}$  where the sequence  $f_0, f_1, \dots$  is the coefficients of a Taylor series<sup>7</sup>:

$$\frac{1}{\sqrt{1-x}} = f_0 + f_1x + f_2x^2 + f_3x^3 + \dots \quad (1.6)$$

This sequence satisfies the recurrence  $f_k = (1 - 1/2k) \cdot f_{k-1}$  for  $k \geq 1$  with  $f_0 = 1$ . Our task in this case is to compute the correlated noise  $(Bz)_k = \sum_{j=0}^k B_{k,j} \cdot z_j = \sum_{j=0}^k f_{k-j} \cdot z_j$  in a streaming fashion for  $k \in [n] := \{0, 1, \dots, n-1\}$ .<sup>8</sup> Unfortunately, the factorization of Equation 1.6 does not seem to admit an efficient sampling or noise generation procedure.

**BLTs: Buffered Linear Toeplitz Matrices (§3):** Suppose the sequence instead satisfied a linear recurrence  $f_k = q \cdot f_{k-1}$  for all  $k \geq 1$ . Then

$$(Bz)_k = \sum_{j=0}^k f_{k-j} \cdot z_j = f_0 \cdot z_k + \sum_{j=0}^{k-1} (q \cdot f_{k-j-1}) \cdot z_j = f_0 \cdot z_k + q \cdot (Bz)_{k-1}. \quad (1.7)$$

This equation gives us an efficient algorithm: given the previous output  $(Bz)_{k-1}$  and the current input  $z_k$ , we can compute the current output  $(Bz)_k$ . The memory requirement of this algorithm is simply to store the previous output in a single memory buffer.

Next, suppose the sequence instead satisfied a linear recurrence of the form

$$f_k = q_1 \cdot f_{k-1} + q_2 \cdot f_{k-2} + \dots + q_d \cdot f_{k-d}. \quad (1.8)$$

As we explain next, this recurrence gives an algorithm where the memory requirement is to only store  $d$  memory buffers.

The recurrence  $f_k = q f_{k-1}$  implies the closed form  $f_k = q^k f_0$ . Similarly, the recurrence in Equation 1.8 implies a closed form expression  $f_k = u^T W^k v$  where  $W \in \mathbb{R}^{d \times d}$  is a matrix and  $u, v \in \mathbb{R}^d$  are vectors. This closed form is what we use in Section 3.3 for our Algorithm 1. Specifically, we can extend Equation 1.7 to this matrix-power closed form

$$(Bz)_k = \sum_{j=0}^k f_{k-j} \cdot z_j = \sum_{j=0}^k u^T W^{k-j} v \cdot z_j = u^T S_{k+1}$$

<sup>7</sup>We overload notation and use  $x$  both for the input to the algorithm ( $x \in \mathbb{R}^n$ ), and the indeterminate in polynomials and generating functions; the meaning should be clear from context.

<sup>8</sup>We zero-index sequences, vectors, and matrices throughout. See Table 1 for a summary of symbols and notation.



for a suitable state vector  $S_k \in \mathbb{R}^d$ , stored in memory. Namely, we initialize  $S_0 = 0$  and, at each iteration, our algorithm updates

$$S_{k+1} = v \cdot z_k + W S_k$$

and then outputs  $(Bz)_k = u^T S_{k+1}$ . We refer to the entries of  $S_k$  as the  $d$  *buffers* of our algorithm.<sup>9</sup> The updates to the buffers on each step are an arbitrary linear function of the previous step’s buffers ( $W S_k$ ) and the current input ( $v z_k$ ), and the output on each step is an arbitrary linear combination of the buffers ( $u^T S_{k+1}$ ). Hence, we term this class “**Buffered Linear Toeplitz matrices**” (BLTs). We overload the acronym and write BLTs as a shorthand encompassing matrices, factorizations, and mechanisms.

Unfortunately, the optimal factorization [FHU22] does *not* satisfy a recurrence like Equation 1.8 and cannot be expressed as a BLT. Hence, our approach is to *approximate* the optimal factorization using BLTs.

**Designing BLTs via rational function approximation (§4):** If we view the Toeplitz sequence  $f_0, f_1, \dots$  as being defined by an ordinary generating function<sup>10</sup>  $f$  as in Equation 1.6, it turns out that satisfying a linear recurrence as in Equation 1.8 is equivalent to the function being rational with degree at most  $d$ , i.e.,  $f(x) = p(x)/q(x)$  for polynomials  $p$  and  $q$  of degree  $\leq d$ . (This equivalence is analogous to the fact that a real number is rational if and only if its decimal representation is repeating.)

This equivalence also suggests our first approach to designing BLTs: we need a low-degree rational *approximation* to the function  $f(x) = 1/\sqrt{1-x}$  from Equation 1.6 that underlies the optimal factorization of Fichtenberger, Henzinger, and Upadhyay [FHU22].

We appeal to known results in approximation theory. Specifically, it is known that the function  $x \mapsto \sqrt{1-x}$  can be uniformly approximated on the unit complex disc  $\{x \in \mathbb{C} : |x| \leq 1\}$  with error  $\eta > 0$  by a rational function of degree  $d = O(\log^2(1/\eta))$  [New64; GT19]. That is, there exists a rational function  $r$  of degree  $\leq d$ , such that  $|r(x) - \sqrt{1-x}| \leq \eta$  for all  $x \in \mathbb{C}$  with  $|x| \leq 1$ .

It “only” remains to translate this approximation guarantee back to the matrix factorization objective. Parseval’s identity allows us to bound the difference between the sequences of Taylor coefficients in terms of an integral: Suppose  $f(x) = \sum_{k=0}^{\infty} f_k x^k$  and  $\tilde{f}(x) = \sum_{k=0}^{\infty} \tilde{f}_k x^k$ . Then

$$\sum_{k=0}^{\infty} |f_k - \tilde{f}_k|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |f(x(\theta)) - \tilde{f}(x(\theta))|^2 d\theta \quad \text{where } x(\theta) = \exp(\sqrt{-1}\theta). \quad (1.9)$$

In our case,  $f(x) = 1/\sqrt{1-x}$  corresponds to the optimal lower triangular Toeplitz factorization while  $\tilde{f}(x)$  is our rational approximation – either  $\tilde{f}(x) = 1/r(x)$  or  $\tilde{f}(x) = r(x)/(1-x)$ , where  $r(x) \approx \sqrt{1-x}$ . We are interested in a finite sum  $\sum_{k=0}^{n-1} |f_k - \tilde{f}_k|^2$ , rather than the infinite sum

<sup>9</sup>Recall that while we treat these as scalars here, in practical ML applications for example, each of these has size equal to the number of parameters of the model being trained, e.g. possibly  $m > 10^9$ , so keeping  $d$  to a small constant is critical.

<sup>10</sup>We use the terms generating function, ordinary generating function, and Taylor series interchangeably; the generating function view of  $f$  emphasizes the sequence being generated and requires (only) a formal power series, while the Taylor series view emphasizes  $f$  is a real or complex function. As long as  $f$  is analytic in a non-empty open neighborhood of zero, no ambiguity is introduced by these two views, see e.g. Thm. 2.8 (Transfer principle) of Kauers and Paule [KP11].

in Equation 1.9 (which does not converge in our setting). Thus we consider a weighted version of Parseval's identity where the integral goes around a circle in the complex plane centered at 0 with radius  $e^{-1/2n}$ . Note that we require the approximation guarantee to hold on the complex plane, not just the real line. Once we have this bound on  $\sum_{k=0}^{n-1} |f_k - \tilde{f}_k|^2$ , the near optimality guarantee of Theorem 1.1 follows from the triangle inequality.

The time and space requirement of Theorem 1.1 is  $O(\log^2 n)$ . This dependence arises from the degree of the rational approximation. A degree at least  $\Omega(\log^2(1/\eta))$  is necessary for approximating  $\sqrt{1-x}$  with error  $\leq \eta$  even for real values  $x \in [-1, 1]$  [New64]. Thus, unless we can exploit some slack in our analysis, it seems we require different techniques to bring the space down to  $O(\log n)$ .

The proof of Theorem 1.1 gives an explicit rational function approximation, which we can directly convert into a matrix factorization and feed into our algorithm. We term the BLTs coming from this approach RA-BLTs, with each choice of the degree  $d$  leading to a different (and successively better) approximation to the optimal Toeplitz factorization.

**Designing BLTs via direct optimization (§5):** Recall that our goal is to minimize  $\text{MaxErr}(B, C) = \text{MaxErr}(AC^{-1}, C)$ . In order to ensure computational efficiency, we restrict the matrix  $C$  to the class of BLTs. While approximation theory lets us directly construct a near-optimal  $C$  as outlined above, we can also approach this as an optimization problem and optimize the BLT parameters that define  $C$  numerically.

This optimization is far from straightforward. The class of BLT matrices can be parameterized in multiple ways. Converting the parameters for  $C$  into  $\text{MaxErr}(AC^{-1}, C)$  is nontrivial to compute – much less optimize – when the size  $n$  is large. Nevertheless, the class of BLTs is algebraically closed under multiplication and addition, and this structure combined with the connection to rational generating functions, provides powerful tools for reasoning about them.

We give a parameterization for the class of BLTs that allows us to efficiently compute  $\text{MaxErr}$  in time practically independent of the size  $n$ , specifically  $\mathcal{O}(\text{poly}(d) \log n)$ , and also to compute gradients. Being able to compute gradients allows us to optimize  $\text{MaxErr}(AC^{-1}, C)$  numerically.

The first challenge is that we need to be able to effectively parameterize both  $C$  and  $AC^{-1}$ . If  $C$  is a BLT, then, as discussed above, its Toeplitz coefficients are given by the Taylor series for a rational generating function  $c(x) = p(x)/q(x)$  for polynomials  $p$  and  $q$ . Further, these coefficients have a simple closed-form expression given by Lemma 3.2 (see also Equation 5.2). Of course  $1/c(x) = q(x)/p(x)$  is also a rational function, and it in fact generates the Toeplitz coefficients of  $C^{-1}$  (Lemma 3.1), and so  $C^{-1}$  is also a BLT. The lower-triangular matrix of ones,  $A$ , is trivially a BLT, and so Lemma 3.1 also implies  $B = AC^{-1}$  is a BLT.

In Section 5, Lemma 5.2 shows that given a BLT  $C$ , we can (explicitly and in closed-form) derive the BLT parameters of  $C^{-1}$ , and hence a closed form for its Toeplitz coefficients.<sup>11</sup> Using these closed-form expressions for the Toeplitz coefficients enables us to directly compute  $\text{MaxErr}(B, C)$  for  $B, C \in \mathbb{R}^{n \times n}$  in time  $\mathcal{O}(\text{poly}(d) \log n)$ , see Lemmas 5.3 and 5.4.

This is immediately useful, for example in Figure 2 allowing us to plot the performance of our mechanisms for  $n$  up to  $10^8$  using only a few seconds of computer time.<sup>12</sup>

<sup>11</sup>More precisely, Lemma 5.2 uses a parameterization of  $p$  and  $q$  which leads to closed forms for both  $C$  and  $C^{-1}$ .

<sup>12</sup>We could have in fact scaled the  $\text{MaxErr}$  calculations for our mechanisms to arbitrary  $n$ ; the bottleneck is in the computation of the exact  $\text{MaxErr}$  for [FHU22], which requires time  $\mathcal{O}(n)$ .

More importantly, however, Section 5 shows computing  $\text{MaxErr}(B, C)$  is a differentiable function of the parameters of the BLTs  $B$  and  $C$ , and hence we can use a gradient-based optimization method such as L-BFGS to directly minimize  $\text{MaxErr}$  targeting a specific number of steps  $n$ , where  $B$  and  $C$  are BLTs with  $d$  buffers defined by  $2d$  parameters. We term these  $\text{Opt-BLT}$  mechanisms, and they perform extremely well in practice. For example, for  $n = 10^7$ , a  $\text{Opt-BLT}(d=4)$  has  $\text{MaxErr}$  that is  $1.032\times$  that of the optimal Toeplitz factorization [FHU22], and  $\text{Opt-BLT}(d=7)$  is  $1.001\times$  optimal; for smaller  $n$  the results are even better, for example for  $n = 10^4$ ,  $\text{Opt-BLT}(d=4)$  achieves  $1.001\times$  optimal. Figure 2 gives more complete results.

**Generalizations of the binary tree mechanism (§6):** The starting point for Theorem 1.2 is the binary tree mechanism of Dwork, Naor, Pitassi, and Rothblum [DNPR10] and Chan, Shi, and Song [CSS11]. The binary tree mechanism can be viewed as a recursive construction of a matrix factorization. A recursion of depth  $\ell$  yields a matrix factorization of size  $n = 2^\ell$  and an algorithm running in time and space  $O(\ell)$ . The binary tree mechanism does not produce a Toeplitz or square matrix factorization; the structure that it relies on for computational efficiency is sparsity. The matrix factorization objective  $\text{MaxErr}(B, C)$  for this construction is  $O(\log n)$  – that is, it is within a constant factor of optimal. Specifically, the binary tree mechanism is asymptotically a factor of  $\frac{\pi}{\log 2} \approx 4.5$  from optimal. This factor is significant in practice, as shown in Figure 2.

We combine the binary tree mechanism’s recursive approach with Theorem 1.1 to get the best of both worlds – near-optimal constants and  $\tilde{O}(\log n)$  space. This proves Theorem 1.2.

We illustrate one step of the recursive construction using the following example for size  $n = 6$ . We can decompose the  $6 \times 6$  all-ones lower triangular matrix  $A^{(6)}$  into a sum of expressions involving a  $2 \times 2$  all-ones lower triangular matrix  $A^{(2)}$  and a  $3 \times 3$  all-ones lower triangular matrix  $A^{(3)}$ :

$$\begin{aligned} A^{(6)} &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \end{pmatrix} \quad (1.10) \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \\ &= I \otimes A^{(2)} + (S^{(3)} A^{(3)}) \otimes (\mathbf{1}\mathbf{1}^T), \end{aligned}$$

where  $\otimes$  denotes the Kronecker product,<sup>13</sup>  $I$  is the identity matrix,  $\mathbf{1}$  is the all-ones vector, and

$S^{(3)} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$  is a non-cyclic shift matrix. Equation 1.10 can be used to take factorizations of size 2 and 3 and combine them into a factorization of size 6. Namely, if  $A^{(2)} = B^{(2)}C^{(2)}$  and

<sup>13</sup>A key property of the Kronecker product is that  $(A \cdot B) \otimes (C \cdot D) = (A \otimes C) \cdot (B \otimes D)$ .

$A^{(3)} = B^{(3)}C^{(3)}$ , then

$$\begin{aligned}
A^{(6)} &= I \otimes A^{(2)} + (S^{(3)}A^{(3)}) \otimes (\mathbf{1}\mathbf{1}^T) & (\text{Equation 1.10}) \\
&= (I \cdot I) \otimes (B^{(2)} \cdot C^{(2)}) + (S^{(3)}B^{(3)} \cdot C^{(3)}) \otimes (\mathbf{1} \cdot \mathbf{1}^T) \\
&= (I \otimes B^{(2)}) \cdot (I \otimes C^{(2)}) + (S^{(3)}B^{(3)} \otimes \mathbf{1}) \cdot (C^{(3)} \otimes \mathbf{1}^T) \\
&= \underbrace{(I \otimes B^{(2)} \mid S^{(3)}B^{(3)} \otimes \mathbf{1})}_{=B^{(6)}} \cdot \underbrace{\begin{pmatrix} I \otimes C^{(2)} \\ C^{(3)} \otimes \mathbf{1}^T \end{pmatrix}}_{=C^{(6)}}. & (1.11)
\end{aligned}$$

The factors  $B^{(6)}$  and  $C^{(6)}$  are non-square matrices represented as block matrices. The reason we move to non-square matrices is that Equation 1.10 decomposes  $A^{(6)}$  as a *sum* of two matrix products and we must re-express this as a single matrix product.

Roughly speaking, the binary tree mechanism corresponds to starting with a factorization of size 2 and applying a recursive step similar to the above  $\ell - 1$  times to obtain a factorization of size  $2^\ell$ . Rather than starting with a factorization of size 2, we can start with a larger factorization of size  $n_1$  given by Theorem 1.1 and then repeat the recursive step above  $\ell - 1$  times to obtain a factorization of size  $n_1^\ell$ . Intuitively, by picking a larger factorization as the starting point we get closer to the optimal constant. With careful analysis and the right choice of parameters, this yields Theorem 1.2.

This recursive construction attains excellent asymptotics, but, for practical parameter regimes, we find that the BLT approach is practically indistinguishable from the optimal Toeplitz mechanism.

### 1.3 An Empirical Comparison of Mechanisms

In this section, we provide an empirical comparison of the primary DP mechanisms discussed, demonstrating their effectiveness in practical regimes.

Figure 2 compares mechanisms (and lower bounds) in terms of MaxErr from  $n = 1$  to  $n = 10^6$  iterations. The sub-optimality of the binary tree mechanism is immediately clear. This plot also shows that little is lost by the restriction from general matrix mechanisms to Toeplitz mechanisms, where Fichtenberger, Henzinger, and Upadhyay [FHU22] provide the optimal (but inefficient) construction. Our BLT mechanisms essentially match this performance, while requiring time and memory  $\tilde{\mathcal{O}}(1)$  instead of  $\mathcal{O}(n)$ .

Figure 1 provides a detailed comparison of our BLT mechanisms and the optimal Toeplitz mechanism. Several important conclusions can immediately be drawn: (1) For both RA-BLT and Opt-BLT, increasing the degree (number of allowed memory buffers) increases performance. (2) A larger number of steps  $n$  requires a higher number of buffers  $d$  for both of our approaches; this is expected and necessary, as shown by our theory, see Remark 5.5. (3) A key point to emphasize is that each blue line for RA-BLT corresponds to a single mechanism (a fixed rational approximation); for Opt-BLT, we compute an optimized matrix factorization of the given degree for each different  $n$ . This specialization of the mechanism to the specific anticipated number of steps  $n$  is critical to the advantage enjoyed by this approach. For example, we see Opt-BLT with only 5 buffers outperforms RA-BLT with 9 buffers across the full range of  $n$ .

It is of course possible to run Opt-BLT mechanisms for a different number of steps than the optimization targeted. Figure 3 (Left column) explores this. We construct three fixed Opt-BLT factorizations, optimized for  $n \in \{100, 1000, 10000\}$ , and compare their performance (in terms of MaxErr relative to the optimal Toeplitz mechanism) across a range of steps, from  $10^1$  to  $10^5$ . As expected, the mechanisms work best for the  $n$ s for which they were optimized; however, the excess error is highly asymmetric; a mechanism optimized for  $n^*$  will generally perform well for  $n < n^*$  steps, but can quickly perform very badly when  $n > n^*$ . This is expected when one considers that Opt-BLT should intuitively be ensuring a good approximation of the optimal Toeplitz coefficients  $r_0, \dots, r_{n^*-1}$ , but the approximation of the optimal Toeplitz coefficients for larger  $n$  can become arbitrarily bad. We see this in Figure 3 (Middle column), where we compare the Toeplitz coefficients defining  $C$  and  $B$  to the optimal coefficients corresponding to the generating function  $1/\sqrt{1-x}$ . To emphasize this issue, we optimize for  $n^* = 100$ , and consider degree  $d = 2$ . The Opt-BLT factorization provides a better approximation to the optimal coefficients for  $n$  up to 100 compared to RA-BLT( $d=2$ ) (which does not depend on  $n$ ), and a generally worse approximation beyond that. Figure 3 (Right column) shows that while both Opt-BLT and RA-BLT correspond to “reasonable” approximations to  $\sqrt{1-x}$  (top), they distribute their errors in the approximation very differently (bottom).

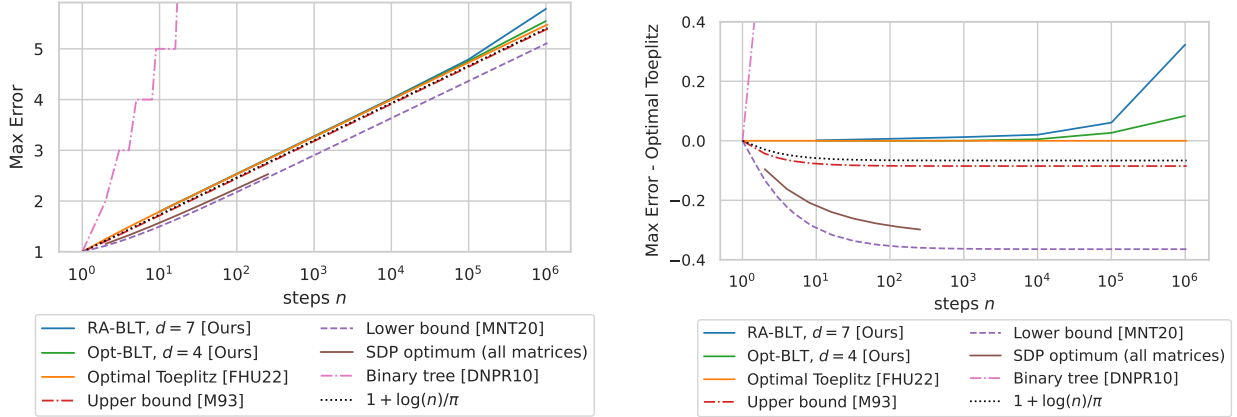


Figure 2: Comparison of known upper and lower bounds for factorizations  $A = BC$  of the all-ones lower triangular matrix  $A_{i,j} = \mathbb{I}[i \geq j]$ . Note that this includes non-Toeplitz factorizations. This illustrates that there is a small gap between lower triangular Toeplitz factorizations and general factorizations; furthermore this gap is asymptotically constant. **Left:** Vertical axis is  $\text{MaxErr}(B, C)$ . **Right:** Vertical axis is  $\text{MaxErr}(B, C) - \text{OptLTToe}(n)$ .

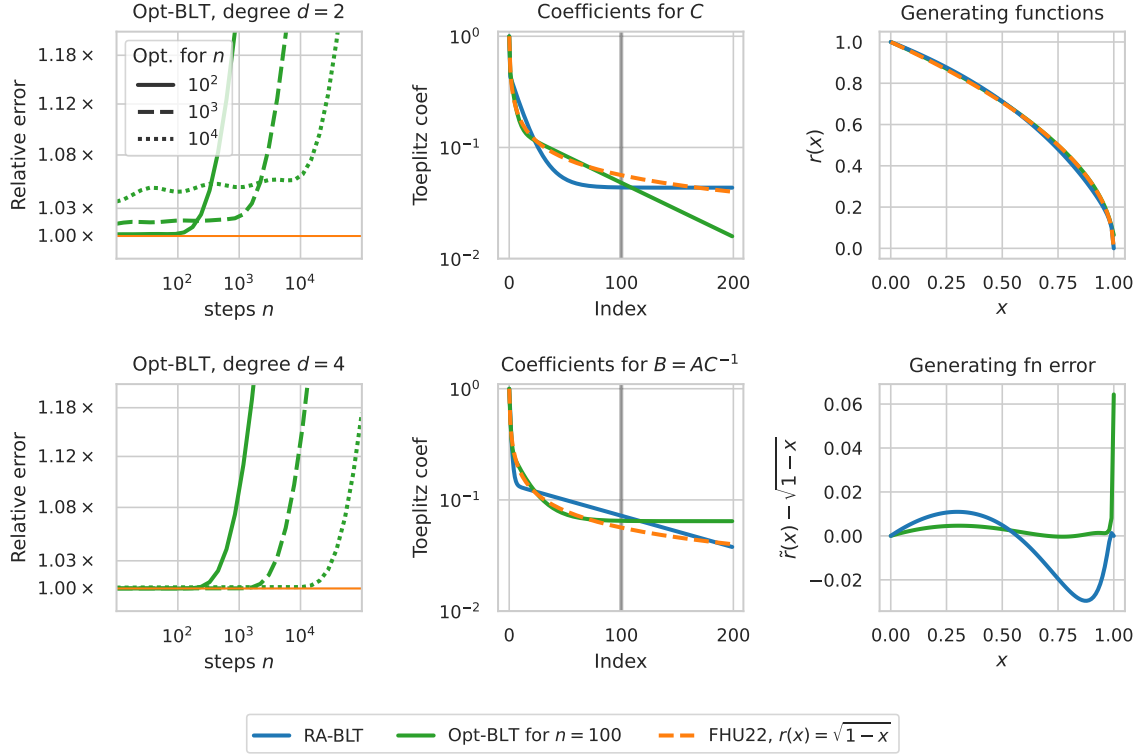


Figure 3: (Left column) Comparison of three fixed Opt-BLT mechanisms across a range of  $n$  (extending beyond the optimization targets). (Center) Comparing the first 200 Toeplitz coefficients defining the  $C$  and  $B$  matrices for Opt-BLT and RA-BLT for degree  $d = 2$ , with the Opt-BLT mechanism optimized for  $n^* = 100$ . (Right) Differences in the generating functions for the Opt-BLT and RA-BLT factorizations of the middle column.

## 2 Background

We start by formally setting up the problem where each stream entry is a vector of dimension  $m$  (as opposed to a scalar in the introduction). Fix a stream length  $n \in \mathbb{N}$  and let

$$A = A^{(n)} := \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & 1 \end{pmatrix} \in \{0, 1\}^{n \times n} \quad (2.1)$$

be the all-ones lower triangular matrix given by  $A_{i,j}^{(n)} = 1 \iff i \geq j$  and  $A_{i,j} = 0 \iff i < j$  for all  $i, j \in [n]$ . Note that  $A$  is invertible and its inverse is a lower triangular bi-diagonal matrix:

$$A^{-1} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 1 \end{pmatrix} \in \{-1, 0, 1\}^{n \times n}. \quad (2.2)$$

Our goal is to compute a matrix factorization  $A = BC$  where  $B, C^T \in \mathbb{R}^{n \times n'}$  that minimizes

$$\text{MaxErr}(B, C) := \|B\|_{2 \rightarrow \infty} \cdot \|C\|_{1 \rightarrow 2} := \sqrt{\max_{i \in [n]} \sum_{j \in [n']} B_{i,j}^2} \cdot \sqrt{\max_{j \in [n]} \sum_{i \in [n']} C_{i,j}^2}. \quad (2.3)$$

Simultaneously we want computational efficiency in the following sense. We need to generate samples from  $BZ \in \mathbb{R}^{n \times m}$  where  $Z \in \mathbb{R}^{n' \times m}$  is a matrix with independent standard Gaussian entries. We want to do this in a streaming setting where we output one row at a time and the memory is limited. Ideally the memory should be linear in  $m$  and constant or logarithmic in  $n$ .

We do not restrict  $B$  and  $C$  to be square matrices. Non-square factorizations may be advantageous from a computational perspective. However, computational efficiency aside, we can assume without loss of generality that they are square (i.e.,  $n' = n$ ) by taking a singular value decomposition (SVD) of the factors and discarding the rows/columns that do not correspond to a non-zero singular value.

It is also natural to restrict  $B$  and  $C$  to be lower triangular matrices, like  $A$ . Lower triangular structure implies each intermediate output  $(Cx)_k$  is only a function of the inputs seen so far  $x_0, x_1, \dots, x_k$  and not future inputs  $x_{k+1}, \dots, x_{n-1}$ . This is particularly valuable for the privacy analysis in the case where the input stream  $x_1, \dots, x_k$  is chosen adaptively, such as in machine learning applications. In the adaptive setting, each input  $x_k$  may depend on the previous intermediate outputs  $(Cx)_0, (Cx)_1, \dots, (Cx)_{k-1}$ . Lower triangular structure prevents a cyclic dependency. Fortunately, we can assume the factors are lower triangular without loss of generality, see for example Prop 2.2 of [DMRST22].



## 2.1 Prior Work on Matrix Factorizations for Continual Counting

There has been a *lot* of work on factorizing the all-ones lower triangular matrix. However, most of that work optimizes the matrix factorization objective  $\text{MaxErr}(B, C)$  with little regard for the computational efficiency of sampling the correlated noise  $Bz$  from i.i.d. seed noise  $z \leftarrow \mathcal{N}(0, \sigma^2 I)$ .

The binary tree mechanism [DNPR10; CSS11] implicitly constructs a factorization which optimizes  $\text{MaxErr}(B, C)$  up to constant factors and is efficiently computable. Prior to our work, this is the only known approach offering both efficient computation and some form of near-optimality. The binary tree factorization can be expressed recursively as  $B^{(1)} = C^{(1)} = (1) \in \{0, 1\}^{1 \times 1}$  and

$$B^{(2n)} = \begin{pmatrix} B^{(n)} & 0 & 0 \\ 0 & B^{(n)} & \vec{1} \end{pmatrix} \in \{0, 1\}^{2n \times (4n-1)} \quad \text{and} \quad C^{(2n)} = \begin{pmatrix} C^{(n)} & 0 \\ 0 & C^{(n)} \\ \vec{1}^T & 0 \end{pmatrix} \in \{0, 1\}^{(4n-1) \times 2n}. \quad (2.4)$$

Note that this only gives a factorization of  $A^{(n)}$  when  $n$  is a power of 2. By discarding rows/columns we can extend this to any  $n$ . The objective value is given by

$$\text{MaxErr}(B, C) = \|B^{(2^\ell)}\|_{2 \rightarrow \infty}^2 = \|C^{(2^\ell)}\|_{1 \rightarrow 2}^2 = \ell + 1 = \left\lceil \frac{\log n}{\log 2} \right\rceil + 1. \quad (2.5)$$

The recursive formulation naturally leads to an efficient sampling algorithm for the correlated noise  $Bz$  using only  $O(\ell)$  space. Subsequent work has attempted to improve the constants [QYL13; Hon15; AP24].

Observe that the binary tree factorization produces sparse matrices. Specifically, each row of  $B^{(n)}$  and column of  $C^{(n)}$  only has  $O(\log n)$  nonzero entries. Intuitively, this sparsity is what enables efficient computation; that is,  $(Bz)_k$  only depends on logarithmically many elements of  $z$ . However, our experience suggests that sparsity requires non-square matrices and cannot produce near-optimal factorizations like BLTs.

Fichtenberger, Henzinger, and Upadhyay [FHU22] give the following elegant explicit lower triangular Toeplitz factorization. Let  $f_0 = 1$  and, for  $k \in \mathbb{Z}_{\geq 1}$ , let  $f_k = f_{k-1} \cdot (1 - 1/2k)$  and  $f_{-k} = 0$ . Equivalently,  $f_k = 4^{-k} \binom{2k}{k} \leq \frac{1}{\sqrt{\pi k}}$  for  $k \geq 1$  [elk]. Define  $B, C \in \mathbb{R}^{n \times n}$  by  $B_{i,j} = C_{i,j} = f_{i-j}$  for all  $i, j \in [n]$ , which we denote  $B = C = M((f_i)_{i=0}^{n-1})$  or simply  $M(f, n)$ . Then, we have that  $A = BC$  and<sup>14</sup>

$$\text{MaxErr}(B, C) = \|B\|_{2 \rightarrow \infty}^2 = \|C\|_{1 \rightarrow 2}^2 = \sum_{k=0}^{n-1} f_k^2 \leq 1 + \frac{\log(n) + \gamma}{\pi}, \quad (2.6)$$

where  $\gamma \leq 0.57722$  is the Euler-Mascheroni constant [SW]. This factorization turns out to be optimal among the class of lower triangular Toeplitz factorizations; we prove this in Proposition 2.2.

However, unlike the binary tree mechanism, these Toeplitz matrices are dense and we do not know how to efficiently generate noise according to this factorization in the streaming setting.

The  $\gamma_2$  factorization norm is in fact a norm, i.e., it satisfies the triangle inequality [TJ89].

<sup>14</sup>Unfortunately there appears to be an off-by-one error in the proof of the bound given by Theorem 1 of Fichtenberger, Henzinger, and Upadhyay [FHU22]; this has since been corrected.

Furthermore, there is a dual characterization of the  $\gamma_2$  factorization norm [LSŠ08, Theorem 9].

$$\begin{aligned}\gamma_2(A) &:= \inf\{\text{MaxErr}(B, C) : B, C \in \mathbb{R}^{n \times n}, A = BC\} \\ &= \sup\left\{\|P^{1/2} \cdot A \cdot Q^{1/2}\|_{\text{trace}} : \begin{array}{l} P, Q \text{ non-negative diagonal matrices} \\ \text{with } \text{trace}(P) = \text{trace}(Q) = 1 \end{array}\right\},\end{aligned}\quad (2.7)$$

where  $\|M\|_{\text{trace}} = \text{trace}(\sqrt{M^T M})$  is the sum of the absolute singular values.

Using the triangle inequality, Mathias [Mat93, Corollary 3.5] gives a non-constructive upper bound

$$\gamma_2(A) \leq \frac{1}{2} + \frac{1}{2n} \sum_{j=1}^n \frac{1}{\sin\left(\pi \frac{2j-1}{2n}\right)} \leq 1 + \frac{\log(n)}{\pi}. \quad (2.8)$$

On the other hand, using the dual characterization (Equation 2.7 with  $P = Q = \frac{1}{n}I$ ), Matoušek, Nikolov, and Talwar [MNT20] give a lower bound<sup>15</sup>

$$\gamma_2(A) \geq \frac{1}{2n} \sum_{j=1}^n \frac{1}{\sin\left(\pi \frac{2j-1}{4n+2}\right)} \geq \frac{\log(2n+1)}{\pi}. \quad (2.9)$$

The upper bound in Equation 2.6 or Equation 2.8 and the lower bound in Equation 2.9 match up to a small *additive* constant. Thus the optimal value of the matrix factorization objective is  $\gamma_2(A) = \frac{\log n}{\pi} \pm \text{constant}$ . Numerically, the maximum gap between the Toeplitz upper bound in Equation 2.6 and the lower bound in Equation 2.9 is less than 0.365.

Figure 2 shows how these upper and lower bounds compare. The binary tree mechanism has the advantage of computational efficiency, but as we can see, it is far from optimal for the objective. The leading term for the binary tree is  $\frac{\log n}{\log 2}$ , while the optimal leading term is  $\frac{\log n}{\pi}$ . Thus the binary tree is asymptotically suboptimal by a multiplicative factor of  $\pi / \log 2 \approx 4.5$ .

## 2.2 Lower Triangular Toeplitz Factorizations versus General Factorizations

Our work mainly focuses on lower triangular Toeplitz factorizations. In particular, Theorem 1.1 gives such a factorization and proves near-optimality with respect to this class. (Although we consider non-square factorizations in Section 6, where we prove Theorem 1.2.) This structure is essential to our algorithms in Sections 3 to 5. However, it is natural to wonder how much we lose in terms of the objective  $\text{MaxErr}(B, C)$  by restricting  $B$  and  $C$  to be lower triangular Toeplitz matrices.

In this subsection we discuss the gap between lower triangular Toeplitz factorizations and general factorizations. We do not have an explicit construction for the optimal general matrix factorization or even a formula for optimal value  $\gamma_2(A)$ . For lower triangular Toeplitz matrix factorizations the best factorization we have is that of Fichtenberger, Henzinger, and Upadhyay [FHU22]. We show that this is in fact optimal among the class of lower triangular Toeplitz factorizations. This is – to the best of our knowledge – a novel result which may be of independent interest.

We begin with a lemma stating some basic properties of the factorization of Fichtenberger, Henzinger, and Upadhyay [FHU22] and we give a proof for completeness.

<sup>15</sup>Proposition 4.1 of Matoušek, Nikolov, and Talwar [MNT20] simply states an  $\Omega(\log n)$  lower bound, but these sharper expressions can easily be extracted from the proof. Mathias [Mat93] also gives a lower bound which differs from the upper bound in Equation 2.8 by less than an additive  $\frac{1}{2}$ , but this seems slightly weaker than Equation 2.9.

**Lemma 2.1.** Define a sequence  $f_0, f_1, \dots$  by  $f_0 = 1$  and  $f_k = (1 - 1/2k)f_{k-1}$  for all  $k \geq 1$ . Then, for all  $k \geq 1$ , we have

$$f_k = 4^{-k} \binom{2k}{k} \in \left[ \frac{1}{\sqrt{\pi(k+1)}}, \frac{1}{\sqrt{\pi k}} \right]. \quad (2.10)$$

For all integers  $n \geq 0$ , we have

$$\sum_{k=0}^n f_k f_{n-k} = 1 \quad (2.11)$$

and

$$\frac{\gamma + \log(n) - 1}{\pi} \leq \sum_{k=1}^{\infty} f_k^2 \leq \frac{\gamma + \log(n)}{\pi}, \quad (2.12)$$

where  $0.57721 \leq \gamma \leq 0.57722$  is the Euler-Mascheroni constant. Furthermore, for all  $x \in \mathbb{C}$  with  $|x| < 1$ , we have

$$\sum_{k=0}^{\infty} f_k x^k = \frac{1}{\sqrt{1-x}}. \quad (2.13)$$

*Proof.* The first part of Equation 2.10 can be shown by induction: For  $k \geq 1$ ,

$$4^{-k} \binom{2k}{k} = \frac{1}{4} \cdot 4^{-(k-1)} \binom{2k-2}{k-1} \frac{2k(2k-1)}{k^2} = \frac{2k(2k-1)}{4k^2} f_{k-1} = \left(1 - \frac{1}{2k}\right) f_{k-1} = f_k.$$

Next consider the derivatives of  $f(x) := \frac{1}{\sqrt{1-x}}$ : For  $k \geq 1$ , we have

$$f^{(k)}(x) = (1-x)^{-k-1/2} \prod_{\ell=0}^{k-1} \left( \ell + \frac{1}{2} \right). \quad (2.14)$$

Thus

$$\frac{f^{(k)}(0)}{k!} = \prod_{\ell=0}^{k-1} \frac{1}{\ell+1} \left( \ell + \frac{1}{2} \right) = \prod_{\ell=0}^{k-1} \frac{(2\ell+1)(2\ell+2)}{2(\ell+1)2(\ell+1)} = \frac{(2k)!}{(k!)^2 2^{2k}} = 4^{-k} \binom{2k}{k} = f_k.$$

By Taylor's theorem, for all  $x \in \mathbb{C}$  with  $|x| < 1$ ,

$$\frac{1}{\sqrt{1-x}} = f(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(0)}{k!} x^k = \sum_{k=0}^{\infty} f_k x^k,$$

as required to prove Equation 2.13. Furthermore, for all  $x \in \mathbb{C}$  with  $|x| < 1$ , we have

$$\sum_{n=0}^{\infty} x^n = \frac{1}{1-x} = f(x)^2 = \left( \sum_{k=0}^{\infty} f_k x^k \right)^2 = \sum_{n=0}^{\infty} x^n \sum_{k=0}^n f_k f_{n-k}.$$

Matching coefficients proves Equation 2.11

To prove the second part of Equation 2.10, we use standard bounds on the central binomial coefficient [elk; spe]: For all integers  $k \geq 1$ ,

$$\frac{4^k}{\sqrt{\pi(k+1)}} \leq \binom{2k}{k} \leq \frac{4^k}{\sqrt{\pi k}}, \quad (2.15)$$

whence  $\frac{1}{\sqrt{\pi(k+1)}} \leq f_k \leq \frac{1}{\sqrt{\pi k}}$ . Next we use standard bounds on the harmonic numbers to prove Equation 2.12: For all integers  $n \geq 1$ ,

$$\log(n) + \gamma + \frac{1}{2(n+1)} \leq \sum_{k=1}^n \frac{1}{k} \leq \log(n) + \gamma + \frac{1}{2n} \quad (2.16)$$

where  $0.57721 \leq \gamma \leq 0.57722$  is the Euler-Mascheroni constant [SW]. It follows that, for all  $n \geq 2$ ,

$$\sum_{k=1}^{n-1} f_k^2 \leq \sum_{k=1}^{n-1} \frac{1}{\pi k} \leq \frac{\gamma + \log(n-1) + 1/2(n-1)}{\pi} \leq \frac{\gamma + \log(n)}{\pi}. \quad (2.17)$$

Similarly, for all  $n \geq 2$

$$\sum_{k=1}^{n-1} f_k^2 \geq \sum_{k=1}^{n-1} \frac{1}{\pi(k+1)} = \sum_{k=2}^n \frac{1}{\pi k} \geq \frac{\gamma + \log(n) + 1/(2n+2) - 1}{\pi}. \quad (2.18)$$

□

Most importantly, we show that the factorization of Fichtenberger, Henzinger, and Upadhyay [FHU22] is optimal among the class of lower triangular Toeplitz factorizations:

**Proposition 2.2** (Optimal lower triangular Toeplitz factorization). *For any integer  $n \geq 1$ , consider the optimization problem*

$$\begin{aligned} \min_{b, c \in \mathbb{R}^n} \quad & \|b\|_2 \|c\|_2 \\ \text{s.t.} \quad & \forall k < n \quad \sum_{i=0}^k b_i c_{k-i} = 1. \end{aligned} \quad (2.19)$$

The minimum is achieved at  $b = c = (f_k)_{k=0}^{n-1}$ , where  $f_0, f_1, \dots$  is given by  $f_0 = 1$  and  $f_k = (1 - \frac{1}{2k}) f_{k-1}$  for all  $k \geq 1$ , which are the coefficients derived by Fichtenberger, Henzinger, and Upadhyay [FHU22].

*Proof.* The proof proceeds in 3 steps: we first reformulate the optimization as a quadratically constrained quadratic program, then derive a natural Lagrangian relaxation of it, and finally construct a primal dual solution to the Lagrangian relaxation that is globally optimal with the primal solution  $b = c = (f_k)_{k=0}^{n-1}$ .

We use the following properties of  $f_k$ 's established by Fichtenberger, Henzinger, and Upadhyay [FHU22] and also shown in Lemma 2.1:

- $f_k > 0 \quad \forall k \in \{0, \dots, n-1\}$ .
- Let  $F \in \mathbb{R}^{n \times n}$  denote the lower triangular Toeplitz matrix whose first column is  $f_0, f_1, \dots, f_{n-1}$ . Then, we have  $F^2 = A$ .
- $\sum_{k=0}^{\infty} f_k x_k = \frac{1}{\sqrt{1-x}} \quad \forall x \in [0, 1]$ .

**Reformulation as a quadratically constrained quadratic program:** We begin by observing that the optimization problem can be rewritten as a quadratic optimization as follows:

$$\|b\|_2 \|c\|_2 = \inf \left\{ \frac{\nu}{2} \|b\|_2^2 + \frac{1}{2\nu} \|c\|_2^2 : \nu > 0 \right\} = \min \left\{ \frac{1}{2} \|\nu b\|_2^2 + \frac{1}{2} \left\| \frac{c}{\nu} \right\|_2^2 : \nu \in \mathbb{R} \setminus \{0\} \right\}.$$

Furthermore, for any feasible  $(b, c)$  that satisfy the constraints of Equation 2.19 and any  $\nu \neq 0$ , the pair  $(\nu b, \frac{c}{\nu})$  also satisfies the constraints. If  $S$  denotes the feasible set of Equation 2.19, we have

$$\min_{b, c \in S} \|b\|_2 \|c\|_2 = \min_{b, c \in S} \min_{\nu \in \mathbb{R} \setminus \{0\}} \frac{1}{2} \|\nu b\|_2^2 + \frac{1}{2} \left\| \frac{c}{\nu} \right\|_2^2 = \min_{b, c \in S} \frac{1}{2} \|b\|_2^2 + \frac{1}{2} \|c\|_2^2.$$

Thus, we can simply write the overall optimization problem as

$$\begin{aligned} \min_{b, c \in \mathbb{R}^n} \quad & \frac{1}{2} (\|b\|_2^2 + \|c\|_2^2) \\ \text{s.t.} \quad & \forall k < n \quad \sum_{i=0}^k b_i c_{k-i} = 1. \end{aligned} \tag{2.20}$$

**Lagrangian relaxation:** We write the Lagrangian relaxation of this optimization problem introducing dual variables  $\lambda_0, \dots, \lambda_{n-1} \in \mathbb{R}$  corresponding to the constraints:

$$\begin{aligned} L(b, c, \lambda) &= \frac{1}{2} (\|b\|_2^2 + \|c\|_2^2) - \sum_{k=0}^{n-1} \lambda_k \left( \sum_{i=0}^k b_i c_{k-i} - 1 \right) \\ &= \frac{1}{2} (\|b\|_2^2 + \|c\|_2^2) - b^\top \Gamma(\lambda) c + \mathbf{1}^\top \lambda \\ &= \frac{1}{2} \begin{pmatrix} b \\ c \end{pmatrix}^\top \begin{pmatrix} I & -\Gamma(\lambda) \\ -\Gamma(\lambda) & I \end{pmatrix} \begin{pmatrix} b \\ c \end{pmatrix} + \mathbf{1}^\top \lambda, \end{aligned} \tag{2.21}$$

where  $\Gamma(\lambda)$  is the symmetric Hankel matrix

$$\Gamma(\lambda) := \begin{pmatrix} \lambda_0 & \lambda_1 & \lambda_2 & \dots & \lambda_{n-1} \\ \lambda_1 & \lambda_2 & \lambda_3 & \dots & 0 \\ \lambda_2 & \lambda_3 & \lambda_4 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \lambda_{n-1} & 0 & 0 & \dots & 0 \end{pmatrix}.$$

The constrained optimization problem in Equation 2.20 is *equivalent* to the unconstrained min-max problem

$$\min_{b, c \in \mathbb{R}^n} \max_{\lambda \in \mathbb{R}^n} L(b, c, \lambda). \tag{2.22}$$

Letting  $\vec{f} = (f_0, \dots, f_{n-1}) \in \mathbb{R}^n$ , we will set  $b = c = \vec{f}$  and exhibit a setting  $\lambda = \lambda^*$  such that  $\nabla_{b, c, \lambda} L(b, c, \lambda) = 0$  and  $L(b, c, \lambda)$  is jointly convex in  $b, c$  and concave in  $\lambda$ . It follows that  $b = c = \vec{f}$  is the optimal solution to the original problem, as required. Since  $b = c = \vec{f}$  is feasible,  $\nabla_\lambda L(b, c, \lambda) = \nabla_\lambda L(\vec{f}, \vec{f}, \lambda) = 0$ . The Lagrangian is linear in  $\lambda$  and hence trivially concave. The Lagrangian is convex in  $(b, c)$  if and only if

$$\begin{pmatrix} I & -\Gamma(\lambda) \\ -\Gamma(\lambda) & I \end{pmatrix} \succeq 0$$

or equivalently (by Schur complements) if  $\|\Gamma(\lambda)\|_* \leq 1$  where  $\|Q\|_* := \sup\{\|Qx\|_2 : \|x\|_2 \leq 1\}$  denotes the operator norm.

**Construction of primal-dual optimal solution:** Now it only remains to compute  $\lambda = \lambda^*$  such that  $\nabla_{b,c}L(b, c, \lambda) = 0$  for  $b = c = \vec{f}$  and  $\|\Gamma(\lambda)^2\|_* \leq 1$ .

For all  $k \in [n]$ , we have  $\frac{\partial}{\partial b_k}L(b, c, \lambda) = b_k - \sum_{i=k}^{n-1} \lambda_i c_{i-k}$  and  $\frac{\partial}{\partial c_k}L(b, c, \lambda) = c_k - \sum_{i=k}^{n-1} \lambda_i b_{i-k}$ . Equivalently,  $\nabla_b L(b, c, \lambda) = b - \Gamma(\lambda)c$  and  $\nabla_c L(b, c, \lambda) = c - \Gamma(\lambda)b$ . Thus, in order to ensure  $\nabla_{b,c}L(b, c, \lambda) = 0$  when  $b = c = \vec{f}$ , it suffices to set  $\lambda = \lambda^*$ , where  $\lambda^*$  is chosen so as to solve

$$\begin{aligned} \vec{f} = \Gamma(\lambda^*) \vec{f} &\iff P\vec{f} = P\Gamma(\lambda^*) \vec{f} \\ &\iff \begin{pmatrix} f_{n-1} \\ f_{n-2} \\ \vdots \\ f_0 \end{pmatrix} = \begin{pmatrix} \lambda_{n-1}^* & 0 & 0 & \cdots & 0 \\ \lambda_{n-2}^* & \lambda_{n-1}^* & 0 & \cdots & 0 \\ \lambda_{n-3}^* & \lambda_{n-2}^* & \lambda_{n-1}^* & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \lambda_0^* & \lambda_1^* & \lambda_2^* & \cdots & \lambda_{n-1}^* \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{n-1} \end{pmatrix} \\ &\iff \begin{pmatrix} f_{n-1} \\ f_{n-2} \\ \vdots \\ f_0 \end{pmatrix} = \begin{pmatrix} f_0 & 0 & 0 & \cdots & 0 \\ f_1 & f_0 & 0 & \cdots & 0 \\ f_2 & f_1 & f_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_{n-1} & f_{n-2} & f_{n-3} & \cdots & f_0 \end{pmatrix} \begin{pmatrix} \lambda_{n-1}^* \\ \lambda_{n-2}^* \\ \vdots \\ \lambda_0^* \end{pmatrix}, \end{aligned}$$

where  $P$  is the permutation matrix such that  $Px$  is the reversal of the vector  $x$ . Thus, we can solve for  $\lambda^*$  to obtain

$$P\lambda^* = \begin{pmatrix} \lambda_{n-1}^* \\ \lambda_{n-2}^* \\ \vdots \\ \lambda_0^* \end{pmatrix} = \begin{pmatrix} f_0 & 0 & 0 & \cdots & 0 \\ f_1 - f_0 & f_0 & 0 & \cdots & 0 \\ f_2 - f_1 & f_1 - f_0 & f_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_{n-1} - f_{n-2} & f_{n-2} - f_{n-3} & f_{n-3} - f_{n-4} & \cdots & f_0 \end{pmatrix} \begin{pmatrix} f_{n-1} \\ f_{n-2} \\ \vdots \\ f_0 \end{pmatrix},$$

where we used the fact that  $\vec{f}$  is a feasible solution:

$$\begin{aligned}
& \begin{pmatrix} f_0 & 0 & 0 & \dots & 0 \\ f_1 & f_0 & 0 & \dots & 0 \\ f_2 & f_1 & f_0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_{n-1} & f_{n-2} & f_{n-3} & \dots & f_0 \end{pmatrix} \begin{pmatrix} f_0 & 0 & 0 & \dots & 0 \\ f_1 & f_0 & 0 & \dots & 0 \\ f_2 & f_1 & f_0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_{n-1} & f_{n-2} & f_{n-3} & \dots & f_0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ 1 & 1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{pmatrix} \\
& \iff \begin{pmatrix} f_0 & 0 & 0 & \dots & 0 \\ f_1 & f_0 & 0 & \dots & 0 \\ f_2 & f_1 & f_0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_{n-1} & f_{n-2} & f_{n-3} & \dots & f_0 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ 1 & 1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{pmatrix}^{-1} \begin{pmatrix} f_0 & 0 & 0 & \dots & 0 \\ f_1 & f_0 & 0 & \dots & 0 \\ f_2 & f_1 & f_0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_{n-1} & f_{n-2} & f_{n-3} & \dots & f_0 \end{pmatrix} \\
& \iff \begin{pmatrix} f_0 & 0 & 0 & \dots & 0 \\ f_1 & f_0 & 0 & \dots & 0 \\ f_2 & f_1 & f_0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_{n-1} & f_{n-2} & f_{n-3} & \dots & f_0 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} f_0 & 0 & 0 & \dots & 0 \\ f_1 & f_0 & 0 & \dots & 0 \\ f_2 & f_1 & f_0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_{n-1} & f_{n-2} & f_{n-3} & \dots & f_0 \end{pmatrix}.
\end{aligned}$$

From the definition  $f_0 = 1$  and  $f_k = f_{k-1}(1 - 1/2k)$ , for all  $k \geq 1$ , we have

$$\begin{aligned}
\lambda_{n-1-i}^* &= f_0 f_{n-1-i} - \sum_{j=1}^i (f_j - f_{j-1}) f_{n-1-i+j} \\
&= f_{n-1-i} - \sum_{j=1}^i f_{j-1} \left( \frac{1}{2j} \right) f_{n-1-i+j} \\
&= f_{n-1-i} \left( 1 - \sum_{j=1}^i f_{j-1} \left( \frac{1}{2j} \right) \frac{f_{n-1-i+j}}{f_{n-1-i}} \right).
\end{aligned}$$

Since  $\frac{f_{n-1-i+j}}{f_{n-1-i}} = \prod_{\ell=1}^j \left( 1 - \frac{1}{2(n-i+\ell)} \right) \leq 1$ , we have

$$\lambda_{n-1-i}^* \geq f_{n-1-i} \left( 1 - \sum_{j=1}^i f_{j-1} \left( \frac{1}{2j} \right) \right) > f_{n-1-i} \left( 1 - \sum_{j=1}^{\infty} f_{j-1} \left( \frac{1}{2j} \right) \right) = f_{n-1-i} \left( 1 - \sum_{j=0}^{\infty} \frac{f_j}{2(j+1)} \right). \quad (2.23)$$

Since  $\sum_{j=0}^{\infty} f_j x^j = \frac{1}{\sqrt{1-x}}$  for all  $x \in [0, 1)$ , integrating both sides between the limits 0 and  $t$ , we have

$$\sum_{j=0}^{\infty} \frac{f_j}{j+1} t^{j+1} = 2(1 - \sqrt{1-t}).$$

Taking limits as  $t \rightarrow 1$ , we obtain

$$\sum_{j=0}^{\infty} \frac{f_j}{j+1} = 2.$$



Thus, from Equation 2.23, we can conclude that  $\lambda_{n-1-i}^* > 0$  for all  $i \in [n]$  and hence  $\Gamma(\lambda^*)^2$  is a matrix with all entries  $(\Gamma(\lambda^*)^2)_{i,j} = \sum_{k=0}^{n-1-\max\{i,j\}} \lambda_{i+k}^* \lambda_{j+k}^*$  strictly positive. Hence, by the Perron-Frobenius theorem, we have that  $\Gamma(\lambda^*)^2$  has a unique eigenvector with all coordinates positive, and this corresponds to the maximum eigenvalue. Since  $f$  is an eigenvector of  $\Gamma(\lambda^*)^2$  by construction, it must be the unique leading eigenvector corresponding to the eigenvalue 1. Hence  $\|\Gamma(\lambda^*)^2\|_* \leq 1$ .

Thus,  $L(b, c, \lambda^*)$  is convex in  $(b, c)$ , since  $\|\Gamma(\lambda)^2\|_* \leq 1 \iff \|\Gamma(\lambda)\|_* \leq 1$ . And  $\lambda^*$  was chosen to ensure  $\nabla_{b,c} L(b, c, \lambda^*) = 0$ . This lets us conclude that  $(\vec{f}, \vec{f}, \lambda^*)$  is a stationary point of the Lagrangian that is convex in  $b, c$  and concave in  $\lambda$ , hence it is the global optimum.  $\square$

Proposition 2.2 fully characterizes the optimal lower triangular Toeplitz factorization. We now compare the value of the matrix factorization objective for lower triangular Toeplitz factorizations versus general factorizations. For convenience, we define a notation for these values:

**Definition 2.3.** Let  $A \in \{0, 1\}^{n \times n}$  be the lower triangular matrix of all ones. Recall that  $C \in \mathbb{R}^{n \times n}$  is lower triangular Toeplitz if  $C_{i,j} = 0$  for all  $i < j$  and  $C_{i,j} = c_{i-j}$  for all  $i \geq j$ , where  $c \in \mathbb{R}^n$  is a vector. Define

$$\text{OptLTToe}(n) := \inf \{ \text{MaxErr}(B, C) : B, C \in \mathbb{R}^{n \times n} \text{ lower triangular Toeplitz}, BC = A \}, \quad (2.24)$$

$$\text{Opt}(n) := \inf \{ \text{MaxErr}(B, C) : B, C \in \mathbb{R}^{n \times n}, BC = A \}, \quad (2.25)$$

where  $\text{MaxErr}(B, C) := \|B\|_{2 \rightarrow \infty} \|C\|_{1 \rightarrow 2} = \sqrt{\max_i \sum_j B_{i,j}^2} \sqrt{\max_j \sum_i C_{i,j}^2}$ .

By Proposition 2.2 and Lemma 2.1, for all integers  $n \geq 1$ , we have

$$1 + \frac{\gamma - 1 + \log(n)}{\pi} \leq \text{OptLTToe}(n) = 1 + \sum_{k=1}^{n-1} \left( 2^{-2k} \binom{2k}{k} \right)^2 \leq 1 + \frac{\gamma + \log(n)}{\pi}. \quad (2.26)$$

The lower bound of Matoušek, Nikolov, and Talwar [MNT20] in Equation 2.9 states the optimal over all factorizations is

$$\text{Opt}(n) = \gamma_2(A^{(n)}) \geq 1 + \frac{\log(2n+1)}{\pi}.$$

Combining these bounds gives a bound on the gap between the optimal general factorization and the optimal lower triangular Toeplitz factorization.

**Corollary 2.4.** The gap between the lower triangular Toeplitz and general optimal factorizations is

$$\text{OptLTToe}(n) - \text{Opt}(n) \leq \left( 1 + \frac{\gamma + \log(n)}{\pi} \right) - \left( 1 + \frac{\log(2n+1)}{\pi} \right) \leq \frac{\gamma + \log(2)}{\pi} \leq 0.405.$$

Numerically, we have  $\text{OptLTToe}(n) - \text{Opt}(n) \leq 0.365$  for all  $n \geq 1$ . Figure 2 shows numerically how the various known upper and lower bounds compare for differing values of  $n$ .

We conclude this subsection by remarking that, without the Toeplitz constraint, we can assume that the factors are lower triangular without loss of generality. This is a consequence of the fact that two factorizations  $A = BC$  and  $A = \hat{B}\hat{C}$  are functionally equivalent as long as  $BB^T = \hat{B}\hat{B}^T$ . To see why this fact is true, note that for  $z \leftarrow \mathcal{N}(0, \sigma^2 I)$ , the distribution of  $Bz$  is  $\mathcal{N}(0, \sigma^2 BB^T)$ . So if  $BB^T = \hat{B}\hat{B}^T$ , then  $Bz$  and  $\hat{B}z$  have the exact same distribution. Note that this fact is specific to the Gaussian distribution. Given this fact, we can take an arbitrary factorization  $A = BC$  and perform a Cholesky decomposition of  $BB^T$  to obtain a lower triangular  $\hat{B}$  satisfying  $BB^T = \hat{B}\hat{B}^T$ . However, if we restrict to Toeplitz matrices, then we cannot assume they are lower triangular without loss of generality, since the Cholesky decomposition may not preserve this structure.

## 2.3 Applications of Matrix Factorizations in Machine Learning

In the recent years, Matrix Factorization (MF) have found extensive use both in obtaining optimal private regret/stochastic convex optimization (SCO) guarantees [ST13; AS17; KMSTTX21; AFKT23; ZTC22], and in practical deployments [KMSTTX21; DMRST22; CCMRT22; CCGMMRGTX24; XZACCKMRZ23]. The main observation that maps MF to the (convex) optimization setup is that (unconstrained) stochastic gradient descent update with constant learning rate can be viewed as computing prefix sums over adaptively chosen gradient vectors<sup>16</sup>. Differentially Private Follow-the-regularized-leader (DP-FTRL) [KMSTTX21; ST13] is an optimization algorithm that arises naturally from this mapping. DP-FTRL has found extensive usage in production grade deployment of DP learning. For example, all next-word-prediction models on Android Gboard is trained with DP-FTRL [XZACCKMRZ23].

For a given set of  $\ell_2$ -Lipschitz loss functions  $\{f_1(\theta), \dots, f_n(\theta)\}$  (where each loss function  $f_i$  can be thought of as the optimization loss on a disjoint minibatch of data samples), the simplest version of DP-FTRL can be described as follows:

$$\text{DP-FTRL: } \theta_t \leftarrow \theta_0 - \eta \left( \sum_{k=0}^{t-1} \nabla f_k(\theta_k) + \hat{z}_k \right). \quad (2.27)$$

In Equation 2.27,  $\eta$  is the learning rate, and  $\{\hat{z}_t\}$  are the correlated Gaussian noise added to ensure that the computation of the sequence of  $\{\theta_t\}$ 's are differentially private. Mapping to the MF view in Equation 1.1, the adaptively chosen data set is set to  $x = \{\nabla f_t(\theta_t) : t \in [n]\}$ , and the noise sequence  $\{\hat{z}_t\}$  is set to  $Bz$ , where  $z \leftarrow \mathcal{N}(0, \sigma^2 I)$ . DP-FTRL captures a large class of noise addition mechanisms (parameterized by the specific factorization scheme  $A = BC$  chosen). In particular, it captures DP-SGD [SCS13; BST14; ACGMMTZ16] (another heavily used private learning algorithm) as a special case [CCGMMRGTX24], and has much better privacy/utility trade-offs in general [CCDPGST24]. [KMSTTX21; AFKT23] showed that for obtaining the best regret/SCO guarantees from DP-FTRL style algorithms, it suffices to optimize for  $\text{MaxErr}(B, C)$  to instantiate the noise mechanism in Equation 2.27.

One major challenge in large deployments of DP-FTRL is that noise generation for an arbitrary matrix mechanism can be prohibitively expensive, as discussed in Section 1. Our work provides

<sup>16</sup>There are more general mappings for constrained optimization, and adaptive optimizers like momentum methods, but a detailed discussion on those is tangential to this work.

a practical approach that only requires a constant factor blow-up (usually at most 4-6 times) in memory as compared to that of DP-SGD, and at privacy/utility trade-off comparable to DP-FTRL with an optimal matrix-factorization mechanism. (DP-SGD adds independent Gaussian noise to the gradients at each state update.)

## 2.4 Other Related Work

**Beyond additive noise:** Our algorithms are based on adding Gaussian noise (with a carefully chosen covariance). Since the noise is additive, the error is independent of the number of data points.<sup>17</sup> The expected maximum error is

$$\forall x \quad \mathbb{E}_{\mathcal{M}} [\|\mathcal{M}(x) - Ax\|_{\infty}] = \mathbb{E}_{\mathcal{M}} \left[ \max_k |(\mathcal{M}(x) - Ax)_k| \right] \leq O(\log(n))^{3/2}, \quad (2.28)$$

while the maximum expected error is

$$\forall x \quad \max_k \mathbb{E}_{\mathcal{M}} [ |(\mathcal{M}(x) - Ax)_k| ] \leq O(\log(n)). \quad (2.29)$$

The additional  $\sqrt{\log(n)}$  term in Equation 2.28 compared to Equation 2.29 comes from taking a union bound over  $n$  Gaussians. Note that, for simplicity, we suppress the privacy parameters; to achieve  $(\varepsilon, \delta)$ -differential privacy, the error above scales with  $\frac{1}{\varepsilon} \log(1/\delta)$ ; to achieve  $\rho$ -zCDP [BS16], the error above scales with  $1/\sqrt{\rho}$ .

The bound in Equation 2.29 on the maximum expected error is tight up to constants in the streaming setting – in fact the lower bound holds for *average* expected error:

$$\forall \mathcal{M} \exists x \quad \frac{1}{n} \sum_k \mathbb{E}_{\mathcal{M}} [ |(\mathcal{M}(x) - Ax)_k| ] \geq \Omega(\log(n)), \quad (2.30)$$

assuming  $\mathcal{M}$  is differentially private [HUU23, Theorem 4]. However, the bound in Equation 2.28 on expected maximum error is *not* tight if we move beyond additive noise mechanisms [DNRR15]. Dwork, Naor, Reingold, and Rothblum [DNRR15] show that we can obtain an improved error guarantee:<sup>18</sup>

$$\exists \mathcal{M} \forall x \quad \mathbb{E}_{\mathcal{M}} [\|\mathcal{M}(x) - Ax\|_{\infty}] \leq O(\log(n) + \log(|x|)^{3/2}), \quad (2.31)$$

where  $|x|$  is the number of data points.<sup>19</sup> The high level idea is to “freeze” the output of  $\mathcal{M}$  until the partial sum has changed significantly and only then update the output. This works because deciding whether the partial sum has changed significantly is easier than evaluating it. Specifically, we can use the so-called “sparse vector” technique to decide when to update. This algorithm is a reduction – when it does decide to update, it uses additive noise – the “win” is that we need to use additive noise for  $\leq |x|$  updates, rather than  $n$  updates. Thus our improvements for additive noise mechanisms can be applied in this setting.

Equation 2.31 is only an improvement over Equation 2.28 in the sparse setting ( $|x| \ll n$ ). Machine learning applications are typically dense ( $|x| \geq n$ ).

<sup>17</sup>The error of our algorithms depends on the number of iterations  $n$ . This is usually closely related to the number of data points  $|x|$ , but, in general, these two quantities can be unrelated.

<sup>18</sup>Dwork, Naor, Reingold, and Rothblum [DNRR15] stated this result for pure differential privacy, with  $\log(|x|)^2$  instead of  $\log(|x|)^{3/2}$ .

<sup>19</sup>In most of the differential privacy literature  $n$  is the number of data points / people, but we use  $n$  for the number of iterations;  $|x|$  should be read as the size of the dataset  $x$ .

**Offline setting:** Our work is in the streaming setting, where we must output an approximation to each partial sum as soon as it is complete. It is also natural to consider the offline setting, where we are given the entire input at once. In the offline setting, the memory constraints that we focus on are not as relevant.

The offline version of continual counting is known as threshold queries or quantiles and this task has been studied extensively [BNS13; BNSV15; BDRS18; ALMM19; KLMNS20; GJK21; CLNSS23]. The offline setting permits more sophisticated algorithms, which achieve asymptotically better error guarantees. Cohen, Lyu, Nelson, Sarlós, and Stemmer [CLNSS23] give an  $(\varepsilon, \delta)$ -differentially private algorithm  $\mathcal{M}$  with error

$$\forall x \quad \mathbb{E}_{\mathcal{M}} [\|\mathcal{M}(x) - Ax\|_{\infty}] \leq \tilde{O} \left( \frac{1}{\varepsilon} \log^2(1/\delta) \log^*(n) \right) \quad (2.32)$$

Note that offline algorithms are not applicable in the machine learning applications discussed in Section 2.3.

**Pure differential privacy:** The early work on continual counting [DNPR10; CSS11; XWG10] works under pure  $(\varepsilon, 0)$ -differential privacy, which rules out Gaussian noise. Instead Laplace noise can be used. In this case, we must bound the  $L_1$  sensitivity. So the objective for the matrix factorization is to minimize  $\|B\|_{2 \rightarrow \infty} \cdot \|C\|_{1 \rightarrow 1}$ , where  $\|C\|_{1 \rightarrow 1} = \max_j \sum_i |C_{i,j}|$  is the maximum 1-norm of a column. This subtly changes the problem. The binary tree factorization still works well with this objective. However, Toeplitz factorizations do not work as well. Intuitively, the binary tree factorization relies on sparsity which controls both the 1- and 2-norms, whereas Toeplitz factorizations produce dense matrices for which the 1- and 2-norms differ substantially.

**Factoring other matrices:** We exclusively look at factorizing the all-ones lower triangular matrix. This is a natural and well-motivated problem. However, there is also work on factorizing other matrices. Most closely related to the all-ones lower triangular matrix  $A$ , Mathias [Mat93] gives exact expressions for  $\gamma_2(A + A^T)$  and  $\gamma_2(A + A^T + I)$ . There is a different line of work [LMHMR15; MMHM21] considering factorizations of generic workload matrices, which have a completely different structure than the matrices we consider.

## Notation Summary

Before proceeding, we briefly summarize the key symbols and notation used throughout:

---

$n$	Number of steps on which DP estimates are released.
$d$	Order of recurrence / degree / number of memory buffers.
$m$	Dimension of per-step user contributions (e.g., model size).
$[k]$	$= \{0, \dots, k-1\}$ for $k \geq 1$ .
$\mathbb{N}$	$= \{0, 1, 2, \dots\}$ , the natural numbers including zero.
$f$	Rational (generating) function where $f(x) = f_0 + f_1x + f_2x^2 \dots$ .
$p$	Polynomial of degree $d$ with $p(x) = p_0 + p_1x + p_2x^2 + \dots + p_dx^d$ .
$(f_k)_{k=0}^\infty$	Sequence $f_0, f_1, \dots$ with corresponding generating function $f$ .
$M(f, n) \in \mathbb{R}^{n \times n}$	Lower-triangular Toeplitz matrices defined by a generating function $f$ .
$\cdot$	Context-dependent multiplication, used where it improves readability.
$f * r$	Cauchy product / convolution $(f * r)_n = \sum_{i=0}^n f_i \cdot r_{n-i}$ .
$A^T$	Transpose of a matrix $A$ .
$A^\star$	A matrix $A$ that is “optimal” in a context-dependent sense.
$A^\dagger$	Moore-Penrose pseudoinverse of matrix $A$ .
$A_{i,j}$	The $(i, j)^{\text{th}}$ entry of matrix $A$ , zero-indexed.
$A_{i,:}$ and $A_{:,j}$	The $i^{\text{th}}$ row and $j^{\text{th}}$ column.
$\ A\ _F$	The Frobenius norm of a matrix $A$ .
$A \in \mathbb{R}^{n \times n}$	Specifically, the lower-triangular matrix of ones to be factorized as $A = BC$ .
$\log$	Natural logarithm, $\log(2.718) \approx 1$ .

---

Table 1: Summary of notation

### 3 Efficiently-Sampleable Factorization via Rational Generating Functions

In this section we describe our main algorithmic tools. First (in §3.1) we present the view of lower triangular Toeplitz matrices in terms of generating functions or sequences. This is a convenient mathematical formalism for analysis [FHU22]. Second (in §3.2) we discuss the special case of rational generating functions or, equivalently, constant-recurrent sequences. Our results focus on this special case, as the additional structure is useful for our algorithms; this structure is where we depart from the prior literature. Third (in §3.3) we present our algorithm for sampling from lower triangular Toeplitz matrices with rational generating functions (that is, the multiplication algorithm for BLT matrices). In subsequent sections (§4, §5) we instantiate these rational generating functions.

### 3.1 Sequences, Lower Triangular Toeplitz Matrices, & Generating Functions

We begin by describing the generating function view of lower triangular Toeplitz matrices used by Fichtenberger, Henzinger, and Upadhyay [FHU22]. Our analysis moves fluidly between three different views of the same mathematical object:

- The sequence  $(f_k)_{k=0}^\infty = f_0, f_1, f_2, \dots$  with  $f_k \in \mathbb{R}$  for each index  $k$ .
- The ordinary generating function of the sequence,  $f : \mathbb{C} \rightarrow \mathbb{C}$ ,

$$f(x) = \sum_{k=0}^{\infty} f_k x^k = \sum_{k=0}^{\infty} \frac{f^{(k)}(0)}{k!} x^k.$$

(It will be clear from context if  $f$  refers to the sequence entries or the generating function).

- The family of  $n \times n$  lower-triangular Toeplitz matrices  $M(f, n) \in \mathbb{R}^{n \times n}$  for  $n \in \mathbb{N}$  that are generated by  $f$  as

$$\forall i, j \in [n] \quad M(f, n)_{i,j} := \begin{cases} f_{i-j} & \text{if } i \geq j \\ 0 & \text{if } i < j \end{cases} \quad (3.1)$$

When the specific  $n$  is unimportant or clear from context, we write  $M(f)$  to refer to any matrix in this family (or equivalently, the infinite-dimensional linear operator).

As a canonical example, we have the all-ones sequence,

$$(g_k)_{k=0}^\infty = 1, 1, 1, 1, \dots, \quad g(x) = \frac{1}{1-x} = \sum_{k=0}^{\infty} x^k, \quad A^{(4)} := M(g, 4) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad (3.2)$$

It is straightforward to see that addition (under the usual definitions) is equivalent across all three representations, and (slightly less obviously) the same fact holds for suitable definitions of multiplication. We denote the Cauchy product or convolution  $*$  for sequences  $f$  and  $g$  by

$$(f_k)_{k=0}^\infty * (g_k)_{k=0}^\infty = (h_k)_{k=0}^\infty \quad \text{where} \quad h_k := \sum_{i=0}^k f_i \cdot g_{k-i}, \quad (3.3)$$

and write  $h = f * g$  as a shorthand.

Then, the mathematical structure summarized in the following lemma is key to our analysis:

**Lemma 3.1.** *Let  $f$ ,  $g$ , and  $h$  be ordinary generating functions with corresponding sequences. Then the Cauchy product of sequences, multiplication of the generating functions, and matrix multiplication are all equivalent. That is,*

$$(h = f * g) \iff (h(x) = f(x)g(x)) \iff (\forall n \ M(h, n) = M(f, n) \cdot M(g, n)).$$

*Similarly the usual definitions of addition are equivalent:*

$$(\forall i \ h_i = f_i + g_i) \iff (h(x) = f(x) + g(x)) \iff (\forall n \ M(h, n) = M(f, n) + M(g, n)).$$

*Proof.* The results for sequences and their ordinary generating functions are well known, see e.g. Kauers and Paule [KP11] or their equivalent results based on well-known properties of the Z-transform; cf. [OWN97]. The results for addition are also immediate. Hence, it is sufficient to show  $h(x) = f(x)g(x)$  iff  $\forall n \in \mathbb{N}, M(h, n) = M(f, n) \cdot M(g, n)$ .

Fix  $i, j \in [n]$ . If  $i < j$ , then  $M(h, n)_{i,j} = 0$  and  $(M(f, n) \cdot M(g, n))_{i,j} = \sum_{k \in [n]} M(f, n)_{i,k} \cdot M(g, n)_{k,j} = \sum_k 0$ , since, for all  $k \in [n]$ , either  $i < k$ , whence  $M(f, n)_{i,k} = 0$ , or  $k < j$ , whence  $M(g, n)_{k,j} = 0$ , (or both). Thus we can assume  $i \geq j$ .

By the product rule and induction, we have the derivative  $h^{(\ell)}(x) = \sum_{k=0}^{\ell} \binom{\ell}{k} f^{(\ell-k)}(x) \cdot g^{(k)}(x)$  for all  $\ell \geq 0$  and all applicable  $x$ . Thus

$$\begin{aligned}
M(h, n)_{i,j} &= \frac{h^{(i-j)}(0)}{(i-j)!} \\
&= \frac{1}{(i-j)!} \sum_{k=0}^{i-j} \binom{i-j}{k} f^{(i-j-k)}(0) \cdot g^{(k)}(0) \\
&= \sum_{k=0}^{i-j} \frac{f^{(i-j-k)}(0)}{(i-j-k)!} \cdot \frac{g^{(k)}(0)}{k!} \\
&= \sum_{k'=j}^i \frac{f^{(i-k')}(0)}{(i-k')!} \cdot \frac{g^{(k'-j)}(0)}{(k'-j)!} \quad (k' = k + j) \\
&= \sum_{k'=j}^i M(f, n)_{i,k'} \cdot M(g, n)_{k',j} \\
&= \sum_{k' \in [n]} M(f, n)_{i,k'} \cdot M(g, n)_{k',j} \\
&\quad (i < k' \implies M(f, n)_{i,k'} = 0, k' < j \implies M(g, n)_{k',j} = 0) \\
&= (M(f, n) \cdot M(g, n))_{i,j}.
\end{aligned}$$

□

Given Equation 3.2 and the above result for multiplication, in order to factorize  $A = BC$ , it is natural to factorize by taking the square root of the generating function. Let  $f(x) = \frac{1}{\sqrt{1-x}}$ . Since  $f(x) \cdot f(x) = \frac{1}{1-x} = g(x)$  is the generating function of the all ones sequence, it follows that  $M(f, n) \cdot M(f, n) = M(g, n) = A^{(n)}$ . Indeed, this is the factorization given by Fichtenberger, Henzinger, and Upadhyay [FHU22], which we show to be optimal among the class of lower triangular Toeplitz matrices in Proposition 2.2.

Given any generating function  $r(x)$ , we can obtain a valid factorization  $A = BC$  where  $B = M(b, n)$  for  $b(x) = r(x)/(1-x)$  and  $C = M(c, n)$  for  $c(x) = 1/r(x)$ . Our approach is to choose a generating function such that  $r(x) \approx \sqrt{1-x}$  but which also permits us to design an efficient algorithm.



## 3.2 Rational Generating Functions and Constant-Recurrent Sequences

We will focus our attention on rational generating functions of the form

$$r(x) = \frac{p(x)}{q(x)} = \frac{\sum_{i=0}^{d-1} p_i x^i}{1 + \sum_{j=1}^d q_j x^j}. \quad (3.4)$$

These have convenient algorithmic properties. In this subsection, we discuss several equivalent representations of rational functions. As a starting point, a rational function of degree  $d$  can be represented as a ratio of polynomials, where the numerator  $p(x)$  has degree  $< d$  and the denominator has degree  $\leq d$ . Note that we normalize the denominator so that  $q(0) = q_0 = 1$ .<sup>20</sup>

Equation 3.4 is a convenient mathematical representation, but for algorithmic purposes we need the sequence  $(r_k)_{k=0}^{\infty}$  of Taylor series coefficients

$$r(x) = \sum_{k=0}^{\infty} r_k x^k. \quad (3.5)$$

Thus our first task is to map from Equation 3.4 to Equation 3.5 with a convenient representation that is amenable to efficient computation.

The generating function  $r(x)$  being a rational function is equivalent to the sequence  $(r_k)_{k=0}^{\infty}$  being a constant-recurrent sequence (a.k.a. linear-recursive or C-finite sequence); see for example Kauers and Paule [KP11]. The sequence terms can be computed by taking powers of a matrix; this is the representation that we will use in our algorithm. We summarize this result in Lemma 3.2 and provide a proof for completeness.

**Lemma 3.2** (Constant-recurrent Taylor series representation of a rational function). *Let*

$$r(x) = \frac{p(x)}{q(x)} = \frac{\sum_{i=0}^{d-1} p_i x^i}{1 + \sum_{j=1}^d q_j x^j}$$

*be a rational function of degree  $\leq d$ . As in Section 3.1, let*

$$r(x) = \sum_{k=0}^{\infty} r_k x^k.$$

*That is,  $r(x)$  generates the sequence  $(r_k)_{k=0}^{\infty}$ . Then, for all  $k \geq d$ , this sequence satisfies the recurrence*

$$r_k = - \sum_{j=1}^d q_j r_{k-j}. \quad (3.6)$$

*And, for  $0 \leq k < d$ , we have  $r_k = p_k - \sum_{j=1}^k q_j r_{k-j}$ . Furthermore, for all  $k$ , we have*

$$r_k = u^T W^k v, \quad (3.7)$$

---

<sup>20</sup>We can rescale both the numerator and denominator by a constant to make this assumption true. Thus this assumption is without loss of generality, unless the rational function has a pole at 0 – but we do not consider such functions, as they are not valid generating functions.

where

$$u := \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \in \mathbb{R}^{d \times 1}, \quad v := \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_{d-2} \\ p_{d-1} \end{pmatrix} \in \mathbb{R}^{d \times 1}, \quad (3.8)$$

and

$$W := \begin{pmatrix} -q_1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ -q_2 & 0 & 1 & 0 & \cdots & 0 & 0 \\ -q_3 & 0 & 0 & 1 & \cdots & 0 & 0 \\ -q_4 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -q_{d-1} & 0 & 0 & 0 & \cdots & 0 & 1 \\ -q_d & 0 & 0 & 0 & \cdots & 0 & 0 \end{pmatrix} \in \mathbb{R}^{d \times d}. \quad (3.9)$$

*Proof.* We have

$$r(x) = \frac{p(x)}{q(x)} = \frac{\sum_{i=0}^{d-1} p_i x^i}{1 + \sum_{j=1}^d q_j x^j} = \sum_{k=0}^{\infty} r_k x^k,$$

which rearranges to

$$p(x) = \sum_{i=0}^{d-1} p_i x^i = q(x) \cdot r(x) = \left(1 + \sum_{j=1}^d q_j x^j\right) \cdot \left(\sum_{k=0}^{\infty} r_k x^k\right) = \sum_{k=0}^{\infty} \left(r_k + \sum_{j=1}^{\min\{d,k\}} q_j r_{k-j}\right) \cdot x^k.$$

Matching coefficients gives  $p_k = r_k + \sum_{j=1}^k q_j r_{k-j}$  for  $1 \leq k \leq d-1$  and  $0 = r_k + \sum_{j=1}^d q_j r_{k-j}$  for  $k \geq d$ , which rearranges to give the recurrence.

Next we turn to the matrix representation. Define  $p^{(0)}(x) = p(x)$  — i.e.,  $p_i^{(0)} = p_i$  for  $0 \leq i \leq d-1$ . For all  $k \geq 1$ , let  $p_j^{(k)} := p_{j+1}^{(k-1)} - p_0^{(k-1)} q_{j+1}$  for  $0 \leq j \leq d-2$  and let  $p_{d-1}^{(k)} := -p_0^{(k-1)} q_d$ . We can check that these values satisfy

$$\frac{\sum_{i=0}^{d-1} p_i^{(k-1)} x^i}{1 + \sum_{j=1}^d q_j x^j} = p_0^{(k-1)} + x \cdot \frac{\sum_{i=0}^{d-1} p_i^{(k)} x^i}{1 + \sum_{j=1}^d q_j x^j}. \quad (3.10)$$

The left hand side of Equation 3.10 with  $k = 1$  is simply  $r(x)$ ; from this we can conclude that  $r_0 = p_0^{(0)} = p_0$ . Expanding Equation 3.10 for  $k = 1$  and  $k = 2$  gives  $r_1 = p_0^{(1)}$ . Specifically, we have

$$r(x) = \frac{\sum_{i=0}^{d-1} p_i^{(0)} x^i}{1 + \sum_{j=1}^d q_j x^j} = p_0^{(0)} + x \cdot \frac{\sum_{i=0}^{d-1} p_i^{(1)} x^i}{1 + \sum_{j=1}^d q_j x^j} = p_0^{(0)} + x \cdot \left( p_0^{(1)} + x \cdot \frac{\sum_{i=0}^{d-1} p_i^{(2)} x^i}{1 + \sum_{j=1}^d q_j x^j} \right).$$

We can iteratively repeat this expansion to extract the entire Taylor series. (This algorithm is known as polynomial long division.) That is, we can show by induction that, for all  $k$ , we have

$$r(x) = \sum_{\ell=0}^{k-1} p_0^{(\ell)} x^\ell + x^k \cdot \frac{\sum_{i=0}^{d-1} p_i^{(k)} x^i}{1 + \sum_{j=1}^d q_j x^j}.$$

Thus  $r_k = p_0^{(k)}$  for all  $k$ .

Now we can write the recursive definition of  $p_j^{(k)}$  in matrix form: For all  $k \geq 1$ , we have

$$\begin{pmatrix} p_0^{(k)} \\ p_1^{(k)} \\ \vdots \\ p_{d-1}^{(k)} \end{pmatrix} = W \cdot \begin{pmatrix} p_0^{(k-1)} \\ p_1^{(k-1)} \\ \vdots \\ p_{d-1}^{(k-1)} \end{pmatrix},$$

where  $W \in \mathbb{R}^{d \times d}$  is given in Equation 3.9. Given this, induction shows that, for all  $k \geq 0$ ,

$$\begin{pmatrix} p_0^{(k)} \\ p_1^{(k)} \\ \vdots \\ p_{d-1}^{(k)} \end{pmatrix} = W^k \cdot \begin{pmatrix} p_0^{(0)} \\ p_1^{(0)} \\ \vdots \\ p_{d-1}^{(0)} \end{pmatrix} = W^k \cdot v$$

and, hence,  $r_k = p_0^{(k)} = u^T \cdot W^k \cdot v$ , as required.  $\square$

We make some remarks about Lemma 3.2:

1. Lemma 3.2 assumes  $\deg(p) < d$  and  $\deg(q) \leq d$ , but it allows for the possibility that some of the coefficients  $p_j, q_j$  (including  $q_d$ ) may be zero. Thus by padding, we can accommodate arbitrary degree of both the numerator and denominator.
2. Lemma 3.2 is in fact an “if and only if” – that is, any Taylor series satisfying either of the conclusions of the Lemma must be a rational function (see e.g. [KP11]). However, we do not use the converse of the result.
3. The eigenvalues of  $W$  can be related to the poles of the corresponding rational function. Specifically,

$$\det(\lambda I - W) = 0 \iff \lambda^d + \sum_{j=1}^d q_j \lambda^{d-j} = 0 \iff \left(q\left(\frac{1}{\lambda}\right) = 0 \text{ or } \lambda = 0 = q_d\right). \quad (3.11)$$

4. Furthermore, the eigenvectors of  $W$  can be written in terms of the poles: Suppose  $q(\frac{1}{\lambda}) = 1 + \sum_{j=1}^d q_j (\frac{1}{\lambda})^j = 0$ . Let  $u^T = (\lambda^{d-1}, \lambda^{d-2}, \dots, \lambda, 1) \in \mathbb{R}^{1 \times d}$  (or  $\lambda = 0$  and  $q_d = 0$ ). Then  $u^T W = \lambda \cdot u^T$  or, equivalently,  $W^T u = \lambda \cdot u$ .
5. If the eigenvalues of  $W$  are distinct (and real), then the above remark produces a complete (real) eigenbasis, and so we can diagonalize the matrix  $W$  (over  $\mathbb{R}$ ). A diagonal matrix  $W$  is particularly convenient for our streaming algorithm (Algorithm 1).
6. If the eigenvalues of  $W$  are contained in the unit circle (i.e.,  $|\lambda| \leq 1$ ), then the computation of  $W^k$  is numerically stable. By Equation 3.11, this is equivalent to the poles of the rational function having magnitude  $\geq 1$ .

7. Suppose the rational function  $r$  has distinct poles  $\frac{1}{\theta_i}$ , and can be presented as<sup>21</sup>

$$r(x) = \sum_{i \in [d]} \frac{\omega_i}{1 - \theta_i \cdot x}$$

via a partial-fraction decomposition (instead of as a ratio of polynomials as in Lemma 3.2). (Indeed, our rational function in Section 4.2 is presented in this form in Equation 4.9.) Then, we have a simpler closed form for the sequence,

$$r_k = \sum_{i \in [d]} \omega_i \theta_i^k.$$

This immediately yields a matrix representation:  $W$  is a diagonal matrix with  $W_{i,i} = \theta_i$  for all  $i$ , and the vectors  $u$  and  $v$  simply need to satisfy  $u_i v_i = \omega_i$  for all  $i$ . This matrix representation may be more convenient than the one given by Lemma 3.2; we use this approach extensively in Section 5.

8. The form of the matrix  $W$  in Lemma 3.2 is known as a “companion matrix” [Wik24]. Namely, the matrix  $W$  is a companion to the denominator polynomial  $q(x)$  since the roots of the polynomial correspond to the eigenvalues of the matrix (per Equation 3.11).

Our choice in how to represent  $r(x)$  has some ramifications for the design of the noise generation (multiplication) algorithm in the next section.

**Remark 3.3.** *The representation in Equation 3.4 assumes the numerator has lower degree than the denominator, i.e.,  $\deg(p) < d$  versus  $\deg(q) \leq d$ . But, in this work, we typically consider rational functions with the numerator and denominator both having the same degree, i.e.,  $\deg(\bar{p}) = \deg(q) = d$ . This case can either be handled by padding (i.e., increment  $d \mapsto d + 1$  and set  $q_{d+1} = 0$ ) or by including an additive constant:*

$$r(x) = \frac{\bar{p}(x)}{q(x)} = t + \frac{p(x)}{q(x)} = \frac{p(x) + tq(x)}{q(x)}, \quad \text{where} \quad t = \frac{\bar{p}_d}{q_d} \quad \text{and} \quad p(x) = \bar{p}(x) - tq(x), \quad (3.12)$$

so that  $\deg(p) \leq d - 1$ . We then have

$$r_k = u^T W^k v + t \mathbb{I}[k = 0] \quad (3.13)$$

with  $(u, W, v)$  as given in Lemma 3.2 applied to  $p(x)/q(x)$ . Here  $\mathbb{I}$  is the indicator function taking value 1 if the condition holds and 0 otherwise. We prefer the representation with an additive constant, as it is more efficient to implement algorithmically.

We note the indicator function can also be written as  $\mathbb{I}[k = 0] = 0^k$  for  $k \geq 0$ . Thus we can incorporate the  $t \mathbb{I}[k = 0]$  into the matrix representation by appending a row/column of 0s to  $W$ , appending 1 to  $u$  and  $t$  to  $v$ :

$$\begin{pmatrix} u \\ 1 \end{pmatrix}^T \begin{pmatrix} W & \mathbf{0} \\ \mathbf{0}^T & 0 \end{pmatrix}^k \begin{pmatrix} v \\ t \end{pmatrix} = \begin{pmatrix} u \\ 1 \end{pmatrix}^T \begin{pmatrix} W^k & \mathbf{0} \\ \mathbf{0}^T & 0^k \end{pmatrix} \begin{pmatrix} v \\ t \end{pmatrix} = u^T W^k v + t \cdot \mathbb{I}[k = 0]. \quad (3.14)$$

This increases the dimension  $d$  by 1, which is equivalent to padding. However, for efficient streaming algorithm we introduce next, we can save one memory buffer by working directly with the  $r_k = u^T W^k v + t \mathbb{I}[k = 0]$  representation rather than the pure matrix representation in Equation 3.14.

<sup>21</sup>Diagonalizing the matrix  $W$  given by Lemma 3.2 corresponds to finding a representation of this form.

### 3.3 Efficient Sampling via BLT Multiplication

The generating function view outlined in Section 3.1 gives us a matrix factorization from any function  $r(x)$ . Namely,  $A^{(n)} = M(b, n) \cdot M(c, n)$  for  $b(x) = r(x)/(1-x)$  and  $c(x) = 1/r(x)$ . This turns out to be a good matrix factorization as long as  $r(x) \approx \sqrt{1-x}$ . The other desideratum is being able to efficiently sample noise according to the matrix factorization.

In this subsection, we show that if  $r(x)$  is a rational function of low degree, then there exists an efficient sampling algorithm or, equivalently, an efficient algorithm for streaming multiplication by the lower-triangular Toeplitz matrix  $M(r(x), n)$ . This algorithm relies on the representation in Section 3.2. The final missing ingredient, which we provide in Sections 4 and 5, is to instantiate  $r(x)$ .

Our algorithmic task is as follows. Fix a lower-triangular Toeplitz matrix  $M(r, n) \in \mathbb{R}^{n \times n}$  with a generating function  $r$ . We are given as input a matrix  $Z \in \mathbb{R}^{n \times m}$  and must produce as output  $\hat{Z} := M(r, n) \cdot Z \in \mathbb{R}^{n \times m}$ . The obvious algorithm for this task is to use standard matrix multiplication; this would take  $O(n^2m)$  time and  $O(nm)$  space. Another approach is to exploit the Toeplitz structure of  $M(r, n)$  which makes the matrix multiplication a convolution; this means it can be accelerated using the fast Fourier transform to take  $O(nm \log n)$  time, but this is not a streaming algorithm and it still requires  $O(nm)$  space. In order to reduce the space required we need an algorithm that is tailored to the streaming setting.

For streaming prefix sum applications (and private deep learning applications like DP-SGD and DP-FTRL in particular), the input  $Z$  consists of  $nm$  independent samples from a Gaussian and  $\hat{Z} = M(r, n) \cdot Z$  is the correlated noise we add to our private learning procedure – each row corresponds to one training iteration and each column to one parameter of the model. In particular, the rows  $Z$  can be sampled as needed – since they are independent noise – and the output  $\hat{Z}$  can be returned one row at a time.

This is our streaming setting: At each iteration  $k$ , our algorithm receives as input the next row  $Z_{k,\cdot} \in \mathbb{R}^m$  and must output the next row  $\hat{Z}_{k,\cdot} \in \mathbb{R}^m$ . Our goal is to develop an algorithm that runs in time and space  $O(m)$  per iteration (i.e.,  $O(nm)$  time in total). Reducing the space usage relies on the fact that we do not need to store all of  $Z$  or  $\hat{Z}$  in the streaming setting. Accomplishing this relies on the structure of the lower triangular Toeplitz matrix  $M(r, n) \in \mathbb{R}^{d \times d}$  generated by a rational function

$$r(x) = t + \frac{\sum_{i=0}^{d-1} p_i x^i}{1 + \sum_{j=1}^d q_j x^j} = r_0 + r_1 x + r_2 x^2 + r_3 x^3 + \dots$$

Following Remark 3.3 and Lemma 3.2, we use the representation of the sequence  $(r_k)_{k=0}^\infty$  in terms of matrix powers:  $r_k = u^T W^k v + t \mathbb{I}[k=0]$ . This is the representation that we use for our algorithm.

For a BLT given via a rational generating function  $r$  in the matrix representation of Lemma 3.2, Algorithm 1 in fact computes  $\hat{Z} = M(r)Z$  in row-by-row streaming fashion:

**Lemma 3.4** (Properties of Algorithm 1). *Algorithm 1 takes as input a stream  $Z_{0,\cdot}, Z_{1,\cdot}, \dots, Z_{n-1,\cdot} \in \mathbb{R}^{1 \times m}$  and outputs a stream  $\hat{Z}_{0,\cdot}, \hat{Z}_{1,\cdot}, \dots, \hat{Z}_{n-1,\cdot} \in \mathbb{R}^{1 \times m}$ . It takes parameters  $u, v \in \mathbb{R}^{d \times 1}$  and  $W \in \mathbb{R}^{d \times d}$  and  $t \in \mathbb{R}$ . At each iteration  $k$ , output satisfies*

$$\hat{Z}_{k,\cdot} = \sum_{j=0}^k r_{k-j} \cdot Z_{j,\cdot} = t Z_{k,\cdot} + \sum_{j=0}^k u^T W^{k-j} v \cdot Z_{j,\cdot}, \quad (3.15)$$

---

**Algorithm 1** Streaming Multiplication by a BLT Matrix

---

**Parameters:** Matrix  $M(r, n)$  defined following Lemma 3.2 and Remark 3.3 via column vectors  $u, v \in \mathbb{R}^{d \times 1}$ , a matrix  $W \in \mathbb{R}^{d \times d}$ , and a scalar  $t \in \mathbb{R}$ .

**Streaming Input:** Row vectors  $Z_{0,\cdot}, Z_{1,\cdot}, \dots, Z_{n-1,\cdot} \in \mathbb{R}^{1 \times m}$ .

**Streaming Output:** Row vectors  $\hat{Z}_{0,\cdot}, \hat{Z}_{1,\cdot}, \dots, \hat{Z}_{n-1,\cdot} \in \mathbb{R}^{1 \times m}$  satisfying  $\hat{Z}_k = (M(r)Z)_k$ .

**Goal:**  $\hat{Z}_{k,\cdot} = \sum_{i=0}^k r_i Z_{k-i,\cdot}$ , where  $r_k = u^T W^k v + t\mathbb{I}[k=0]$ .

Initialize  $S_0 = 0 \in \mathbb{R}^{d \times m}$ .

▷ State of the Algorithm

**for**  $k = 0, \dots, n-1$  **do**

    Receive input  $Z_{k,\cdot} \in \mathbb{R}^{1 \times m}$ .

    Compute  $S_{k+1} = v Z_{k,\cdot} + W \cdot S_k \in \mathbb{R}^{d \times m}$ .

    Return output  $\hat{Z}_{k,\cdot} = t Z_{k,\cdot} + u^T S_{k+1} \in \mathbb{R}^{1 \times m}$ .

---

where  $r_k = u^T W^k v + t\mathbb{I}[k=0]$ . The space usage is  $O(dm + d^2)$  and the runtime per iteration is dominated by a matrix multiplication  $W \cdot S$ , where  $S \in \mathbb{R}^{d \times m}$ .

*Proof.* By induction, for all  $k$ , we have

$$S_{k+1} = \sum_{j=0}^k W^{k-j} v Z_{j,\cdot}$$

and, hence,

$$t Z_{k,\cdot} + u^T S_{k+1} = t Z_{k,\cdot} + \sum_{j=0}^k u^T W^{k-j} v Z_{j,\cdot} = \sum_{j=0}^k r_{k-j} Z_{j,\cdot} = \hat{Z}_{k,\cdot},$$

where  $\hat{Z} = M(r, n)Z$ .

The state of the algorithm after iteration  $k$  is given by  $S_{k+1} \in \mathbb{R}^{d \times m}$ . The space usage of the algorithm is dominated by storing this state ( $dm$  registers) plus storing the parameters  $u, v, W$  ( $2d + d^2$  registers). More precisely, the only space that the algorithm requires is the state, plus whatever registers are required to perform the update  $S_{k+1} = v Z_{k,\cdot} + W \cdot S_k$ . (The output computation  $\hat{Z}_{k,\cdot} = t Z_{k,\cdot} + u^T S_{k+1}$  requires  $m$  registers to store the output.) The exact number of registers required for this update depends on the structure of  $v$  and  $W$ . If  $W$  is a diagonal matrix or has the structure given in Equation 3.9, then the update can be performed “in place” and we only require  $m$  extra registers to store the input  $Z_{k,\cdot}$ . In general, we can always perform this update using  $dm$  additional registers.

The running time of one iteration consists of the update  $S_{k+1} = v Z_{k,\cdot} + W \cdot S_k$  and the matrix-vector product  $\hat{Z}_{k,\cdot} = u^T S_{k+1}$ . The matrix-matrix product will dominate over the matrix-vector products. The runtime of the matrix-matrix product depends on the structure of  $W$ . If  $W$  is sparse (e.g., if it is diagonal), then this is  $O(dm)$  time. In general, we can perform this update in  $O(d^2 m)$  time.  $\square$

## 4 Factorizations via Rational Function Approximation

In this section we prove Theorem 1.1. We follow the generating functions view presented in Section 3 and use the corresponding sampling algorithm (Algorithm 1). The proof is split into three

steps. In Section 4.1 we recap how a generating function  $r$  yields a valid matrix factorization and prove that, if  $r(x) \approx \sqrt{1-x}$ , then this yields an approximately optimal factorization. In Section 4.2 we construct a low-degree rational function  $r$  that appropriately approximates the square root. Finally we assemble the parts of the proof in Section 4.3.

## 4.1 Reduction to Approximating the Square Root

Proposition 2.2 shows that the optimal matrix factorization is given by  $A^{(n)} = BC$  where  $B = C = M(f, n)$  and  $f(x) = 1/\sqrt{1-x}$ . Suppose we have a generating function  $r(x) \approx \sqrt{1-x}$ . Let  $b(x) := \frac{r(x)}{1-x} \approx \frac{1}{\sqrt{1-x}} = f(x)$  and  $c(x) := \frac{1}{r(x)} \approx \frac{1}{\sqrt{1-x}}$ . Since  $b(x) \cdot c(x) = \frac{r(x)}{1-x} \cdot \frac{1}{r(x)} = \frac{1}{1-x} = g(x)$ , we always obtain a valid factorization  $M(b, n) \cdot M(c, n) = A^{(n)}$ . We can bound the objective value of this factorization in terms of the approximation  $r(x) \approx \sqrt{1-x}$ .

**Proposition 4.1** (Approximating the square root approximates the matrix factorization objective). *Let  $\tau > 0$  and  $n \in \mathbb{N}$ . Let  $r : C \rightarrow \mathbb{C}$ , where  $C := \{z \in \mathbb{C} : |z| < 1\}$  is the open unit disc in the complex plane centered at zero. Let*

$$\gamma_\tau := \sup \{ |r(x) - \sqrt{1-x}| : x \in \mathbb{C}, |x| = \exp(-\tau) \}.$$

*Let  $b(x) := \frac{r(x)}{1-x}$  and  $c(x) := \frac{1}{r(x)}$  and  $f(x) = \frac{1}{\sqrt{1-x}}$ . Then  $M(b, n) \cdot M(c, n) = M(c, n) \cdot M(b, n) = A^{(n)} = M(f, n)^2$ , where  $M(\cdot, n)$  is defined in Equation 3.1 and  $A^{(n)}$  is defined in Equation 2.1. Moreover, if  $\gamma_\tau \leq \left( \frac{1 - \exp(-2\tau)}{4} \right)^2$ , then*

$$\begin{aligned} \|M(b, n)\|_{2 \rightarrow \infty} &\leq \|M(f, n)\|_{2 \rightarrow \infty} + \frac{\exp(\tau n) \cdot \gamma_\tau}{\sqrt{\exp(2\tau) - 1}}, \\ \|M(c, n)\|_{1 \rightarrow 2} &\leq \|M(f, n)\|_{1 \rightarrow 2} + \frac{\exp(\tau n) \cdot \gamma_\tau}{\sqrt[4]{(\exp(2\tau) - 1)^2 - 2^{7/2} \gamma_\tau \exp(4\tau)}}. \end{aligned}$$

We will end up setting  $\tau = \frac{1}{2n}$ . This ensures that  $\frac{\exp(\tau n) \cdot \gamma_\tau}{\sqrt{\exp(2\tau) - 1}} \leq \sqrt{n} \cdot \exp(1/2) \cdot \gamma_\tau < 2\sqrt{n} \gamma_\tau$  and, if  $n \geq 6$  and  $\gamma_\tau \leq \frac{1}{32 \cdot n^2}$ , then  $\frac{\exp(\tau n) \cdot \gamma_\tau}{\sqrt[4]{(\exp(2\tau) - 1)^2 - 2^{7/2} \gamma_\tau \exp(4\tau)}} \leq \sqrt[4]{2} \cdot \sqrt{n} \cdot \exp(1/2) \cdot \gamma_\tau < 2\sqrt{n} \gamma_\tau$ .

*Proof.* Let  $b(x) = \sum_{k=0}^{\infty} b_k x^k$ ,  $c(x) = \sum_{k=0}^{\infty} c_k x^k$ , and  $f(x) = \sum_{k=0}^{\infty} f_k x^k$ . By linearity and the triangle inequality,

$$\begin{aligned} \|M(b, n)\|_{2 \rightarrow \infty} &= \|M(f, n) + M(b - f, n)\|_{2 \rightarrow \infty} \leq \|M(f, n)\|_{2 \rightarrow \infty} + \|M(b - f, n)\|_{2 \rightarrow \infty}, \\ \|M(c, n)\|_{1 \rightarrow 2} &= \|M(f, n) + M(c - f, n)\|_{1 \rightarrow 2} \leq \|M(f, n)\|_{1 \rightarrow 2} + \|M(c - f, n)\|_{1 \rightarrow 2}. \end{aligned}$$

Thus it suffices to bound  $\|M(b - f, n)\|_{2 \rightarrow \infty} = \sqrt{\sum_{k=0}^{n-1} (b_k - f_k)^2}$  and  $\|M(c - f, n)\|_{1 \rightarrow 2} = \sqrt{\sum_{k=0}^{n-1} (c_k - f_k)^2}$ . We bound these using a weighted version of Parseval's identity.

**Lemma 4.2** (Weighted Parseval's Identity). *Let  $g, h : C \rightarrow \mathbb{C}$  be analytic, where  $C := \{z \in \mathbb{C} : |z| < 1\}$  is the open unit disc in the complex plane centered at zero. Let  $g(x) = \sum_{k=0}^{\infty} g_k x^k$  and*



$h(x) = \sum_{k=0}^{\infty} h_k x^k$  for  $x \in C$ , where  $g_k, h_k \in \mathbb{C}$ . Let  $\tau > 0$ . Then

$$\begin{aligned} \frac{1}{2\pi} \int_{-\pi}^{\pi} |g(\exp(\sqrt{-1}\theta - \tau)) - h(\exp(\sqrt{-1}\theta - \tau))|^2 d\theta &= \sum_{k=0}^{\infty} |g_k - h_k|^2 \cdot \exp(-2\tau k) \\ &\geq \exp(-2\tau(n-1)) \sum_{k=0}^{n-1} |g_k - h_k|^2. \end{aligned}$$

*Proof.* Let  $w(x) = g(x) - h(x) = \sum_{k=0}^{\infty} w_k x^k$ , where  $w_k = g_k - h_k$ . Then

$$\begin{aligned} &\frac{1}{2\pi} \int_{-\pi}^{\pi} |g(\exp(\sqrt{-1}\theta - \tau)) - h(\exp(\sqrt{-1}\theta - \tau))|^2 d\theta \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} |w(\exp(\sqrt{-1}\theta - \tau))|^2 d\theta \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} w(\exp(\sqrt{-1}\theta - \tau)) \cdot \overline{w(\exp(\sqrt{-1}\theta - \tau))} d\theta \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \left( \sum_{k=0}^{\infty} w_k \cdot (\exp(\sqrt{-1}\theta - \tau))^k \right) \cdot \overline{\left( \sum_{\ell=0}^{\infty} w_{\ell} \cdot (\exp(\sqrt{-1}\theta - \tau))^{\ell} \right)} d\theta \\ &= \sum_{k,\ell=0}^{\infty} \frac{1}{2\pi} \int_{-\pi}^{\pi} w_k \cdot (\exp(\sqrt{-1}\theta - \tau))^k \cdot \overline{w_{\ell}} \cdot (\exp(-\sqrt{-1}\theta - \tau))^{\ell} d\theta \\ &= \sum_{k,\ell=0}^{\infty} w_k \cdot \overline{w_{\ell}} \cdot \exp(-(k+\ell)\tau) \cdot \frac{1}{2\pi} \int_{-\pi}^{\pi} \exp((k-\ell)\sqrt{-1}\theta) d\theta \\ &= \sum_{k,\ell=0}^{\infty} w_k \cdot \overline{w_{\ell}} \cdot \exp(-(k+\ell)\tau) \cdot \mathbb{I}[k-\ell=0] \\ &= \sum_{k=0}^{\infty} |w_k|^2 \cdot \exp(-2k\tau) \\ &= \sum_{k=0}^{\infty} |g_k - h_k|^2 \cdot \exp(-2\tau k). \end{aligned}$$

□

Recall  $b(x) := \frac{r(x)}{1-x}$  and  $f(x) = \frac{1}{\sqrt{1-x}}$ , whence  $b(x) - f(x) = \frac{r(x) - \sqrt{1-x}}{1-x}$ . Now we have

$$\begin{aligned}
\|M(b - f, n)\|_{2 \rightarrow \infty} &= \sqrt{\sum_{k=0}^{n-1} (b_k - f_k)^2} \\
&\leq \exp(\tau(n-1)) \sqrt{\frac{1}{2\pi} \int_{-\pi}^{\pi} |b(\exp(\sqrt{-1}\theta - \tau)) - f(\exp(\sqrt{-1}\theta - \tau))|^2 d\theta} \\
&= \exp(\tau(n-1)) \sqrt{\frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{\left| r(\exp(\sqrt{-1}\theta - \tau)) - \sqrt{1 - \exp(\sqrt{-1}\theta - \tau)} \right|^2}{|1 - \exp(\sqrt{-1}\theta - \tau)|^2} d\theta} \\
&\leq \exp(\tau(n-1)) \sqrt{\frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{\gamma_\tau^2}{|1 - \exp(\sqrt{-1}\theta - \tau)|^2} d\theta} \\
&= \exp(\tau(n-1)) \cdot \gamma_\tau \cdot \sqrt{\frac{\exp(\tau)}{4\pi} \int_{-\pi}^{\pi} \frac{1}{\cosh(\tau) + \cos(\theta)} d\theta} \\
&= \exp(\tau(n-1)) \cdot \gamma_\tau \cdot \sqrt{\frac{\exp(\tau)}{2} \frac{1}{\sqrt{\cosh(\tau)^2 - 1}}} \\
&= \frac{\exp(\tau(n-1)) \cdot \gamma_\tau}{\sqrt{1 - \exp(-2\tau)}} = \frac{\exp(\tau n) \cdot \gamma_\tau}{\sqrt{\exp(2\tau) - 1}} \leq \frac{\exp(\tau n) \cdot \gamma_\tau}{\sqrt{2\tau}}.
\end{aligned}$$

In the above we use the identity

$$\begin{aligned}
|1 - \exp(\sqrt{-1}\theta - \tau)|^2 &= (1 - e^{-\tau} \cos \theta)^2 + (e^{-\tau} \sin \theta)^2 = 1 - 2e^{-\tau} \cos \theta + e^{-2\tau} (\cos^2 \theta + \sin^2 \theta) \\
&= e^{-\tau} \cdot (e^\tau - 2 \cos \theta + e^{-\tau}) = 2e^{-\tau} \cdot (\cosh \tau + \cos \theta)
\end{aligned}$$

and the integral

$$\forall \sigma > 1 \quad \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{1}{\sigma + \cos(\theta)} d\theta = \frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{1}{\sigma + \sin(\theta)} d\theta = \frac{1}{\sqrt{\sigma^2 - 1}}, \quad (4.1)$$

which follows from the derivative  $\frac{d}{d\theta} \left[ \frac{2}{\sqrt{\sigma^2 - 1}} \tan^{-1} \left( \frac{1 + \sigma \tan(\theta/2)}{\sqrt{\sigma^2 - 1}} \right) \right] = \frac{1}{\sigma + \sin(\theta)}$ .

Similarly, recall  $c(x) := \frac{1}{r(x)}$  and  $f(x) = \frac{1}{\sqrt{1-x}}$ , whence  $c(x) - f(x) = \frac{\sqrt{1-x} - r(x)}{r(x) \cdot \sqrt{1-x}}$ . Assuming

$\gamma_\tau < \frac{(1-\exp(-2\tau))^2}{2^{7/2}}$ , we have

$$\begin{aligned}
& \|M(c - f, n)\|_{1 \rightarrow 2} \\
&= \sqrt{\sum_{k=0}^{n-1} (c_k - f_k)^2} \\
&\leq \exp(\tau(n-1)) \sqrt{\frac{1}{2\pi} \int_{-\pi}^{\pi} |c(\exp(\sqrt{-1}\theta - \tau)) - f(\exp(\sqrt{-1}\theta - \tau))|^2 d\theta} \\
&= \exp(\tau(n-1)) \sqrt{\frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{\left| r(\exp(\sqrt{-1}\theta - \tau)) - \sqrt{1 - \exp(\sqrt{-1}\theta - \tau)} \right|^2}{\left| r(\exp(\sqrt{-1}\theta - \tau)) \cdot \sqrt{1 - \exp(\sqrt{-1}\theta - \tau)} \right|^2} d\theta} \\
&= \exp(\tau(n-1)) \sqrt{\frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{\left| r(\exp(\sqrt{-1}\theta - \tau)) - \sqrt{1 - \exp(\sqrt{-1}\theta - \tau)} \right|^2}{\left| 1 - \exp(\sqrt{-1}\theta - \tau) + \left( r(\exp(\sqrt{-1}\theta - \tau)) - \sqrt{1 - \exp(\sqrt{-1}\theta - \tau)} \right) \cdot \sqrt{1 - \exp(\sqrt{-1}\theta - \tau)} \right|^2} d\theta} \\
&\leq \exp(\tau(n-1)) \sqrt{\frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{\gamma_\tau^2}{\left( |1 - \exp(\sqrt{-1}\theta - \tau)| - \gamma_\tau \cdot \sqrt{|1 - \exp(\sqrt{-1}\theta - \tau)|} \right)^2} d\theta} \\
&\leq \exp(\tau(n-1)) \sqrt{\frac{1}{2\pi} \int_{-\pi}^{\pi} \frac{\gamma_\tau^2}{|1 - \exp(\sqrt{-1}\theta - \tau)|^2 - 2\sqrt{2}\gamma_\tau} d\theta} \\
&= \exp(\tau(n-1)) \cdot \gamma_\tau \cdot \sqrt{\frac{\exp(\tau)}{4\pi} \int_{-\pi}^{\pi} \frac{1}{\cosh(\tau) - \sqrt{2}\exp(\tau)\gamma_\tau + \cos(\theta)} d\theta} \\
&= \exp(\tau(n-1)) \cdot \gamma_\tau \cdot \sqrt{\frac{\exp(\tau)}{2} \frac{1}{\sqrt{(\cosh(\tau) - \sqrt{2}\exp(\tau)\gamma_\tau)^2 - 1}}} \\
&= \frac{\exp(\tau n) \cdot \gamma_\tau}{\sqrt{2\exp(\tau) \sqrt{(\cosh(\tau) - \sqrt{2}\exp(\tau)\gamma_\tau)^2 - 1}}} \\
&= \frac{\exp(\tau n) \cdot \gamma_\tau}{\sqrt[4]{(\exp(2\tau)(1 - 2\sqrt{2}\gamma_\tau) + 1)^2 - 4\exp(2\tau)}} \\
&= \frac{\exp(\tau n) \cdot \gamma_\tau}{\sqrt[4]{1 - 2(1 + 2^{3/2}\gamma_\tau)\exp(2\tau) + (1 - 2^{3/2}\gamma_\tau)^2\exp(4\tau)}} \\
&= \frac{\exp(\tau n) \cdot \gamma_\tau}{\sqrt[4]{(\exp(2\tau) - 1)^2 - 2^{5/2}\gamma_\tau\exp(2\tau) - 2^{5/2}\gamma_\tau\exp(4\tau) + 2^3\gamma_\tau^2\exp(4\tau)}} \\
&\leq \frac{\exp(\tau n) \cdot \gamma_\tau}{\sqrt[4]{(\exp(2\tau) - 1)^2 - 2^{5/2}\gamma_\tau\exp(2\tau)(1 + \exp(2\tau))}} \\
&\leq \frac{\exp(\tau n) \cdot \gamma_\tau}{\sqrt[4]{(\exp(2\tau) - 1)^2 - 2^{7/2}\gamma_\tau\exp(4\tau)}}.
\end{aligned}$$

Combining the bounds yields the result.  $\square$

## 4.2 Rational Approximation of $\sqrt{x}$

Sections 3 and 4.1 reduce the problem to finding a low-degree rational function that uniformly approximates  $\sqrt{1-x}$  for  $x \in \mathbb{C}$  with  $|x| \leq 1$ . Fortunately, such functions are known to exist. In this subsection we construct the required rational functions. The proof is included for completeness and because it provides an explicit construction with explicit bounds.

For notational simplicity and consistency with the literature, we rescale so that we are approximating  $\sqrt{x}$  instead of  $\sqrt{1-x}$ . To get an equivalent result, we must uniformly approximate  $\sqrt{x}$  for  $x \in \mathbb{C}$  with  $|x - 1/2| \leq 1/2$ .

Newman [New64] gave an explicit rational function that approximates  $\sqrt{x}$  for  $x \in [0, 1]$ . Specifically, let

$$p(x) = \prod_{k=0}^{d-1} (x + \exp(-k/\sqrt{d})) \quad \text{and} \quad r(x) = \frac{\sqrt{x} \cdot (p(\sqrt{x}) - p(-\sqrt{x}))}{p(\sqrt{x}) + p(-\sqrt{x})}. \quad (4.2)$$

Then  $r(x)$  is a rational function of degree  $\lceil d/2 \rceil$  – specifically,  $r(x)$  is the ratio where the numerator is a polynomial in  $x$  of degree  $\lceil d/2 \rceil$  and the denominator is a polynomial in  $x$  of degree  $\lfloor d/2 \rfloor$ . Furthermore,

$$\sup_{x \in [0,1]} |r(x) - \sqrt{x}| \leq 3 \cdot \exp(-\sqrt{d}). \quad (4.3)$$

The surprising aspect of this result is that the dependence on the degree is exponential rather than polynomial, as is the case if we restrict to polynomial approximations, rather than rational approximations. Newman also showed that this result was optimal up to constant factors in the degree  $d$  and the approximation error. Subsequent work [Sta93] obtained optimal constants for this approximation.

In order to apply Proposition 4.1, we need to extend Newman’s result into the complex plane. We provide a proof adapted from that of Gopal and Trefethen [GT19].

**Theorem 4.3.** *Let  $d \in \mathbb{N}$  with  $d \geq 2$ . Then there exists a rational function  $r$  of degree  $d$  with real coefficients and real negative simple poles such that*

$$|r(x) - \sqrt{x}| \leq \left(4 + \frac{4}{\pi}\right) \cdot \exp\left(-\pi \sqrt{\frac{1}{2} \left\lfloor \frac{d-1}{2} \right\rfloor}\right) \leq 6 \cdot \exp\left(-\frac{\pi}{2} \sqrt{d-2}\right) \quad (4.4)$$

for all  $x \in \mathbb{C}$  with  $\Re(x) \geq 0$  and  $|x| \leq 1$ . Specifically,

$$r(x) = \frac{2h}{\pi} \sum_{k=-d_-}^{d_+} \frac{x \cdot \exp(hk)}{x + \exp(2hk)}, \quad \text{where } d_+ = \left\lfloor \frac{d-1}{2} \right\rfloor, \quad d_- = \left\lceil \frac{d-1}{2} \right\rceil, \quad \text{and } h = \frac{\pi}{\sqrt{2d_+}}. \quad (4.5)$$

*Proof.* We invoke Proposition 4.5. Let  $d_- = \lceil (d-1)/2 \rceil$  and  $d_+ = \lfloor (d-1)/2 \rfloor$  and let  $h \in (0, 4.9)$  be determined later. Let  $r(x) = r_{d_+, d_-, h}(x)$  be as in Equation 4.11. Clearly  $r(x)$  is a rational function of degree  $d_+ + d_- + 1 = d$  with real coefficients and the poles are given by  $-\exp(2hk) \in$

$(-\infty, 0)$  for  $k \in \{-d_-, -d_- + 1, \dots, d_+\}$ . It remains to simplify the approximation guarantee and set  $h$ . From Equation 4.11, for all  $x \in \mathbb{C}$  with  $\Re(x) \geq 0$  and  $|x| \leq 1$ , we have

$$\begin{aligned}
|r(x) - \sqrt{x}| &\leq 2\sqrt{|x|} \left( \frac{1}{\exp((1-c)\pi^2/h) - 1} + \frac{1}{\exp((1+c)\pi^2/h) - 1} \right) \\
&\quad + \frac{2h}{\pi(\exp(h) - 1)} (|x| \cdot \exp(-hd_+) + \exp(-hd_-)) \\
&\leq 2 \left( \frac{1}{\exp((1-|c|)\pi^2/h) - 1} + \frac{1}{\exp((1+|c|)\pi^2/h) - 1} \right) \\
&\quad \quad \quad (|x| \leq 1 \text{ \& symmetry in } c) \\
&\quad + \frac{4h \exp(-hd_+)}{\pi(\exp(h) - 1)} \quad (d_- \geq d_+) \\
&\leq 2 \left( \frac{1}{\exp(\pi^2/2h) - 1} + \frac{1}{\exp(\pi^2/h) - 1} \right) + \frac{4}{\pi} \cdot \exp(-hd_+) \cdot \frac{h}{\exp(h) - 1} \\
&\quad \quad \quad (|c| = |\frac{1}{\pi} \arg(x)| \leq \frac{1}{2}) \\
&\leq 4 \cdot \exp(-\pi^2/2h) + \frac{4}{\pi} \cdot \exp(-hd_+) \quad (4.6)
\end{aligned}$$

$$\begin{aligned}
&= \left(4 + \frac{4}{\pi}\right) \cdot \exp\left(-\pi\sqrt{\frac{d_+}{2}}\right) \quad (4.7) \\
&= \left(4 + \frac{4}{\pi}\right) \cdot \exp\left(-\pi\sqrt{\frac{1}{2} \left\lfloor \frac{d-1}{2} \right\rfloor}\right) \\
&\leq 6 \cdot \exp\left(-\frac{\pi}{2}\sqrt{d-1}\right).
\end{aligned}$$

The fourth inequality Equation 4.6 follows from two simple facts: First,  $h \geq 0$  implies  $\frac{h}{\exp(h)-1} \leq 1$ . Second,  $\frac{1}{a-1} + \frac{1}{a^2-1} \leq \frac{2}{a}$  when  $a = \exp(\pi^2/2h) \geq 1 + \sqrt{3}$  or, equivalently,  $h \leq \frac{\pi^2}{2 \log(1+\sqrt{3})} \approx 4.91$ . The penultimate equality Equation 4.7 follows by setting  $h = \frac{\pi}{\sqrt{2d_+}} \leq \frac{\pi}{\sqrt{2}} < 2.5$ .  $\square$

**Corollary 4.4** (Rescaling of Theorem 4.3). *Let  $d \geq 2$  be an integer. Then there exists a rational function  $\tilde{r}$  of degree  $d$  with real coefficients such that all poles are simple and  $> 1$  and we have*

$$|\tilde{r}(x) - \sqrt{1-x}| \leq 8 \cdot \exp\left(-\frac{\pi}{2}\sqrt{d-2}\right) \quad (4.8)$$

for all  $x \in \mathbb{C}$  with  $|x| \leq 1$ . Specifically,

$$\tilde{r}(x) = \frac{2h\sqrt{2}}{\pi} \sum_{k=-d_-}^{d_+} \frac{(1-x) \cdot \exp(hk)}{1-x+2 \cdot \exp(2hk)} = \frac{2h\sqrt{2}}{\pi} \sum_{k=-d_-}^{d_+} \exp(hk) - \frac{2 \cdot \exp(3hk)}{1+2 \cdot \exp(2hk) - x} \quad (4.9)$$

where  $d_+ = \lfloor (d-1)/2 \rfloor$ ,  $d_- = \lceil (d-1)/2 \rceil$ , and  $h = \frac{\pi}{\sqrt{2d_+}}$ .

*Proof.* Let  $r$  be the rational function of degree  $\leq d$  promised by Theorem 4.3. Define  $\tilde{r}(x) = \sqrt{2} \cdot r(\frac{1-x}{2})$ . Suppose  $x \in \mathbb{C}$  with  $|x| \leq 1$ . Then  $\Re(\frac{1-x}{2}) = \frac{1}{2} - \frac{1}{2}\Re(x) \geq 0$ ,  $|\frac{1-x}{2}| \leq \frac{1+|x|}{2} \leq 1$ ,

and

$$|\tilde{r}(x) - \sqrt{1-x}| = \sqrt{2} \cdot \left| r\left(\frac{1-x}{2}\right) - \sqrt{\frac{1-x}{2}} \right| \leq \sqrt{2} \cdot \left(4 + \frac{4}{\pi}\right) \cdot \exp\left(-\pi \sqrt{\frac{1}{2} \left\lceil \frac{d-1}{2} \right\rceil}\right) \leq 8 \exp\left(-\frac{\pi}{2} \sqrt{d-1}\right).$$

If  $\tilde{x}$  is a pole of  $\tilde{r}$ , then  $\frac{1-\tilde{x}}{2}$  is a pole of  $r$ ; since the poles of  $r$  are negative we have  $\frac{1-\tilde{x}}{2} < 0$  and, hence,  $\tilde{x} > 1$ .  $\square$

The general form of the approximation promised by Theorem 4.3 is given below.

**Proposition 4.5.** *Let  $d_+, d_- \in \mathbb{N}$  and  $h > 0$ . For  $x \in \mathbb{C} \setminus [-\exp(2d_+h), -\exp(-2d_-h)]$ , define*

$$r_{d_+, d_-, h}(x) := \frac{2h}{\pi} \sum_{k=-d_-}^{d_+} \frac{x \cdot \exp(hk)}{x + \exp(2hk)}. \quad (4.10)$$

Then, for all  $x \in \mathbb{C}$  with  $\Re(x) \geq 0$ , we have

$$\begin{aligned} |r_{d_+, d_-, h}(x) - \sqrt{x}| &\leq 2\sqrt{|x|} \left( \frac{1}{\exp((1-c)\pi^2/h) - 1} + \frac{1}{\exp((1+c)\pi^2/h) - 1} \right) \\ &\quad + \frac{2h}{\pi(\exp(h) - 1)} (|x| \cdot \exp(-hd_+) + \exp(-hd_-)), \end{aligned} \quad (4.11)$$

where  $c = \frac{1}{\pi} \arg(x) \in [-\frac{1}{2}, \frac{1}{2}]$ .

*Proof.* Fix  $x \in \mathbb{C}$  with  $\Re(x) > 0$ .<sup>22</sup> Let  $\arg(x) = \tan^{-1}\left(\frac{\Im(x)}{\Re(x)}\right) = c\pi \in (-\frac{\pi}{2}, \frac{\pi}{2})$ .

We begin with the Cauchy distribution integral

$$\int_0^\infty \frac{1}{1+u^2} du = \int_0^\infty \left( \frac{d}{du} \tan^{-1}(u) \right) du = \frac{\pi}{2}.$$

We perform the variable substitutions  $u = v/\sqrt{x}$  and  $v = \exp(s)$  and rearrange to obtain

$$\sqrt{x} = \frac{2}{\pi} \int_0^\infty \frac{x}{x+v^2} dv = \frac{2}{\pi} \int_{-\infty}^\infty \frac{x \cdot \exp(s)}{x + \exp(2s)} ds. \quad (4.12)$$

Now we make the approximation

$$\sqrt{x} = \frac{2}{\pi} \int_{-\infty}^\infty \frac{x \cdot \exp(s)}{x + \exp(2s)} ds \approx \frac{2h}{\pi} \sum_{k \in \mathbb{Z}} \frac{x \cdot \exp(hk)}{x + \exp(2hk)} \approx \frac{2h}{\pi} \sum_{k=-d_-}^{d_+} \frac{x \cdot \exp(hk)}{x + \exp(2hk)} = r_{d_+, d_-, h}(x). \quad (4.13)$$

Thus all that remains to complete the proof is to make the the two approximations precise. Intuitively, the first approximation replaces the continuous integral with a discrete-but-infinite Riemann sum. If the function is smooth and  $h$  is small, this approximation should be good. The second approximation truncates the infinite sum, which is a good approximation as long as the tails of the function decay rapidly and  $hd_+$  and  $hd_-$  are large.

<sup>22</sup>For the proof we assume  $\Re(x) > 0$ . The case  $\Re(x) = 0$  follows by continuity.

Define  $f(z) := \frac{2h}{\pi} \frac{x \cdot \exp(hz)}{x + \exp(2hz)}$ . Let

$$\begin{aligned}
\widehat{f}(t) &:= \int_{-\infty}^{\infty} f(z) \cdot \exp(-2\pi z t \sqrt{-1}) dz \\
&= \int_{-\infty}^{\infty} \frac{2h}{\pi} \frac{x \cdot \exp(hz - 2\pi z t \sqrt{-1})}{x + \exp(2hz)} dz \\
&= \frac{2h}{\pi} \int_{-\infty}^{\infty} \frac{\exp(-2\pi z t \sqrt{-1})}{\exp(-hz) + \exp(hz - \log x)} dz \\
&= \frac{2h}{\pi} \int_{-\infty}^{\infty} \frac{\exp(-2\pi(y + \frac{\log x}{2h})t \sqrt{-1})}{\exp(-hy - \frac{1}{2} \log x) + \exp(hy - \frac{1}{2} \log x)} dy \quad (z = y + \frac{\log x}{2h}) \\
&= \sqrt{x} \cdot \frac{2h}{\pi} \cdot \exp\left(-\pi t \frac{\log x}{h} \sqrt{-1}\right) \cdot \int_{-\infty}^{\infty} \frac{\exp(-2\pi y t \sqrt{-1})}{\exp(-hy) + \exp(hy)} dy \\
&= \sqrt{x} \cdot \frac{2h}{\pi} \cdot \exp\left(-\pi t \frac{\log x}{h} \sqrt{-1}\right) \cdot \frac{1}{2} \int_{-\infty}^{\infty} \frac{\exp(-2\pi y t \sqrt{-1})}{\cosh(hy)} dy \\
&= \sqrt{x} \cdot \frac{2h}{\pi} \cdot \exp\left(-\pi t \frac{\log x}{h} \sqrt{-1}\right) \cdot \frac{\pi}{2h \cdot \cosh(\pi^2 t/h)} \tag{4.14} \\
&= \sqrt{x} \cdot \exp\left(-\pi t \frac{\log x}{h} \sqrt{-1}\right) \cdot \frac{1}{\cosh(\pi^2 t/h)}.
\end{aligned}$$

The penultimate equality (4.14) is the Fourier transform of the hyperbolic secant [use14]. The Poisson summation formula states that

$$\sum_{k \in \mathbb{Z}} f(k) = \sum_{t \in \mathbb{Z}} \widehat{f}(t). \tag{4.15}$$

Per Equation 4.12, we have

$$\widehat{f}(0) = \int_{-\infty}^{\infty} f(z) dz = \int_{-\infty}^{\infty} \frac{2h}{\pi} \frac{x \cdot \exp(hz)}{x + \exp(2hz)} dz = \int_{-\infty}^{\infty} \frac{2}{\pi} \frac{x \cdot \exp(s)}{x + \exp(2s)} ds = \sqrt{x}.$$

Thus the Poisson summation formula allows us to analyze the first approximation in Equation 4.13.

Namely,

$$\begin{aligned}
& \left| \sqrt{x} - \frac{2h}{\pi} \sum_{k \in \mathbb{Z}} \frac{x \cdot \exp(hk)}{x + \exp(2hk)} \right| \\
&= \left| \widehat{f}(0) - \sum_{k \in \mathbb{Z}} f(k) \right| \\
&= \left| \widehat{f}(0) - \sum_{t \in \mathbb{Z}} \widehat{f}(t) \right| \\
&\leq \sum_{t=1}^{\infty} |\widehat{f}(t)| + |\widehat{f}(-t)| \\
&= \sum_{t=1}^{\infty} |\sqrt{x}| \cdot \left( \left| \exp \left( -\pi t \frac{\log x}{h} \sqrt{-1} \right) \right| + \left| \exp \left( \pi t \frac{\log x}{h} \sqrt{-1} \right) \right| \right) \cdot \frac{1}{\cosh(\pi^2 t/h)} \\
&= \sum_{t=1}^{\infty} |\sqrt{x}| \cdot \left( \exp \left( \Im \left( \pi t \frac{\log x}{h} \right) \right) + \exp \left( \Im \left( -\pi t \frac{\log x}{h} \right) \right) \right) \cdot \frac{1}{\cosh(\pi^2 t/h)} \\
&\quad (|\exp(v\sqrt{-1})| = \exp(\Re(v\sqrt{-1})) = \exp(\Im(-v))) \\
&= \sqrt{|x|} \sum_{t=1}^{\infty} 2 \cosh \left( \pi \frac{t}{h} \Im(\log x) \right) \cdot \frac{1}{\cosh(\pi^2 t/h)} \\
&= \sqrt{|x|} \sum_{t=1}^{\infty} 2 \cosh \left( \pi \frac{t}{h} \cdot c\pi \right) \cdot \frac{1}{\cosh(\pi^2 t/h)} \quad (\Im(\log x) = \arg(x) = c\pi) \\
&= 2\sqrt{|x|} \sum_{t=1}^{\infty} \frac{\exp(c\pi^2 t/h) + \exp(-c\pi^2 t/h)}{\exp(\pi^2 t/h) + \exp(-\pi^2 t/h)} \\
&\leq 2\sqrt{|x|} \sum_{t=1}^{\infty} \frac{\exp(c\pi^2 t/h) + \exp(-c\pi^2 t/h)}{\exp(\pi^2 t/h) + 0} \\
&= 2\sqrt{|x|} \sum_{t=1}^{\infty} \exp(-(1-c)\pi^2 t/h) + \exp(-(1+c)\pi^2 t/h) \\
&= 2\sqrt{|x|} \left( \frac{1}{\exp((1-c)\pi^2/h) - 1} + \frac{1}{\exp((1+c)\pi^2/h) - 1} \right). \tag{4.16}
\end{aligned}$$

The final equality (4.16) follows from the usual geometric series formula: If  $a > 0$ , then  $\sum_{t=1}^{\infty} \exp(-at) = \frac{\exp(-a)}{1 - \exp(-a)} = \frac{1}{\exp(a) - 1}$ .

Next we analyze the second approximation in Equation 4.13, which requires bounding the tails



of  $f$ . On one side, for  $k \in \mathbb{N}$ , we have

$$\begin{aligned}
|f(k)| &= \left| \frac{2h}{\pi} \frac{x \cdot \exp(hk)}{x + \exp(2hk)} \right| \\
&= \frac{2h}{\pi} \frac{|x|}{|x \cdot \exp(-hk) + \exp(hk)|} \\
&= \frac{2h}{\pi} \frac{|x|}{\sqrt{\Re(x \cdot \exp(-hk) + \exp(hk))^2 + \Im(x \cdot \exp(-hk) + \exp(hk))^2}} \\
&\leq \frac{2h}{\pi} \frac{|x|}{|\Re(x \cdot \exp(-hk) + \exp(hk))|} \\
&= \frac{2h}{\pi} \frac{|x|}{|\Re(x) \cdot \exp(-hk) + \exp(hk)|} \quad (hk \in \mathbb{R}) \\
&= \frac{2h}{\pi} \frac{|x|}{\Re(x) \cdot \exp(-hk) + \exp(hk)} \quad (\Re(x) > 0) \\
&\leq \frac{2h}{\pi} \frac{|x|}{\exp(hk)}. \quad (4.17)
\end{aligned}$$

On the other side, for  $k \in \mathbb{N}$ , we have

$$\begin{aligned}
|f(-k)| &= \left| \frac{2h}{\pi} \frac{x \cdot \exp(-hk)}{x + \exp(-2hk)} \right| \\
&= \frac{2h}{\pi} \frac{\exp(-hk)}{|1 + \exp(-2hk)/x|} \\
&\leq \frac{2h}{\pi} \frac{\exp(-hk)}{|\Re(1 + \exp(-2hk)/x)|} \\
&= \frac{2h}{\pi} \frac{\exp(-hk)}{|1 + \exp(-2hk)/\Re(x)|} \quad (\Re(1/x) = 1/\Re(x)) \\
&= \frac{2h}{\pi} \frac{\exp(-hk)}{1 + \exp(-2hk)/\Re(x)} \quad (\Re(x) > 0) \\
&\leq \frac{2h}{\pi} \exp(-hk). \quad (4.18)
\end{aligned}$$

Combining Equations 4.17 and 4.18, we have

$$\begin{aligned}
\left| \frac{2h}{\pi} \sum_{k \in \mathbb{Z}} \frac{x \cdot \exp(hk)}{x + \exp(2hk)} - r_{d_+, d_-, h}(x) \right| &= \frac{2h}{\pi} \left| \sum_{k \in \mathbb{Z}} \frac{x \cdot \exp(hk)}{x + \exp(2hk)} - \sum_{k=-d_-}^{d_+} \frac{x \cdot \exp(hk)}{x + \exp(2hk)} \right| \\
&= \left| \sum_{k=d_++1}^{\infty} f(k) + \sum_{k=d_-+1}^{\infty} f(-k) \right| \\
&\leq \sum_{k=d_++1}^{\infty} |f(k)| + \sum_{k=d_-+1}^{\infty} |f(-k)| \\
&\leq \frac{2h}{\pi} \sum_{k=d_++1}^{\infty} \frac{|x|}{\exp(hk)} + \frac{2h}{\pi} \sum_{k=d_-+1}^{\infty} \exp(-hk) \\
&= \frac{2h}{\pi} |x| \frac{\exp(-h(d_++1))}{1 - \exp(-h)} + \frac{2h}{\pi} \frac{\exp(-h(d_-+1))}{1 - \exp(-h)} \\
&\quad (4.19) \\
&= \frac{2h}{\pi(\exp(h) - 1)} (|x| \cdot \exp(-hd_+) + \exp(-hd_-)). \\
&\quad (4.20)
\end{aligned}$$

Combining Equations 4.16 and 4.20 yields the result in Equation 4.11.  $\square$

### 4.3 Putting Everything Together

Now we can assemble the tools we have developed to prove our main result (Theorem 1.1). In Section 4.1 we connected near-optimal lower triangular Toeplitz matrix factorizations to approximating the square root function. In Sections 3.1 and 3.3 we connected low-degree rational functions to efficient streaming algorithms. In Section 4.2 we gave a low-degree rational function that approximates the square root function. The combination of these three tools gives a near-optimal matrix factorization with a corresponding efficient streaming algorithm.

**Theorem 4.6** (Main Theorem - Formal version of Theorem 1.1). *Let  $n, d, m \in \mathbb{N}$  satisfy  $n \geq 5$  and  $d \geq 2 + \left(\frac{12+4\log n}{\pi}\right)^2$ . There exists a lower triangular Toeplitz matrix factorization  $B, C, A = BC \in \mathbb{R}^{n \times n}$  with the following properties.*

- **Near-Optimality:** *Let  $B^* = C^* = M(f, n)$  for  $f(x) = 1/\sqrt{1-x}$  be the optimal lower triangular Toeplitz matrix factorization of size  $n$ . Then*

$$\begin{aligned}
\|B\|_{2 \rightarrow \infty} &\leq \|B^*\|_{2 \rightarrow \infty} + 16\sqrt{n} \exp\left(-\frac{\pi}{2}\sqrt{d-2}\right), \\
\|C\|_{1 \rightarrow 2} &\leq \|C^*\|_{1 \rightarrow 2} + 16\sqrt{n} \exp\left(-\frac{\pi}{2}\sqrt{d-2}\right).
\end{aligned}$$

- **Efficiency:** *There is an algorithm that takes as input a stream  $Z_{1,\cdot}, Z_{2,\cdot}, \dots, Z_{n,\cdot} \in \mathbb{R}^{1 \times m}$  and outputs a stream  $\hat{Z}_{1,\cdot}, \hat{Z}_{2,\cdot}, \dots, \hat{Z}_{n,\cdot} \in \mathbb{R}^{1 \times m}$  such that  $\hat{Z} = C^{-1}Z$  and the space and time (per iteration) required by the algorithm is  $O(dm)$ .*

To obtain Theorem 1.1 in the introduction from Theorem 4.6, we simply need to set  $d = O(\log(n/\mu))^2$  such that  $d \geq 2 + \left(\frac{12+4\log n}{\pi}\right)^2$  and  $16\sqrt{n} \exp\left(-\frac{\pi}{2}\sqrt{d-2}\right) \leq \frac{\mu}{4+\log(n)}$ .

*Proof.* Corollary 4.4 gives us a rational function  $r$  of degree  $d$  with real coefficients such that

$$|r(x) - \sqrt{1-x}| \leq 8 \cdot \exp\left(-\frac{\pi}{2}\sqrt{d-2}\right)$$

for all  $x \in \mathbb{C}$  with  $|x| \leq 1$ .

Let  $b(x) := \frac{r(x)}{1-x}$  and  $c(x) := \frac{1}{r(x)}$  and  $f(x) := \frac{1}{\sqrt{1-x}}$  and  $g(x) := 1/(1-x) = b(x)c(x)$ . Let  $B := M(b, n)$ ,  $C := M(c, n) \in \mathbb{R}^{n \times n}$  be as in Equation 3.1. By Lemma 3.1, this is a valid matrix factorization, as  $B \cdot C = M(b, n) \cdot M(c, n) = M(g, n)$ .

Let  $\tau > 0$  and

$$\gamma_\tau := \sup \left\{ |r(x) - \sqrt{1-x}| : x \in \mathbb{C}, |x| = \exp(-\tau) \right\} \leq 8 \cdot \exp\left(-\frac{\pi}{2}\sqrt{d-2}\right).$$

By Proposition 4.1, if  $\gamma_\tau \leq \left(\frac{1-\exp(-2\tau)}{4}\right)^2$ , then

$$\begin{aligned} \|M(b, n)\|_{2 \rightarrow \infty} &\leq \|M(f, n)\|_{2 \rightarrow \infty} + \frac{\exp(\tau n) \cdot \gamma_\tau}{\sqrt{\exp(2\tau) - 1}}, \\ \|M(c, n)\|_{1 \rightarrow 2} &\leq \|M(f, n)\|_{1 \rightarrow 2} + \frac{\exp(\tau n) \cdot \gamma_\tau}{\sqrt[4]{(\exp(2\tau) - 1)^2 - 2^{7/2}\gamma_\tau \exp(4\tau)}}. \end{aligned}$$

Set  $\tau = \frac{1}{2n}$ . Then  $\exp(2\tau) - 1 \geq 1/n$  and  $\exp(\tau n) = \exp(1/2)$ . Thus

$$\frac{\exp(\tau n) \cdot \gamma_\tau}{\sqrt{\exp(2\tau) - 1}} \leq \sqrt{n} \cdot \exp(1/2) \cdot \gamma_\tau < 2\sqrt{n}\gamma_\tau$$

and, if  $n \geq 5$  and  $\gamma_\tau \leq \left(\frac{1}{6n}\right)^2$ , then

$$\frac{\exp(\tau n) \cdot \gamma_\tau}{\sqrt[4]{(\exp(2\tau) - 1)^2 - 2^{7/2}\gamma_\tau \exp(4\tau)}} \leq \frac{\exp(1/2) \cdot \gamma_\tau}{\sqrt[4]{1/n^2 - 2^{7/2}\gamma_\tau \exp(2/n)}} \leq \frac{\exp(1/2)\gamma_\tau}{\sqrt[4]{\frac{1}{n^2} - \frac{2^{7/2}\exp(2/5)}{36n^2}}} < 2\sqrt{n}\gamma_\tau.$$

Assuming  $n \geq 5$  and  $d \geq 2 + \left(\frac{2}{\pi} \log(8 \cdot (6n)^2)\right)^2$ , we have  $\gamma_\tau \leq \left(\frac{1}{6n}\right)^2 \leq \left(\frac{1-\exp(-2\tau)}{4}\right)^2$ , which means all the above inequalities hold. Combining inequalities yields the first part of the result.

Per Remark 3.3, we can write

$$r(x) = t + \frac{p(x)}{q(x)}$$

where  $\deg(p) < d$  and  $\deg(q) \leq d$  and  $t \in \mathbb{R}$ . Lemma 3.2 states that there exist  $u, v \in \mathbb{R}^d$  and  $W \in \mathbb{R}^{d \times d}$  such that we can express

$$r(x) = \sum_{k=0}^{\infty} r_k x^k \quad \text{for} \quad r_k = u^T W^k v + t \mathbb{I}[k=0].$$

Furthermore  $W$  is sparse in the sense that each row has only two nonzero entries. Now we can feed  $u, v, W, t$  as parameters to Algorithm 1. Lemma 3.4 tells us that Algorithm 1 receives the rows of  $Z \in \mathbb{R}^{n \times m}$  as a stream and returns the rows of  $\hat{Z} \in \mathbb{R}^{n \times m}$  as a stream, where  $\hat{Z} = M(r, n)Z = C^{-1}Z$ . (The fact that  $M(r, n) = C^{-1}$  follows from Lemma 3.1.) The space required by the algorithm is  $O(dm)$  and the time per iteration is dominated by computing the matrix multiplication  $WS$  for  $S \in \mathbb{R}^{d \times m}$ . Since  $W$  is sparse, this takes  $O(dm)$  time.  $\square$

## 5 Factorizations via Direct Optimization

In this section, we begin (in §5.1) by showing the following result:

**Theorem 5.1** (Informal). *Let  $r$  be a rational generating function. Then, for the matrix factorization of  $A$  given by  $B = M(r(x)/(1-x), n)$  and  $C = M(1/r(x), n)$ , we can compute in “closed-form” (more specifically, in time  $\mathcal{O}(\text{poly}(d) \log n)$ ) the sensitivity and error of this matrix factorization, namely  $\|B\|_{2 \rightarrow \infty}^2$  and  $\|C\|_{1 \rightarrow 2}^2$ .*

*Proof.* The result follows from Lemmas 5.2 to 5.4.  $\square$

We remark that the  $\log n$  dependence in Theorem 5.1 comes (only) from the need to compute quantities like  $\theta^k$  for  $\theta \in \mathbb{R}$  and  $k \leq n$ , which can be done via iterated squaring. For practical  $n$  and modern hardware, this term can essentially be ignored, making the running time practically independent of  $n$ . If the roots of the numerator and denominator of  $r$  are already known (as assumed in Lemma 5.2) then  $\text{MaxErr}$  can be computed in time  $\mathcal{O}(d^2 \log(n))$ ; and such roots can always be found in time  $\mathcal{O}(\text{poly}(d))$  (omitting the dependence on the desired numerical accuracy).

In Section 5.2 we then show how Theorem 5.1 can be used to directly optimize the parameters of a BLT factorization to minimize  $\text{MaxErr}$  for a specific  $n$ .

### 5.1 Fast and Exact Error Computation

In addition to enabling efficient streaming multiplication, the rational function structure of the matrix factorizations we consider allows us to compute the total error of these mechanisms efficiently for any  $n$ .

**Lemma 5.2.** *Let  $r(x) = p(x)/q(x)$  be a rational function where  $\deg(p) = \deg(q) = d$  with  $q(0) = p(0) = 1$ . Further, suppose  $p$  and  $q$  have pairwise distinct real roots, in particular there exist  $\theta_i, \hat{\theta}_i \in (0, 1]$  for  $i \in [d]$  such that*

$$\begin{aligned} p(x) &= (1 - \hat{\theta}_0 x)(1 - \hat{\theta}_1 x) \cdots (1 - \hat{\theta}_{d-1} x), \\ q(x) &= (1 - \theta_0 x)(1 - \theta_1 x) \cdots (1 - \theta_{d-1} x) \end{aligned}$$

with  $\theta_i \neq \theta_j$  and  $\hat{\theta}_i \neq \hat{\theta}_j$  when  $i \neq j$ . Then, there exist  $\omega_i, \hat{\omega}_i \in \mathbb{R}$  for  $i \in [d]$  such that

$$r(x) = 1 + x \left( \sum_{i=0}^{d-1} \frac{\omega_i}{1 - \theta_i x} \right) \quad \text{and} \quad s(x) := \frac{1}{r(x)} = 1 + x \left( \sum_{i=0}^{d-1} \frac{\hat{\omega}_i}{1 - \hat{\theta}_i x} \right). \quad (5.1)$$

These  $\omega_i, \hat{\omega}_i$  can in fact be computed in closed form, Equation 5.3. Further, the coefficients of the corresponding sequences can be computed directly as

$$r_i = \begin{cases} 1 & i = 0 \\ \sum_{j \in [d]} \omega_j \theta_j^{i-1} & i > 0. \end{cases} \quad \text{and} \quad s_i = \begin{cases} 1 & i = 0 \\ \sum_{j \in [d]} \hat{\omega}_j \hat{\theta}_j^{i-1} & i > 0. \end{cases} \quad (5.2)$$

Before giving a proof, we observe that Equation 5.2 yields a diagonalized companion matrix of the form of Lemma 3.2, and using the representation of Remark 3.3 we can take  $t = 1 - \sum_{i=0}^{d-1} \omega_i / \theta_i$  so

$$r_k = t\mathbb{I}[k=0] + u^T W^k v := \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}^T \begin{bmatrix} \theta_0 & 0 & 0 & 0 \\ 0 & \theta_1 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & 0 & \dots & \theta_{d-1} \end{bmatrix}^k \begin{bmatrix} \omega_0 / \theta_0 \\ \omega_1 / \theta_1 \\ \vdots \\ \omega_{d-1} / \theta_{d-1} \end{bmatrix},$$

and hence we can efficiently multiply by  $M(r)$  using Algorithm 1; a similar construction with  $(\hat{\omega}, \hat{\theta})$  enables efficient multiplication by  $M(1/r)$  using only  $d$  buffers.

This construction is particularly useful for optimization, but assumes we already have factored  $p$  and  $q$ ; one can of course always find the roots of  $p$  and  $q$  if they are given in some other polynomial representation.<sup>23</sup> Specifically, if one is instead given the matrix-power representation of  $r$ , Proposition 5.6 of Section 5.3 shows these can directly be converted to a parameterization of  $1/r$ .

*Proof of Lemma 5.2.* Equation 5.2 follows from Equation 5.1 and the Taylor series

$$\frac{\omega x}{1 - \theta x} = \sum_{i=1}^{\infty} \omega \theta^{i-1} x^i.$$

It remains to show we can find  $\omega_i$  to instantiate Equation 5.1 for  $r(x)$  (the case of  $\hat{\omega}_i$  for  $s(x)$  is completely symmetric). Letting

$$q^{(-i)}(x) := \frac{q(x)}{1 - \theta_i x} = \prod_{\substack{j \in [d] \\ j \neq i}} (1 - \theta_j x),$$

we need  $\omega_i$  that satisfy

$$1 + x \left( \sum_{i=0}^{d-1} \frac{\omega_i}{1 - \theta_i x} \right) = \frac{q(x) + \sum_{i=0}^{d-1} \omega_i x q^{(-i)}(x)}{q(x)} = \frac{p(x)}{q(x)}.$$

Since  $p(0) = q(0) = 1$ ,  $p(x) - q(x)$  is divisible by  $x$ , so  $f(x) := \frac{1}{x}(p(x) - q(x))$  is a polynomial of degree  $d - 1$ . Hence, we wish to solve

$$\sum_{i=0}^{d-1} \omega_i q^{(-i)}(x) = f(x)$$

---

<sup>23</sup>While we only need to consider real roots in our constructions and our optimization, the results of this section go through for complex roots as well.

for  $\omega_0, \dots, \omega_{d-1}$ . Equating the polynomial coefficients, this is a linear system of  $d$  equations and  $d$  unknowns. It remains to show that this system has full-rank or, equivalently, that the set of polynomials  $\{q^{(-i)}\}_{i \in [d]}$  forms a basis for the vector space of polynomials of degree  $\leq d-1$ .

Let  $z := \prod_{i \in [d]} -\theta_i$  and  $w_i := \left( \prod_{j \neq i} (\theta_i^{-1} - \theta_j^{-1}) \right)^{-1}$  and define

$$\ell(x) := \frac{q(x)}{z} = (x - \theta_0^{-1})(x - \theta_1^{-1}) \cdots (x - \theta_{d-1}^{-1}) \quad \text{and} \quad \ell^{(-i)}(x) := \ell(x) \frac{w_i}{x - \theta_i^{-1}}.$$

Then  $\{\ell^{(-i)}\}_{i \in [d]}$  is exactly the Lagrange polynomial basis for interpolating points at  $\theta_0^{-1}, \dots, \theta_{d-1}^{-1}$ ; further, we have

$$q^{(-i)}(x) = \frac{q(x)}{-\theta_i(x - \theta_i^{-1})} = \frac{-z}{\theta_i w_i} \ell^{(-i)}(x),$$

and since the  $\theta_i$  are distinct and non-zero,  $\frac{-\theta_i z}{w_i}$  is non-zero. It follows from Lagrange interpolation that

$$f(x) = \sum_{i \in [d]} f(\theta_i^{-1}) \ell^{(-i)}(x) = \sum_{i \in [d]} f(\theta_i^{-1}) \frac{-\theta_i w_i}{z} q^{(-i)}(x)$$

and so

$$\omega_i = f(\theta_i^{-1}) \frac{-\theta_i w_i}{z}. \tag{5.3}$$

□

It will be useful to define the prefix sums of the geometric series by

$$\gamma_n(\theta) := 1 + \theta + \theta^2 + \cdots + \theta^{n-1} = \sum_{i=0}^{n-1} \theta^i.$$

with  $\gamma_\infty(\theta) = 1/(1 - \theta)$  and  $\gamma_0(\theta) = 0$ . When  $|\theta| < 1$ , we have of course

$$\gamma_n(\theta) = \sum_{i=0}^{n-1} \theta^i = \frac{1 - \theta^n}{1 - \theta}. \tag{5.4}$$

(This use of  $\gamma$  is unrelated to the  $\gamma_2(A)$  factorization norm discussed earlier.)

**Lemma 5.3.** *Let  $C^{(n)} = M(s(x), n)$  for a rational function*

$$s(x) = 1 + x \left( \sum_{i=0}^{d-1} \frac{\hat{\omega}_i}{1 - \hat{\theta}_i x} \right),$$

*for distinct  $\hat{\theta}_i$  as per the assumptions of Lemma 5.2. Then, there exists a closed form expression for the sensitivity of  $C^{(n)}$ ,  $\text{sens}(C^{(n)}) = \|C^{(n)}\|_{1 \rightarrow 2}$  allowing computation in time and space  $\mathcal{O}(d^2 \log n)$ .*

*Proof.* The squared sensitivity can be computed as

$$\begin{aligned}
\text{sens}^2(C) &= \sum_{i=0}^{n-1} s_i^2 = 1 + \sum_{i=0}^{n-2} \left( \sum_{j \in [d]} \hat{\omega}_j \hat{\theta}_j^i \right)^2 && \text{by Equation 5.2} \\
&= 1 + \sum_{i=0}^{n-2} \left( \sum_{j \in [d]} \sum_{k \in [d]} \hat{\omega}_j \hat{\omega}_k \hat{\theta}_j^i \hat{\theta}_k^i \right) \\
&= 1 + \sum_{j \in [d]} \sum_{k \in [d]} \left( \sum_{i=0}^{n-2} \hat{\omega}_j \hat{\omega}_k (\hat{\theta}_j \hat{\theta}_k)^i \right) \\
&= 1 + \sum_{j \in [d]} \sum_{k \in [d]} \hat{\omega}_j \hat{\omega}_k \gamma_{n-1}(\hat{\theta}_j \hat{\theta}_k),
\end{aligned}$$

and taking a square root completes the proof.  $\square$

**Lemma 5.4.** *Let  $B^{(n)} = A^{(n)}(C^{(n)})^{-1} = A^{(n)}M(r(x), n) = M(r(x)/(1-x), n)$  for a rational function*

$$r(x) = 1 + x \left( \sum_{i=0}^{d-1} \frac{\omega_i}{1 - \theta_i x} \right)$$

*for distinct  $\theta_i$  as per the assumptions of Lemma 5.2. Then, there exists a closed form expression for the variance induced in the  $n$ th prefix sum,  $\|B^{(n)}\|_{2 \rightarrow \infty}^2$  allowing computation in time and space  $\mathcal{O}(d^2 \log n)$ .*

*Proof.* In order to make our notation more compact, we will compute sensitivity for an arbitrary row index  $n$  of  $M(r(x)/(1-x), n+1)$ . Since multiplication by  $(1-x)^{-1}$  corresponds to taking prefix sums, we know  $B = M(t)$  where  $t_0 = r_0 = 1$  and for any  $n > 0$ ,

$$t_n := \sum_{i=0}^n r_i = 1 + \sum_{j \in [d]} \sum_{i=0}^{n-1} \omega_j \theta_j^i = 1 + \sum_{j \in [d]} \omega_j \gamma_n(\theta_j) \quad (5.5)$$

and we calculate

$$t_n^2 = \left( 1 + \sum_{j \in [d]} \omega_j \gamma_n(\theta_j) \right)^2 = 1 + 2 \sum_{j \in [d]} \omega_j \gamma_n(\theta_j) + \sum_{j \in [d]} \sum_{k \in [d]} \omega_j \gamma_n(\theta_j) \omega_k \gamma_n(\theta_k). \quad (5.6)$$

It is thus sufficient to compute

$$\begin{aligned}
\|B_{n,:}\|_2^2 &= \sum_{i=0}^{n-1} t_i^2 \\
&= 1 + \sum_{i=1}^{n-1} t_i^2 \\
&= 1 + \sum_{i=1}^{n-1} \left( 1 + 2 \sum_{j \in [d]} \omega_j \gamma_i(\theta_j) + \sum_{j \in [d]} \sum_{k \in [d]} \omega_j \gamma_i(\theta_j) \omega_k \gamma_i(\theta_k) \right).
\end{aligned} \quad (5.7)$$

where we used Equation 5.6 for the last step. Bringing the sum over  $i$  inside the sums over the buffers  $j$  and  $k$ , it will be sufficient to consider the following terms. For any  $j \in [b]$ ,

$$\begin{aligned}\Gamma_j &:= \sum_{i=1}^{n-1} \omega_j \gamma_i(\theta_j) = \sum_{i=1}^{n-1} \frac{\omega_j(1 - \theta_j^i)}{1 - \theta_j} \text{ assuming } 0 \leq \theta < 1 \\ &= \frac{\omega_j}{1 - \theta_j} \left( (n-1) - \sum_{i=1}^{n-1} \theta_j^i \right) \\ &= \frac{\omega_j}{1 - \theta_j} (n - \gamma_n(1, \theta_j))\end{aligned}\tag{5.8}$$

Similarly, for any  $j \in [d], k \in [d]$ , let

$$\begin{aligned}\Gamma_{j,k} &:= \sum_{i=1}^{n-1} \omega_j \gamma_i(\theta_j) \omega_k \gamma_i(\theta_k) \\ &= \sum_{i=1}^{n-1} \frac{\omega_j(1 - \theta_j^i)}{1 - \theta_j} \frac{\omega_k(1 - \theta_k^i)}{1 - \theta_k} \quad \text{since } 0 \leq \theta < 1 \\ &= \frac{\omega_j \omega_k}{(1 - \theta_j)(1 - \theta_k)} \sum_{i=1}^{n-1} (1 - \theta_j^i - \theta_k^i + (\theta_j \theta_k)^i) \\ &= \frac{\omega_j \omega_k}{(1 - \theta_j)(1 - \theta_k)} \sum_{i=0}^{n-1} (1 - \theta_j^i - \theta_k^i + (\theta_j \theta_k)^i) \\ &= \frac{\omega_j \omega_k}{(1 - \theta_j)(1 - \theta_k)} (n - \gamma_n(1, \theta_j) - \gamma_n(1, \theta_k) + \gamma_n(1, \theta_j \theta_k)).\end{aligned}\tag{5.9}$$

Putting everything together we have

$$\|B_{n,:}\|_2^2 = \sum_{i=0}^{n-1} t_i^2 = n + 2 \sum_{j \in [d]} \Gamma_j + \sum_{j \in [d]} \sum_{k \in [d]} \Gamma_{j,k}.\tag{5.10}$$

Since Equation 5.10 computes the  $L_2$ -norm squared for the row indexed by  $n$ , and this quantity is non-decreasing in  $n$ , it follows that

$$\|B^{(n)}\|_{2 \rightarrow \infty} = \sqrt{\|B_{n-1,:}\|_2^2},$$

which can be computed using Equation 5.10. □

**Remark 5.5.** Rearranging Equation 5.10 and in particular expanding the terms  $\Gamma_j$  (Equation 5.8) and  $\Gamma_{j,k}$  (Equation 5.9) reveals there exists constants  $\alpha_0, \alpha_1 \in \mathbb{R}$  and  $\tilde{\omega}_i \in \mathbb{R}, \tilde{\theta}_i \in (0, 1]$  for  $i \in [d + d^2]$  such that

$$\|B^{(n)}\|_{2 \rightarrow \infty} = \alpha_0 + \alpha_1 \cdot n + \sum_{i \in [d + d^2]} \tilde{\omega}_i \tilde{\theta}_i^{n-1}$$

where in particular

$$\alpha_1 = 1 + 2 \sum_{j \in [d]} \frac{\omega_j}{1 - \theta_j} + \sum_{j \in [d]} \sum_{k \in [d]} \frac{\omega_j \omega_k}{(1 - \theta_j)(1 - \theta_k)}.$$



We must have  $\alpha_1 \geq 0$  as for a valid factorization the norm must remain positive for all  $n$ , and  $\lim_{n \rightarrow \infty} \tilde{\omega}_i \tilde{\theta}^{n-1} = 0$  (or another constant  $\tilde{\omega}_i$  if  $\tilde{\theta}_i = 1$ ). For all the BLTs we consider (and we believe all useful BLTs), we find  $\alpha_1 > 0$ . Since sensitivity, Lemma 5.3, is non-decreasing in  $n$ , for any such BLT, eventually MaxErr must grow linearly. Hence, one must either increase  $d$  with  $n$  or optimize the BLT for a specific  $n$  in order to hope for good performance. In particular, by optimizing for a specific  $n$  (§5.2), we must keep  $\alpha$  small enough that the  $\alpha_1 n$  term remains relatively small (say, of the same order as  $\alpha_0$ ). However, for larger  $n$  this term can quickly explode; we see exactly this behavior in Figure 3 (Left column).

## 5.2 Direct Optimization of BLTs

Inspection of Lemmas 5.2 to 5.4 reveals that the function  $(\theta, \hat{\theta}) \rightarrow \text{MaxErr}$  is in fact differentiable. With this result, together with the automatic differentiation capabilities of JAX [Bra+18], we can in fact directly optimize for rational functions of a given degree  $d$  which minimize MaxErr for a given number of steps  $n$ .

Some care is needed in the implementation. In particular, we find:

- Using `float64` precision is necessary, particularly when optimizing for  $n > 10^6$ .
- As  $n$  becomes large,  $\max_{i \in [d]} \theta_i$  tends to 1. Eqs. (5.4), (5.8) and (5.9) can be numerically unstable in such situations; this can be remedied by switching to a Taylor-series approximation taken around  $\theta = 1$  for  $\theta$  sufficiently near (or equal to) 1.
- The optimization is more stable (particularly for larger  $n$ ) if we add a barrier function

$$\ell(\theta, \omega) = 10^{-7} \sum_{i \in [d]} -\log(\theta_i) - \log(\omega_i)$$

to ensure these parameters remain strictly positive.

- For large  $n$  (e.g  $n > 10^7$ ), larger numbers of buffers  $d$  can make the optimization less stable. For example, with our current implementation we find for  $n = 10^8$ , optimizing for more than  $d = 6$  buffers actually slightly decreases performance (while theoretically it should only help).

With the above setup, we find the L-BFGS optimizer works well. Convergence generally takes less than a second running on CPUs on a modern workstation<sup>24</sup>, even from a naive initialization (initialization from Section 4 would likely require even fewer iterations). Numerical results for these Opt-BLT mechanisms are given in Figures 1 and 3.

Proving the convergence properties of this approach and deriving more robust optimization algorithms are interesting directions for future work.

---

<sup>24</sup>There is approximately 5-15 seconds of overhead for JAX to just-in-time compile the loss function; this is only incurred once even if mechanisms are optimized for many different  $n$ .

### 5.3 Derivation of the Inverse BLT Parameterization

We now show that the matrix-power representation of a rational function  $r(x)$  (see Lemma 3.2) can be converted to a matrix-power representation for its reciprocal  $1/r(x)$ . This is useful for our algorithm. Note that the dimension of the representation increases by one.

**Proposition 5.6** (Representation of the reciprocal of a rational generating function). *Let  $u, v \in \mathbb{R}^d$  and  $W \in \mathbb{R}^{d \times d}$ . Assume  $\langle u, v \rangle = 1$ .<sup>25</sup> Define  $r(x) = \sum_{k=0}^{\infty} u^T W^k v \cdot x^k$ . Then*

$$\frac{1}{r(x)} = 1 - \sum_{\ell=1}^{\infty} u^T W (W - vu^T W)^{\ell-1} v \cdot x^{\ell} \quad (5.11)$$

$$= \sum_{k=0}^{\infty} \tilde{u}^T \tilde{W}_0^k \tilde{v} \cdot x^k \quad (5.12)$$

and

$$\frac{1}{r(x) \cdot (1-x)} = \sum_{k=0}^{\infty} \tilde{u}^T \tilde{W}_1^k \tilde{v} \cdot x^k \quad (5.13)$$

where, for both  $\beta \in \{0, 1\}$ ,

$$\tilde{W}_{\beta} = \begin{pmatrix} \beta & 0 \cdots 0 \\ v & W - vu^T W \end{pmatrix} \in \mathbb{R}^{(d+1) \times (d+1)}, \quad (5.14)$$

$$\tilde{u} = \begin{pmatrix} 1 \\ -W^T u \end{pmatrix} \in \mathbb{R}^{d+1}, \quad (5.15)$$

$$\tilde{v} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \in \mathbb{R}^{d+1}. \quad (5.16)$$

*Proof.* To prove the first part of the result (5.11) it suffices to show that

$$r(x) \cdot \left( 1 - \sum_{\ell=1}^{\infty} u^T W (W - vu^T W)^{\ell-1} v \cdot x^{\ell} \right) = 1.$$

---

<sup>25</sup>This assumption is only made for simplicity and holds without loss of generality, as we can always rescale the function by  $r(0) = u^T v$ : That is, let  $\hat{r}(x) := r(x)/r(0)$ , apply the result, and rescale back  $1/r(x) = r(0) \cdot 1/\hat{r}(x)$ .

We have

$$\begin{aligned}
& r(x) \cdot \left( 1 - \sum_{\ell=1}^{\infty} u^T W (W - vu^T W)^{\ell-1} v \cdot x^{\ell} \right) \\
&= \sum_{k=0}^{\infty} u^T W^k v \cdot x^k \cdot \left( 1 - \sum_{\ell=1}^{\infty} u^T W (W - vu^T W)^{\ell-1} v \cdot x^{\ell} \right) \\
&= \sum_{i=0}^{\infty} \left( u^T W^i v - \sum_{\ell=1}^i u^T W^{i-\ell} v \cdot u^T W (W - vu^T W)^{\ell-1} v \right) \cdot x^i \\
&= \sum_{i=0}^{\infty} \left( u^T W^i v - u^T W (W - vu^T W)^{i-1} v - \sum_{\ell=1}^{i-1} u^T W^{i-\ell} v \cdot u^T W (W - vu^T W)^{\ell-1} v \right) \cdot x^i \\
&= u^T v + \sum_{i=1}^{\infty} u^T \left( W^i - W (W - vu^T W)^{i-1} - \sum_{\ell=1}^{i-1} W^{i-\ell} v \cdot u^T W (W - vu^T W)^{\ell-1} \right) v \cdot x^i \\
&= 1 + \sum_{i=1}^{\infty} u^T \left( W^{i-1} - (W - Wvu^T)^{i-1} - \sum_{\ell=1}^{i-1} W^{i-\ell} v \cdot u^T (W - Wvu^T)^{\ell-1} \right) Wv \cdot x^i,
\end{aligned}$$

where the final equality uses the fact

$$W(W - vu^T W)^{\ell-1} = W((I - vu^T)W)^{\ell-1} = (W(I - vu^T))^{\ell-1} W = (W - Wvu^T)^{\ell-1} W.$$

Now it suffices to show that, for all  $i \geq 1$ , we have

$$W^{i-1} = (W - Wvu^T)^{i-1} + \sum_{\ell=1}^{i-1} W^{i-\ell} v \cdot u^T (W - Wvu^T)^{\ell-1}. \quad (5.17)$$

We prove Equation 5.17 by induction on  $i$ . For  $i = 1$ , the equation is trivially true (both sides are the identity). Assuming Equation 5.17 holds for a given  $i \geq 1$ , we have

$$\begin{aligned}
W^i &= W \cdot W^{i-1} = ((W - Wvu^T) + Wvu^T) \cdot W^i \\
&= ((W - Wvu^T) + Wvu^T) \cdot (W - Wvu^T)^{i-1} + W \cdot \sum_{\ell=1}^{i-1} W^{i-\ell} v \cdot u^T (W - Wvu^T)^{\ell-1} \\
&= (W - Wvu^T)^i + Wvu^T \cdot (W - Wvu^T)^{i-1} + \sum_{\ell=1}^{i-1} W^{i+1-\ell} v \cdot u^T (W - Wvu^T)^{\ell-1} \\
&= (W - Wvu^T)^i + \sum_{\ell=1}^i W^{i+1-\ell} v \cdot u^T (W - Wvu^T)^{\ell-1} \\
&\quad - W^{i+1-i} v \cdot u^T (W - Wvu^T)^{i-1} + Wvu^T \cdot (W - Wvu^T)^{i-1} \\
&= (W - Wvu^T)^i + \sum_{\ell=1}^i W^{i+1-\ell} v \cdot u^T (W - Wvu^T)^{\ell-1},
\end{aligned}$$

which establishes that Equation 5.17 holds for the next value of  $i$ .

To prove Equation 5.12, we must show that  $\tilde{u}^T \widetilde{W}_0^0 \tilde{v} = 1$  and  $\tilde{u}^T \widetilde{W}_0^k \tilde{v} = -u^T W(W - vu^T W)^{k-1} v$  for  $k \geq 1$ . We immediately have  $\tilde{u}^T \tilde{v} = 1$ . By induction we can show that, for all  $k \geq 1$ , we have

$$\widetilde{W}_0^k = \begin{pmatrix} 0 & 0 \cdots 0 \\ (W - vu^T W)^{k-1} v & (W - vu^T W)^k \end{pmatrix} \in \mathbb{R}^{(d+1) \times (d+1)}.$$

From this it follows that  $\tilde{u}^T \widetilde{W}_0^k \tilde{v} = -u^T W(W - vu^T W)^{k-1} v$  for all  $k \geq 1$ , as required.

More generally, for all  $\beta \in \mathbb{R}$  and  $k \geq 1$ ,

$$\widetilde{W}_\beta^k = \begin{pmatrix} \beta^k & 0 \cdots 0 \\ \sum_{\ell=0}^{k-1} \beta^{k-1-\ell} (W - vu^T W)^\ell v & (W - vu^T W)^k \end{pmatrix} \in \mathbb{R}^{(d+1) \times (d+1)}.$$

Setting  $\beta = 1$  still gives  $\tilde{u}^T \widetilde{W}_1^0 \tilde{v} = 1$  and now

$$\tilde{u}^T \widetilde{W}_1^k \tilde{v} = 1 - \sum_{\ell=0}^{k-1} u^T W(W - vu^T W)^\ell v = 1 + \sum_{\ell=0}^{k-1} \tilde{u}^T \widetilde{W}_0^{\ell+1} \tilde{v} = \sum_{j=0}^k \tilde{u}^T \widetilde{W}_0^j \tilde{v}$$

for  $k \geq 1$ .

We have  $\frac{1}{r(x)} = \sum_{k=0}^{\infty} \tilde{u}^T \widetilde{W}_0^k \tilde{v} \cdot x^k$  and, hence,

$$\frac{1}{r(x) \cdot (1-x)} = \left( \sum_{k=0}^{\infty} \tilde{u}^T \widetilde{W}_0^k \tilde{v} \cdot x^k \right) \cdot \left( \sum_{i=0}^{\infty} x^i \right) = \sum_{\ell=0}^{\infty} x^\ell \sum_{j=0}^{\ell} \tilde{u}^T \widetilde{W}_0^j \tilde{v} = \sum_{\ell=0}^{\infty} \tilde{u}^T \widetilde{W}_1^\ell \tilde{v} \cdot x^\ell,$$

which completes the proof. □

## 6 Generalizing the Binary Tree Approach

In this section we prove Theorem 1.2. Compared to Theorem 1.1 this attains a weaker approximation to the optimal matrix factorization, but is asymptotically better in terms of space usage.

Our starting point is the binary tree mechanism of Dwork, Naor, Pitassi, and Rothblum [DNPR10] and Chan, Shi, and Song [CSS11]. The binary tree mechanism can be viewed as a recursive construction of a matrix factorization; see Equation 2.5 for a precise statement. A recursion of depth  $\ell$  yields a matrix factorization of size  $n = 2^\ell$  and an algorithm running in time and space  $O(\ell)$ . The binary tree mechanism is within a constant factor of optimal; we will improve the constant factor to be arbitrarily close to optimal. Our approach is to combine a recursive construction with the factorization we already developed to prove Theorem 1.1. We will start with a matrix factorization of size  $n_1$  and, after a recursion of depth  $\ell$ , we obtain a matrix factorization of size  $n_1^\ell$ . The space required is  $O(\ell \cdot \log^2 n_1)$ . Appropriately setting the parameters yields Theorem 1.2.

First we define the non-cyclic shift matrix  $S^{(n)} \in \{0, 1\}^{n \times n}$  by  $S_{i,j} = 1 \iff i = j + 1$  for all  $i, j \in [n]$ . If we multiply the non-cyclic shift matrix with the lower-triangular all-ones matrix, it has the effect of zeroing out the diagonal and producing a strictly lower-triangular matrix. For

example,

$$S^{(6)} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \quad \text{and} \quad A^{(6)} S^{(6)} = S^{(6)} A^{(6)} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

The non-cyclic shift matrix corresponds to the generating function  $h(x) = x$  – i.e.,  $S^{(n)} = M(h, n)$ .

Given matrices  $M \in \mathbb{R}^{n \times m}$  and  $M' \in \mathbb{R}^{n' \times m'}$ , define the Kronecker product

$$M \otimes M' = \begin{pmatrix} M_{0,0} \cdot M' & M_{0,1} \cdot M' & \cdots & M_{0,m-1} \cdot M' \\ M_{1,0} \cdot M' & M_{1,1} \cdot M' & \cdots & M_{1,m-1} \cdot M' \\ \vdots & \vdots & \ddots & \vdots \\ M_{n-1,0} \cdot M' & M_{n-1,1} \cdot M' & \cdots & M_{n-1,m-1} \cdot M' \end{pmatrix} \in \mathbb{R}^{(n \cdot n') \times (m \cdot m')} \quad (6.1)$$

by  $(M \otimes M')_{i,j} = M_{i_1,j_1} \cdot M'_{i_2,j_2}$  where  $i = i_1 \cdot n' + i_2 \in [n \cdot n']$  and  $j = j_1 \cdot m' + j_2 \in [m \cdot m']$ . The Kronecker product has the “mixed-product property” with matrix multiplication: We have  $(A \otimes B)(A' \otimes B') = (AA') \otimes (BB')$  whenever the matrix dimensions match so that  $AA'$  and  $BB'$  are well-defined.

## 6.1 Recursive Matrix Factorization

Without further ado, we present the basis of our recursive construction. Suppose we are given two matrix factorizations  $A^{(n_1)} = B_1 C_1$  and  $A^{(n_2)} = B_2 C_2$ . We will combine these into one matrix factorization for  $A^{(n_1 \cdot n_2)}$ .

**Definition 6.1** (Combining Matrix Factorizations). *For  $B_1 \in \mathbb{R}^{n_1 \times n'_1}$  and  $B_2 \in \mathbb{R}^{n_2 \times n'_2}$ , define*

$$B_1 \mathbin{\textcircled{B}} B_2 := \left( \underbrace{I^{(n_1)} \otimes B_2}_{\in \mathbb{R}^{(n_1 n_2) \times (n_1 n'_2)}} \underbrace{S^{(n_1)} B_1 \otimes \mathbf{1}^{(n_2)}}_{\in \mathbb{R}^{(n_1 n_2) \times n'_1}} \right) \in \mathbb{R}^{(n_1 n_2) \times (n_1 n'_2 + n'_1)}, \quad (6.2)$$

where  $I^{(n_1)}$  is the  $n_1 \times n_1$  identity matrix and  $\mathbf{1}^{(n_2)}$  is the all-ones column vector of length  $n_2$ . For  $C_1 \in \mathbb{R}^{n'_1 \times n_1}$  and  $C_2 \in \mathbb{R}^{n'_2 \times n_2}$ , define

$$C_1 \mathbin{\textcircled{C}} C_2 := \begin{pmatrix} I^{(n_1)} \otimes C_2 \\ C_1 \otimes (\mathbf{1}^{(n_2)})^T \end{pmatrix} \in \mathbb{R}^{(n_1 n'_2 + n'_1) \times (n_1 n_2)}, \quad (6.3)$$

where  $I^{(n_1)}$  is the  $n_1 \times n_1$  identity matrix and  $(\mathbf{1}^{(n_2)})^T$  is the all-ones row vector of length  $n_2$ .

**Lemma 6.2** (Properties of Definition 6.1). *Let  $B_1, C_1^T \in \mathbb{R}^{n_1 \times n'_1}$  and  $B_2, C_2^T \in \mathbb{R}^{n_2 \times n'_2}$ . Suppose  $B_1 C_1 = A^{(n_1)}$  and  $B_2 C_2 = A^{(n_2)}$ , where  $A^{(n)}$  is the all-ones lower triangular matrix defined in Equation 2.1. Then*

$$(B_1 \mathbin{\textcircled{B}} B_2)(C_1 \mathbin{\textcircled{C}} C_2) = A^{(n_1 n_2)}.$$

Furthermore,

$$\begin{aligned} \|B_1 \mathbin{\textcircled{B}} B_2\|_{2 \rightarrow \infty}^2 &= \|S^{(n_1)} B_1\|_{2 \rightarrow \infty}^2 + \|B_2\|_{2 \rightarrow \infty}^2 \leq \|B_1\|_{2 \rightarrow \infty}^2 + \|B_2\|_{2 \rightarrow \infty}^2, \\ \|C_1 \mathbin{\textcircled{C}} C_2\|_{1 \rightarrow 2}^2 &= \|C_1\|_{1 \rightarrow 2}^2 + \|C_2\|_{1 \rightarrow 2}^2. \end{aligned}$$

*Proof.* We have

$$\begin{aligned}
(B_1 \mathbin{\textcircled{B}} B_2)(C_1 \mathbin{\textcircled{C}} C_2) &= (I^{(n_1)} \otimes B_2 \ S^{(n_1)} B_1 \otimes \mathbf{1}^{(n_2)}) \begin{pmatrix} I^{(n_1)} \otimes C_2 \\ C_1 \otimes (\mathbf{1}^{(n_2)})^T \end{pmatrix} \\
&= (I^{(n_1)} \otimes B_2)(I^{(n_1)} \otimes C_2) + (S^{(n_1)} B_1 \otimes \mathbf{1}^{(n_2)})(C_1 \otimes (\mathbf{1}^{(n_2)})^T) \\
&= (I^{(n_1)} I^{(n_1)}) \otimes (B_2 C_2) + (S^{(n_1)} B_1 C_1) \otimes (\mathbf{1}^{(n_2)} (\mathbf{1}^{(n_2)})^T) \\
&= I^{(n_1)} \otimes A^{(n_2)} + (S^{(n_1)} A^{(n_2)}) \otimes (\mathbf{1}^{(n_2)} (\mathbf{1}^{(n_2)})^T) \\
&= A^{(n_1 n_2)}.
\end{aligned}$$

The last equality is best understood by looking at an example for  $n_1 = 3$  and  $n_2 = 2$ :<sup>26</sup>

$$\begin{aligned}
&I^{(3)} \otimes A^{(2)} + (S^{(3)} A^{(3)}) \otimes (\mathbf{1}^{(2)} (\mathbf{1}^{(2)})^T) \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \\
&= A^{(6)}.
\end{aligned}$$

---

<sup>26</sup>If you don't like this proof by example, imagine some ellipses ( $\cdots, \dot{\cdot}, \dot{\cdot}$ ) inserted into the matrices and it will look like a general proof. Formally, we can do an index-by-index case analysis.

Now we have

$$\begin{aligned}
\|B_1 \mathbin{\textcircled{B}} B_2\|_{2 \rightarrow \infty}^2 &= \max_{i \in [n_1 n_2]} \sum_{j \in [n_1 n'_2 + n'_1]} (B_1 \mathbin{\textcircled{B}} B_2)_{i,j}^2 \\
&= \max_{i \in [n_1 n_2]} \sum_{j' \in [n_1 n'_2]} (I^{(n_1)} \otimes B_2)_{i,j'}^2 + \sum_{j'' \in [n_1]} (S^{(n_1)} B_1 \otimes \mathbf{1}^{(n_2)})_{i,j''}^2 \\
&= \max_{i_1 \in [n_1], i_2 \in [n_2]} \sum_{j'_1 \in [n_1], j'_2 \in [n'_2]} (I_{i_1, j'_1}^{(n_1)} \cdot (B_2)_{i_2, j'_2})^2 + \sum_{j'' \in [n_1]} ((S^{(n_1)} B_1)_{i_1, j''} \cdot \mathbf{1}^{(n_2)})_{j''}^2 \\
&= \max_{i_1 \in [n_1], i_2 \in [n_2]} \sum_{j'_2 \in [n'_2]} (B_2)_{i_2, j'_2}^2 + \sum_{j'' \in [n_1]} (S^{(n_1)} B_1)_{i_1, j''}^2 \\
&= \max_{i_1 \in [n_1]} \sum_{j'_2 \in [n'_2]} (B_2)_{i_2, j'_2}^2 + \max_{i_2 \in [n_2]} \sum_{j'' \in [n_1]} (S^{(n_1)} B_1)_{i_1, j''}^2 \\
&= \|B_2\|_{2 \rightarrow \infty}^2 + \|S^{(n_1)} B_1\|_{2 \rightarrow \infty}^2.
\end{aligned}$$

Similarly,  $\|C_1 \mathbin{\textcircled{C}} C_2\|_{1 \rightarrow 2}^2 = \|C_1\|_{1 \rightarrow 2}^2 + \|C_2\|_{1 \rightarrow 2}^2$ . Finally,

$$\|S^{(n_1)} B_1\|_{2 \rightarrow \infty}^2 = \max_{i \in [n_1]} \sum_{j \in [n'_1]} (S^{(n_1)} B_1)_{i,j}^2 = \max_{i \in [n_1-1]} \sum_{j \in [n'_1]} (B_1)_{i,j}^2 \leq \|B_1\|_{2 \rightarrow \infty}^2.$$

□

Lemma 6.2 can be applied recursively. We start with some base factorization  $B_1 C_1 = A^{(n_1)}$  and inductively define a factorization satisfying  $B_\ell C_\ell = A^{(n_\ell)}$  and

$$\|B_\ell\|_{2 \rightarrow \infty} \cdot \|C_\ell\|_{1 \rightarrow 2} \leq \ell \cdot \|B_1\|_{2 \rightarrow \infty} \cdot \|C_1\|_{1 \rightarrow 2}.$$

**Proposition 6.3.** Let  $B_1, C_1^T \in \mathbb{R}^{n_1 \times n'_1}$ . Following Definition 6.1, for  $\ell \geq 2$ , define

$$B_\ell = B_1 \mathbin{\textcircled{B}} B_{\ell-1} \quad \text{and} \quad C_\ell = C_1 \mathbin{\textcircled{C}} C_{\ell-1}. \tag{6.4}$$

Then, for all  $\ell \geq 1$ , we have  $B_\ell, C_\ell^T \in \mathbb{R}^{n_\ell \times n'_\ell}$ , where  $n_\ell := n_1^\ell$  and

$$n'_\ell := n'_1 \cdot \sum_{k=0}^{\ell-1} n_1^k = n'_1 \cdot \frac{n_1^\ell - 1}{n_1 - 1}$$

If  $B_1 C_1 = A^{(n_1)}$ , then  $B_\ell C_\ell = A^{(n_\ell)}$  for all  $\ell \geq 1$ . Furthermore,

$$\begin{aligned}
\|B_\ell\|_{2 \rightarrow \infty} &\leq \sqrt{\ell} \cdot \|B_1\|_{2 \rightarrow \infty}, \\
\|C_\ell\|_{1 \rightarrow 2} &= \sqrt{\ell} \cdot \|C_1\|_{1 \rightarrow 2}.
\end{aligned}$$

*Proof.* Perform induction on  $\ell$  using Lemma 6.2 with  $B_2 = B_{\ell-1}$  and  $C_2 = C_{\ell-1}$ . □

## 6.2 Recursive Algorithm

Now we show that the recursive matrix factorization in Proposition 6.3 corresponds to an efficient algorithm. The recursive construction naturally corresponds to a recursive algorithm. That is, for streaming multiplication by  $B_\ell$  we must call a base algorithm for  $B_1$  and recursively call the algorithm for  $B_{\ell-1}$ .

Recall the streaming multiplication setting: We receive the rows of  $Z \in \mathbb{R}^{n'_\ell \times m}$  one by one as a stream and we must output the rows of  $B_\ell \cdot Z \in \mathbb{R}^{n_\ell \times m}$  as a stream. Note that our matrix  $B_\ell$  is not a lower triangular matrix; in fact, it is not even square. This makes the problem setting not entirely straightforward, as we do not have a one-to-one correspondence between streaming inputs and outputs.

In our application, the input  $Z$  is independent random noise. In particular, the order of the rows is unimportant. The time when we read each row is also not important as the noise can be generated as needed. The constraint is simply that we can only read each row once. (We can store the rows for later use, but that requires memory, so we want to avoid this.) In other words, the input is a stream of independent random noise, which we can read at any time, but we cannot “rewind” it.

---

### Algorithm 2 Recursive Streaming Algorithm corresponding to Proposition 6.3

---

**Parameters:** Base streaming algorithm  $\mathcal{A}_1$ . Recursion depth  $\ell \geq 1$ .

**Streaming Input:** Row vectors  $Z_{0,\cdot}, Z_{1,\cdot}, \dots, Z_{n'_\ell-1,\cdot} \in \mathbb{R}^{1 \times m}$ .

**Streaming Output:** Row vectors  $\tilde{Z}_{0,\cdot}, \tilde{Z}_{1,\cdot}, \dots, \tilde{Z}_{n_\ell-1,\cdot} \in \mathbb{R}^{1 \times m}$  such that  $\tilde{Z} = B_\ell \cdot Z$ .

**if**  $\ell = 1$  **then**

    Run  $\mathcal{A}_1$  and directly output what it outputs.

**else**

    Start a copy of  $\mathcal{A}_1$ . (Copy I)

    Let  $z'_0 = (\mathbf{0}^{(m)})^T \in \mathbb{R}^{1 \times m}$ .

**for**  $i = 0 \dots n_1 - 1$  **do**

        Start a copy of  $\mathcal{A}_{\ell-1}$ . (Copy II)

        ▷ Recursion  $\ell \mapsto \ell - 1$ .

**for**  $j = 0 \dots n_{\ell-1} - 1$  **do**

            Get  $z''_{i,j} \in \mathbb{R}^{1 \times m}$  the next output of  $\mathcal{A}_{\ell-1}$  (copy II).

            Output  $\tilde{Z}_{n_{\ell-1}i+j+1,\cdot} = z'_i + z''_{i,j} \in \mathbb{R}^{1 \times m}$ .

            Delete  $z''_{i,j}$ .

        Terminate  $\mathcal{A}_{\ell-1}$  (copy II) and delete  $z'_i$ .

        ▷ Free up memory.

        Get  $z'_{i+1} \in \mathbb{R}^{1 \times m}$  the next output of  $\mathcal{A}_1$  (copy I).

    Terminate  $\mathcal{A}_1$  (copy I).

---

Our recursive streaming algorithm is presented as Algorithm 2. We will instantiate the base streaming algorithm  $\mathcal{A}_1$  with Algorithm 1. The recursive algorithm runs many copies of  $\mathcal{A}_1$ . The streaming input is simply split among these copies. For simplicity, we do not belabor the precise order in which the input stream is read.

**Lemma 6.4** (Properties of Algorithm 2). *For  $\ell \geq 1$ , let  $B_\ell, C_\ell^T \in \mathbb{R}^{n_\ell \times n'_\ell}$  be as in Proposition 6.3. Let  $\mathcal{A}_1$  be a streaming algorithm that takes as input a stream  $Z_{0,\cdot}, Z_{1,\cdot}, \dots, Z_{n'_\ell-1,\cdot} \in \mathbb{R}^{1 \times m}$  and*



outputs a stream  $\widehat{Z}_{0,\cdot}, \widehat{Z}_{1,\cdot}, \dots, \widehat{Z}_{n_{\ell-1}-1,\cdot} \in \mathbb{R}^{1 \times m}$  such that  $\widehat{Z} = B_1 Z$ . Let  $\mathcal{A}_\ell$  denote Algorithm 2 instantiated with the base algorithm  $\mathcal{A}_1$  and recursion depth  $\ell \geq 1$ .

Then  $\mathcal{A}_\ell$  is a streaming algorithm that takes as input a stream  $Z_{0,\cdot}, Z_{1,\cdot}, \dots, Z_{n_{\ell-1}-1,\cdot} \in \mathbb{R}^{1 \times m}$  (in an arbitrary – but fixed – order) and outputs a stream  $\widetilde{Z}_{0,\cdot}, \widetilde{Z}_{1,\cdot}, \dots, \widetilde{Z}_{n_{\ell-1}-1,\cdot} \in \mathbb{R}^{1 \times m}$  such that  $\widetilde{Z} = B_1 Z$ .

Moreover, the space required by  $\mathcal{A}_\ell$  is at most  $\ell$  times the space required by  $\mathcal{A}_1$ . And the time required by  $\mathcal{A}_\ell$  is  $\frac{n_1^{\ell+1}-1}{n_1-1} \leq O(n_\ell)$  times the time required by  $\mathcal{A}_1$ .

*Proof.* We may inductively assume that the lemma holds for  $\ell - 1$  in place of  $\ell$ . That is, we assume  $\mathcal{A}_{\ell-1}$  is a streaming algorithm that takes as input a stream  $Z_{0,\cdot}, Z_{1,\cdot}, \dots, Z_{n_{\ell-1}-1,\cdot} \in \mathbb{R}^{1 \times m}$  (in an arbitrary – but fixed – order) and outputs a stream  $\check{Z}_{0,\cdot}, \check{Z}_{1,\cdot}, \dots, \check{Z}_{n_{\ell-1}-1,\cdot} \in \mathbb{R}^{1 \times m}$  such that  $\check{Z} = B_{\ell-1} Z$ . Let

$$Z' = \begin{pmatrix} z'_0 \\ z'_0 \\ \vdots \\ z'_0 \\ z'_1 \\ z'_1 \\ \vdots \\ z'_1 \\ \vdots \\ z'_{n_1-1} \\ z'_{n_1-1} \\ \vdots \\ z'_{n_1-1} \end{pmatrix} = \begin{pmatrix} z'_0 \\ z'_1 \\ \vdots \\ z'_{n_1-1} \end{pmatrix} \otimes \mathbf{1}^{(n_{\ell-1})} \in \mathbb{R}^{n_1 n_{\ell-1} \times m}$$

and

$$Z'' = \begin{pmatrix} z''_{0,0} \\ z''_{0,1} \\ \vdots \\ z''_{0,n_{\ell-1}} \\ z''_{1,0} \\ z''_{1,1} \\ \vdots \\ z''_{1,n_{\ell-1}} \\ \vdots \\ z''_{n_1-1,0} \\ z''_{n_1-1,1} \\ \vdots \\ z''_{n_1-1,n_{\ell-1}} \end{pmatrix} \in \mathbb{R}^{n_1 n_{\ell-1} \times m}.$$

We have  $\tilde{Z} = Z' + Z''$ . We want to show that

$$\tilde{Z} = B_\ell Z = (B_1 \textcircled{+} B_\ell) Z = (I^{(n_1)} \otimes B_2 \ S^{(n_1)} B_1 \otimes \mathbf{1}^{(n_2)}) \begin{pmatrix} Z_{0:n_1 n'_{\ell-1}, \cdot} \\ Z_{n_1 n'_{\ell-1}:n'_\ell, \cdot} \end{pmatrix},$$

where  $Z_{n_1 n'_{\ell-1}:n'_\ell, \cdot} \in \mathbb{R}^{n'_\ell \times m}$  and  $Z_{0:n_1 n'_{\ell-1}, \cdot} \in \mathbb{R}^{n_1 n'_{\ell-1} \times m}$  are a partition of the rows of  $Z \in \mathbb{R}^{n'_\ell \times m}$ . (Note that  $n'_\ell = n_1 n'_{\ell-1} + n'_1$ . The notation  $Z_{i:j, \cdot}$  denotes the submatrix of  $Z$  formed by rows  $i, i+1, \dots, j-1$  and all columns.) Thus it suffices to show that  $Z' = (S^{(n_1)} B_1 \otimes \mathbf{1}^{(n_2)}) Z_{n_1 n'_{\ell-1}:n'_\ell, \cdot}$  and  $Z'' = (I^{(n_1)} \otimes B_2) Z_{0:n_1 n'_{\ell-1}, \cdot}$ . By our inductive assumption on  $\mathcal{A}_{\ell-1}$ , for all  $i \in [n_1]$ ,

$$\begin{pmatrix} z''_{i,0} \\ z''_{i,1} \\ \vdots \\ z''_{i,n_{\ell-1}} \end{pmatrix} = B_\ell \cdot Z_{i n'_{\ell-1}:(i+1)n'_{\ell-1}, \cdot}$$

Thus

$$Z'' = \begin{pmatrix} B_\ell \cdot Z_{0:n'_{\ell-1}, \cdot} \\ B_\ell \cdot Z_{n'_{\ell-1}:2n'_{\ell-1}, \cdot} \\ \vdots \\ B_\ell \cdot Z_{(n_1-1)n'_{\ell-1}:n_1 n'_{\ell-1}, \cdot} \end{pmatrix} = (I^{(n_1)} \otimes B_{\ell-1}) \cdot Z_{0:n_1 n_{\ell-1}, \cdot},$$

as required. By our assumption about the base algorithm  $\mathcal{A}_1$ , we have

$$\begin{pmatrix} z'_1 \\ z'_2 \\ \vdots \\ z'_{n_1} \end{pmatrix} = B_1 \cdot Z_{n_1 n'_{\ell-1}:n'_\ell, \cdot} \in \mathbb{R}^{n_1 \times m}.$$

(Note that  $z'_{n_1}$  is never actually used in the algorithm.) With  $z'_0 = (\mathbf{0}^{(m)})^T$ , the non-cyclic shift  $S^{(n_1)}$  gives

$$\begin{pmatrix} z'_0 \\ z'_1 \\ \vdots \\ z'_{n_1-1} \end{pmatrix} = S^{(n_1)} \cdot B_1 \cdot Z_{n_1 n'_{\ell-1}:n'_\ell, \cdot} \in \mathbb{R}^{n_1 \times m}.$$

Thus

$$Z' = \begin{pmatrix} z'_0 \\ z'_1 \\ \vdots \\ z'_{n_1-1} \end{pmatrix} \otimes \mathbf{1}^{(n_{\ell-1})} = (S^{(n_1)} \cdot B_1 \cdot Z_{n_1 n'_{\ell-1}:n'_\ell, \cdot}) \otimes \mathbf{1}^{(n_{\ell-1})} = ((S^{(n_1)} \cdot B_1) \otimes \mathbf{1}^{(n_{\ell-1})}) \cdot Z_{n_1 n'_{\ell-1}:n'_\ell, \cdot},$$

which establishes the correctness of the algorithm.

The space usage of  $\mathcal{A}_\ell$  is comprised of that of  $\mathcal{A}_1$  plus that of  $\mathcal{A}_{\ell-1}$ . (Note that, although many copies of  $\mathcal{A}_{\ell-1}$  are run, they are run sequentially, so the space usage does not add up.) Thus, by induction, the space usage of  $\mathcal{A}_\ell$  is  $\ell$  times that of  $\mathcal{A}_1$ .

The runtime of  $\mathcal{A}_\ell$  is comprised of that of  $\mathcal{A}_1$  plus  $n_1$  times that of  $\mathcal{A}_{\ell-1}$ . By induction, this gives a runtime of  $1 + n_1 + n_1^2 + \dots + n_1^\ell = \frac{n_1^{\ell+1} - 1}{n_1 - 1} = \frac{n'_\ell}{n_1}$  times that of  $\mathcal{A}_1$ , as required.  $\square$

### 6.3 Combining Recursion with BLTs

The algorithm in Section 6.2 requires a base factorization and a corresponding algorithm. We can instantiate this with the RA-BLT rational function approximation algorithm from Sections 3 and 4. This combination yields the following result.

**Proposition 6.5** (Instantiating the recursive construction with the rational function approximation). *Let  $m, \ell \geq 1$ ,  $n_1 \geq 5$ , and  $d \geq 2 + \left(\frac{12+4\log n_1}{\pi}\right)^2$  be integers. Let  $n = n_1^\ell$  and  $n' = \frac{n_1^{\ell+1} - n_1}{n_1 - 1}$ . Then there exist matrices  $B, C^T \in \mathbb{R}^{n \times n'}$  and a streaming algorithm  $\mathcal{A}$  satisfying the following.*

- **Valid Matrix Factorization:**  $BC = A^{(n)}$ .

- **Near-Optimality:**

$$\text{MaxErr}(B, C) \leq \ell \cdot \left( \sqrt{\text{OptLTToe}(n_1)} + 16\sqrt{n_1} \exp\left(-\frac{\pi}{2}\sqrt{d-2}\right) \right)^2,$$

$$\text{where } \text{OptLTToe}(n_1) = 1 + \sum_{k=1}^{n_1-1} \left(2^{-2k} \binom{2k}{k}\right)^2 \leq 1 + \frac{0.57722 + \log(n_1)}{\pi}.$$

- **Valid Algorithm:** The algorithm  $\mathcal{A}$  takes as input a stream  $Z_{0,\cdot}, Z_{1,\cdot}, \dots, Z_{n'-1,\cdot} \in \mathbb{R}^{1 \times m}$  in some arbitrary-but-fixed order (but only reading each  $Z_{i,\cdot}$  once) and outputs a stream  $\tilde{Z}_{0,\cdot}, \tilde{Z}_{1,\cdot}, \dots, \tilde{Z}_{n-1,\cdot} \in \mathbb{R}^{1 \times m}$  such that  $\tilde{Z} = B \cdot Z$ .

- **Efficient Algorithm:** The algorithm  $\mathcal{A}$  uses space  $O(\ell dm)$  and has total runtime  $O(ndm)$ .

*Proof.* Theorem 4.6 provides a base factorization  $B_1, C_1 \in \mathbb{R}^{n_1 \times n_1}$  and a base algorithm  $\mathcal{A}_1$  (Algorithm 1) for computing  $B_1 Z$  in a streaming manner. We are guaranteed that

$$\begin{aligned} \|B_1\|_{2 \rightarrow \infty} &\leq \|B_1^*\|_{2 \rightarrow \infty} + 16\sqrt{n_1} \exp\left(-\frac{\pi}{2}\sqrt{d-2}\right), \\ \|C_1\|_{1 \rightarrow 2} &\leq \|C_1^*\|_{1 \rightarrow 2} + 16\sqrt{n_1} \exp\left(-\frac{\pi}{2}\sqrt{d-2}\right), \end{aligned}$$

where  $B_1^* = C_1^* = M(f, n_1)$  for  $f(x) = 1/\sqrt{1-x}$ . By Proposition 2.2,  $B_1^*, C_1^*$  is the optimal matrix factorization. In particular,

$$\|B_1^*\|_{2 \rightarrow \infty} = \|C_1^*\|_{1 \rightarrow 2} = \sqrt{\text{OptLTToe}(n_1)} = \sqrt{1 + \sum_{k=1}^{n_1-1} \left(2^{-2k} \binom{2k}{k}\right)^2} \leq \sqrt{1 + \frac{\gamma + \log(n_1)}{\pi}},$$

where  $\gamma \leq 0.57722$  is the Euler-Mascheroni constant. The algorithm  $\mathcal{A}_1$  runs in space and time (per iteration)  $O(dm)$ .

Now we apply the recursive algorithm with this base construction. By Proposition 6.3, we obtain a matrix factorization  $B_\ell, C_\ell \in \mathbb{R}^{n \times n'}$  such that  $B_\ell C_\ell = A^{(n)}$ ,

$$\|B_\ell\|_{2 \rightarrow \infty} \leq \sqrt{\ell} \|B_1\|_{2 \rightarrow \infty} \leq \sqrt{\ell} \left( \sqrt{\text{OptLTToe}(n_1)} + 16\sqrt{n_1} \exp\left(-\frac{\pi}{2}\sqrt{d-2}\right) \right),$$

and

$$\|C_\ell\|_{1 \rightarrow 2} = \sqrt{\ell} \|C_1\|_{1 \rightarrow 2} \leq \sqrt{\ell} \left( \sqrt{\text{OptLTToe}(n_1)} + 16\sqrt{n_1} \exp\left(-\frac{\pi}{2}\sqrt{d-2}\right) \right).$$

By Lemma 6.4, Algorithm 2 is a valid streaming algorithm for computing  $\tilde{Z} = B_\ell Z$  and runs in space  $O(\ell dm)$  and total time  $O(ndm)$ , as required.  $\square$

Setting parameters in Proposition 6.5 yields our result.

**Theorem 6.6** (Formal version of Theorem 1.2). *Let  $n, n_1, m \geq 1$  be integers. Then there exist matrices  $B, C^T \in \mathbb{R}^{n \times n'}$  (for some  $n' \leq O(n)$ ) and a streaming algorithm  $\mathcal{A}$  satisfying the following.*

- **Valid Matrix Factorization:**  $BC = A^{(n)}$ .
- **Near-Optimality:**

$$\text{MaxErr}(B, C) \leq \text{Opt}(n) \cdot \left(1 + O\left(\frac{1}{\log(n_1)}\right)\right) + O(\log(n_1)),$$

where  $\text{Opt}(n) = \inf\{\text{MaxErr}(B_*, C_*) : B_* C_* = A^{(n)}\} = \frac{\log n}{\pi} \pm O(1)$  is the optimal value of the matrix factorization objective over all possible factorizations.

- **Valid Algorithm:** The algorithm  $\mathcal{A}$  takes as input a stream  $Z_{0,\cdot}, Z_{1,\cdot}, \dots, Z_{n'-1,\cdot} \in \mathbb{R}^{1 \times m}$  in some arbitrary-but-fixed order (but only reading each  $Z_{i,\cdot}$  once) and outputs a stream  $\tilde{Z}_{0,\cdot}, \tilde{Z}_{1,\cdot}, \dots, \tilde{Z}_{n-1,\cdot} \in \mathbb{R}^{1 \times m}$  such that  $\tilde{Z} = B \cdot Z$ .
- **Efficient Algorithm:** The algorithm  $\mathcal{A}$  uses space  $O(\log(n) \log(n_1)m)$  and has total runtime  $O(n \log(n_1)^2 m)$ .

To obtain Theorem 1.2 from Theorem 6.6 we simply need to set  $n_1$  to a value that is superconstant (so that  $1/O(\log(n_1)) = o(1)$ ) but not too large (so that  $O(\log(n) \log(n_1)m) = \tilde{O}(\log(n)m)$ ). For example, we can set  $n_1 = \Theta(\log(n))$ .

*Proof.* Let  $n_1 \geq 5$ ,  $d = \left\lceil 2 + \left(\frac{12+4\log n_1}{\pi}\right)^2 \right\rceil$ ,  $\ell = \lceil \log(n)/\log(n_1) \rceil$ , and  $n' = \frac{n_1^{\ell+1} - n_1}{n_1 - 1}$  be integers.

Then  $n \leq n_1^\ell < n \cdot n_1$ . We will provide a matrix factorization of size  $n_1^\ell$ , which can be truncated to one of size  $n$ .

Proposition 6.5 provides the matrix factorization and algorithm. It gives the near-optimality guarantee

$$\|B_\ell\|_{2 \rightarrow \infty} \cdot \|C_\ell\|_{1 \rightarrow 2} \leq \ell \cdot \left( \sqrt{\text{OptLTToe}(n_1)} + 16\sqrt{n_1} \exp\left(-\frac{\pi}{2}\sqrt{d-2}\right) \right)^2,$$

where  $\text{OptLTToe}(n_1) = 1 + \sum_{k=1}^{n_1-1} \left(2^{-2k} \binom{2k}{k}\right)^2 \leq 1 + \frac{\gamma + \log(n_1)}{\pi}$ . It only remains for us to simplify this guarantee.

We assume  $d \geq 2 + \left(\frac{2\log(16)+3\log(n_1)}{\pi}\right)^2$  so that  $16\sqrt{n_1} \exp\left(-\frac{\pi}{2}\sqrt{d-2}\right) \leq 1/n_1$ .

Thus

$$\begin{aligned}
\|B_\ell\|_{2 \rightarrow \infty} \|C_\ell\|_{1 \rightarrow 2} &\leq \ell \cdot \left( \sqrt{\text{OptLTToe}(n_1)} + \frac{1}{n_1} \right)^2 \\
&= \ell \cdot \text{OptLTToe}(n_1) + \frac{\ell}{n_1} \cdot \left( 2\sqrt{\text{OptLTToe}(n_1)} + \frac{1}{n_1} \right) \\
&\leq \ell \cdot \left( 1 + \frac{\gamma + \log(n_1)}{\pi} \right) + \frac{\ell}{n_1} \cdot \left( 2\sqrt{n_1} + \frac{1}{n_1} \right) \\
&= \frac{\ell \log(n_1)}{\pi} + \ell \cdot (1 + \gamma/\pi) + \ell \cdot \left( \frac{2}{\sqrt{n_1}} + \frac{1}{n_1^2} \right) \\
&= \frac{\log(n_1^\ell)}{\pi} + \ell \cdot \left( 1 + \frac{\gamma}{\pi} + \frac{2}{\sqrt{n_1}} + \frac{1}{n_1^2} \right) \\
&\leq \frac{\log(n) + \log(n_1)}{\pi} + \ell \cdot \left( 1 + \frac{\gamma}{\pi} + \frac{2}{\sqrt{5}} + \frac{1}{5^2} \right) \\
&\leq \frac{\log(n)}{\pi} + 3\ell + \frac{\log(n_1)}{\pi}.
\end{aligned}$$

Now we invoke the lower bound of Matoušek, Nikolov, and Talwar [MNT20] or Mathias [Mat93] (see Equation 2.9), which tells us

$$\text{Opt}(n) \geq \frac{\log(n)}{\pi} - 1,$$

where  $\text{Opt}(n)$  is the optimal value of the objective function over all factorizations of size  $n$ . Thus

$$\begin{aligned}
\|B_\ell\|_{2 \rightarrow \infty} \|C_\ell\|_{1 \rightarrow 2} &\leq \text{Opt}(n) + 1 + 3\ell + \frac{\log(n_1)}{\pi} \\
&\leq \text{Opt}(n) + 4 + 3 \frac{\log(n)}{\log(n_1)} + \frac{\log(n_1)}{\pi} \\
&\leq \text{Opt}(n) + 4 + 3 \frac{1 + \pi \text{Opt}(n)}{\log(n_1)} + \frac{\log(n_1)}{\pi} \\
&= \text{Opt}(n) \cdot \left( 1 + \frac{3\pi}{\log(n_1)} \right) + 4 + \frac{3}{\log(n_1)} + \frac{\log(n_1)}{\pi}.
\end{aligned}$$

By Proposition 6.5, the space usage of the algorithm is  $O(\ell dm) = O\left(\frac{\log(n)}{\log(n_1)} (\log(n_1))^2 m\right) = O(\log(n) \log(n_1) m)$  and the total runtime is  $O(ndm) = O(n \log(n_1)^2 m)$ .  $\square$

## 7 Numerical Lower Bound on Optimal Performance

In this section, we develop *numerical* lower bounds on the optimal matrix factorizations of the all-ones lower triangular matrix  $A^{(n)}$  for various classes of matrices. More precisely, given the sequence length  $n$ , we write down a semidefinite program that provides a lower bound on the lowest achievable error over a specific class of correlation matrices (for example, Toeplitz matrices or triangular matrices).

In general, we consider mechanisms whose matrices can be written as  $C = \mathbb{J}(c) = \sum_i \mathbb{J}_i c_i$  where  $c \in \mathbb{R}^m$  denotes the parameters of the mechanism (to be optimized) and  $\mathbb{J} : \mathbb{R}^m \mapsto \mathbb{R}^{n \times n}$  is an linear operator and  $\mathbb{J}_i \in \mathbb{R}^{n \times n}$  are “basis” matrices. Any linear subspace of the space of  $n \times n$  matrices (including lower triangular matrices and Toeplitz matrices) can be expressed in this form.

All results in this section derive from the following fundamental result which is a straightforward consequence of weak duality:

**Theorem 7.1.** *Let  $\mathbb{A}$  denote the  $n \times n$  lower triangular matrix with all entries equal to 1. For any  $c \in \mathbb{R}^m$ , define the objective for the correlated mechanism with correlation matrix  $\mathbb{J}(c)$  by*

$$F(c) = \text{MaxErr}((\mathbb{J}(c))^{-1} \mathbb{A}, \mathbb{J}(c)) = \|\mathbb{J}(c)\|_{1 \rightarrow 2}^2 \left\| (\mathbb{J}(c))^{-1} \mathbb{A} \right\|_{2 \rightarrow \infty}^2.$$

Further, for any set  $\mathfrak{C} \subseteq \mathbb{R}^m$ , we have

$$\min_{c \in \mathfrak{C}} F(c) \geq \frac{1}{\min_{\substack{\Gamma_0, \dots, \Gamma_N \in \mathbb{S}_+^n \\ \sum_i \mathbb{A}_i^T \Gamma_i \mathbb{A}_i = 1}} \max_{\|\mathbb{J}(c)\|_{1 \rightarrow 2} = 1} \left\langle \mathbb{J}(c)^\top \mathbb{J}(c), \sum_i \Gamma_i \right\rangle} \quad (7.1)$$

where  $\mathbb{S}_+^n$  denotes the space of symmetric positive semidefinite  $n \times n$  matrices,  $\mathbb{A}_i$  denotes the  $i$ -th column of  $\mathbb{A}$  and  $\langle A, B \rangle = \text{trace}(AB)$  denotes the standard inner product between symmetric matrices  $A, B \in \mathbb{S}^N$ .

Furthermore, if the inner maximization can be relaxed to a concave maximization problem, i.e., there exists a compact convex set  $\mathcal{C} \subset \mathbb{S}^N$  such that

$$\max_{\substack{c \in \mathfrak{C} \\ \|\mathbb{J}(c)\|_{1 \rightarrow 2} = 1}} \left\langle \mathbb{J}(c)^\top \mathbb{J}(c), \Gamma \right\rangle = \max_{S \in \mathcal{C}} \langle S, \Gamma \rangle \quad \forall \Gamma \in \mathbb{S}_+^N$$

then the inequality in Equation 7.1 is an equality.

*Proof.* Since the objective is invariant to scaling  $c$  by any positive constant, i.e.,  $F(\alpha c) = F(c)$  for all  $\alpha \in \mathbb{R}, \alpha > 0$ , we can set  $\alpha = \frac{1}{\|\mathbb{J}c\|_{1 \rightarrow 2}}$  so that the optimization problem becomes

$$\min_{\substack{c \in \mathfrak{C} \\ \|\mathbb{J}(c)\|_{1 \rightarrow 2} = 1}} \left\| (\mathbb{J}(c))^{-1} \mathbb{A} \right\|_{2 \rightarrow \infty}^2 = \min_{\substack{c \in \mathfrak{C} \\ \|\mathbb{J}(c)\|_{1 \rightarrow 2} = 1}} \max_i \left\| \left( \mathbb{J}(c)^\top \right)^{-1} \mathbb{A}_i \right\|^2 \quad (7.2)$$

From Lemma 7.3, we have that

$$\mathbf{e}^\top (M)^{-1} \mathbf{e} = \frac{1}{\min_{\substack{\Gamma \in \mathbb{S}_+^N \\ \mathbf{e}^\top \Gamma \mathbf{e}}} \langle \Gamma, M \rangle}$$

for any  $M \in \mathbb{S}_+^n$ . Applying this to  $M = \mathbb{J}(c)^\top \mathbb{J}(c)$  and  $\mathbf{e} = \mathbb{A}_i$  (for  $i = 0, \dots, n$ ) and plugging this into (7.2), we obtain

$$\max_i \left\| \left( \mathbb{J}(c)^\top \right)^{-1} \mathbb{A}_i \right\|^2 = \max_i \max_{\substack{\Gamma_i \succeq 0 \\ \mathbb{A}_i^T \Gamma_i \mathbb{A}_i = 1}} \frac{1}{\langle \mathbb{J}(c)^\top \mathbb{J}(c), \Gamma_i \rangle} = \frac{1}{\min_i \min_{\substack{\Gamma_i \succeq 0 \\ \mathbb{A}_i^T \Gamma_i \mathbb{A}_i = 1}} \langle \mathbb{J}(c)^\top \mathbb{J}(c), \Gamma_i \rangle}$$

The denominator can be rewritten as

$$\begin{aligned} \min_{\nu \in \Delta^n} \sum_i \nu_i \left( \min_{\substack{\Gamma_i \succeq 0 \\ \mathbb{A}_i^T \Gamma_i \mathbb{A}_i = 1}} \left\langle \mathbb{J}(c)^\top \mathbb{J}(c), \Gamma_i \right\rangle \right) &= \min_{\nu \in \Delta^n} \sum_i \min_{\substack{\Gamma_i \succeq 0 \\ \mathbb{A}_i^T \Gamma_i \mathbb{A}_i = \nu_i}} \left\langle \mathbb{J}(c)^\top \mathbb{J}(c), \Gamma_i \right\rangle \\ &= \min_{\substack{\Gamma_0, \dots, \Gamma_N \in \mathbb{S}_+^n \\ \sum_i \mathbb{A}_i^T \Gamma_i \mathbb{A}_i = 1}} \left\langle \mathbb{J}(c)^\top \mathbb{J}(c), \sum_i \Gamma_i \right\rangle \end{aligned}$$

where  $\Delta^n$  is the simplex in  $n$  dimensions. Thus, the optimization problem (7.2) can be rewritten as

$$\min_{\|\mathbb{J}(c)\|_{1 \rightarrow 2} \leq 1} \frac{1}{\min_{\substack{\Gamma_0, \dots, \Gamma_N \in \mathbb{S}_+^n \\ \sum_i \mathbb{A}_i^T \Gamma_i \mathbb{A}_i = 1}} \left\langle \mathbb{J}(c)^\top \mathbb{J}(c), \sum_i \Gamma_i \right\rangle} = \frac{1}{\max_{\|\mathbb{J}(c)\|_{1 \rightarrow 2} \leq 1} \min_{\substack{\Gamma_0, \dots, \Gamma_N \in \mathbb{S}_+^n \\ \sum_i \mathbb{A}_i^T \Gamma_i \mathbb{A}_i = 1}} \left\langle \mathbb{J}(c)^\top \mathbb{J}(c), \sum_i \Gamma_i \right\rangle}$$

By weak duality, we have that the optimal value is bounded below by

$$\frac{1}{\min_{\substack{\Gamma_0, \dots, \Gamma_N \in \mathbb{S}_+^n \\ \sum_i \mathbb{A}_i^T \Gamma_i \mathbb{A}_i = 1}} \max_{\|\mathbb{J}(c)\|_{1 \rightarrow 2} \leq 1} \left\langle \mathbb{J}(c)^\top \mathbb{J}(c), \sum_i \Gamma_i \right\rangle}$$

which gives Equation 7.1.

Furthermore, if the inner maximization can be rewritten as a concave maximization problem, the overall problem is a convex-concave problem and strong duality holds by the Von-Neumann minimax theorem [Neu28], so that the order of the min and max can be interchanged without changing the optimal value. Hence, the lower bound is tight.  $\square$

**Corollary 7.2.** *Let  $\mathcal{J} = \{\mathbb{J}(c) : c \in \mathfrak{C}\}$ . A tight lower bound in Theorem 7.1 is achieved in the following 3 cases:*

- $\mathcal{J}$  is the set of all lower triangular matrices.
- $\mathcal{J}$  is the set of all lower triangular Toeplitz matrices.
- $\mathcal{J}$  is the set of all lower triangular Toeplitz matrices formed from a degree 1 constant recurrent sequence.

*Proof.* 1. If  $\mathcal{J}$  is the set of all lower triangular matrices, for any  $\mathbb{J} \in \mathcal{J}$ , letting  $\Delta = \mathbb{J}^\top \mathbb{J}$ , we have that

$$\max_{\|\mathbb{J}\|_{1 \rightarrow 2} \leq 1, \mathbb{J} \in \mathcal{J}} \left\langle \mathbb{J}^\top \mathbb{J}, \Gamma \right\rangle \leq \max_{\substack{\Delta \succeq 0 \\ \text{diag}(\Delta) \leq 1}} \langle \Delta, \Gamma \rangle$$

since for any feasible  $\mathbb{J}$ ,  $\Delta = \mathbb{J}^\top \mathbb{J}$  satisfies the constraints listed in the optimization problem on the right hand side (note that  $\text{diag}(\Delta)$  refers to the vector of diagonal elements of the matrix  $\Delta$ ).

Conversely, given any feasible solution  $\Delta$  satisfying the constraints, one can compute its upper triangular Cholesky factorization (the regular Cholesky factorization multiplied by the matrix with anti-diagonals equal to 1), to obtain  $\mathbb{J}$  that satisfies the constraint  $\|\mathbb{J}\|_{1 \rightarrow 2} \leq 1$  and  $\Delta = \mathbb{J}^\top \mathbb{J}$ . Hence the two optimization problems have equal optimal values, and hence the lower bound in theorem 7.1 is tight.

2. If  $\mathcal{J}$  is the set of all  $n \times n$  lower triangular Toeplitz matrices, let  $\mathbb{J}(c)$  denote the lower triangular Toeplitz matrix whose first column is  $c$ . Then, we have

$$\|\mathbb{J}(c)\|_{1 \rightarrow 2} = \|c\|, \mathbb{J}(c) = \sum_i c_i H_i$$

where  $H_i$  is the matrix with 0s everywhere except the  $i$ -th subdiagonal equal to 1. Thus

$$\max_{\|\mathbb{J}(c)\|_{1 \rightarrow 2} \leq 1, \mathbb{J}(c) \in \mathcal{J}} \langle \mathbb{J}^\top \mathbb{J}, \Gamma \rangle = \max_{\|c\|=1} \langle \mathbb{J}(c)^\top \mathbb{J}(c), \Gamma \rangle$$

and the right hand side is a standard trust region subproblem with a quadratic objective and a single quadratic constraint, that can be solved exactly via an SDP relaxation by the S-lemma [PT07], i.e., there exists an affine map  $Q : \mathbb{S}^n \rightarrow \mathbb{S}^n$  such that the right hand side equals  $\max_{C \succeq 0, \text{trace}(C)=1} \langle Q(\Gamma), C \rangle$ . Hence, the lower bound in Theorem 7.1 is tight.

3. If  $\mathcal{J}$  is the set of all  $n \times n$  lower triangular Toeplitz matrices formed from a degree 1 constant recurrent sequence, we have that  $\mathbb{J} = \mathbb{J}(c)$  where  $c_i = \alpha \theta^i$ . where  $\alpha, \theta \in \mathbb{R}$  are arbitrary real numbers. Then, the problem

$$\max_{\|c\|=1} \langle \mathbb{J}(c)^\top \mathbb{J}(c), \Gamma \rangle = \max_c \frac{\langle \mathbb{J}(c)^\top \mathbb{J}(c), \Gamma \rangle}{c^\top c}$$

can be written as

$$\max_{\theta, \alpha} \frac{\alpha^2 p(\theta; \Gamma)}{\alpha^2 q(\theta)} = \max_{\theta} \frac{p(\theta; \Gamma)}{q(\theta)}$$

where  $p, q$  are polynomials in  $\theta$  and the coefficients of  $q$  are independent of  $\Gamma$  while the coefficients of  $p$  are affine functions of  $\Gamma$ . Using the fact that

$$p(\theta) - \beta q(\theta) \leq 0 \iff p(\theta) - \beta q(\theta) \text{ is a sum of squares polynomial}$$

we can write the maximization problem as being equal to

$$\min_{p(\theta; \Gamma) - \beta q(\theta) \text{ is a sum of squares polynomial}} \beta$$

Thus, the inner maximization can be written as a concave maximization problem of the form required for lower bound tightness in Theorem 7.1 [Par12]. □

**Lemma 7.3.** [Lower bound on matrix fractional function] For any  $\mathbf{e} \in \mathbb{R}^n$  and  $M \in \mathbb{S}_{++}^n$ , we have

$$\mathbf{e}^\top (M)^{-1} \mathbf{e} = \frac{1}{\min_{\substack{\Gamma \in \mathbb{S}_+^n \\ \mathbf{e}^\top \Gamma \mathbf{e} = 1}} \langle \Gamma, M \rangle}$$



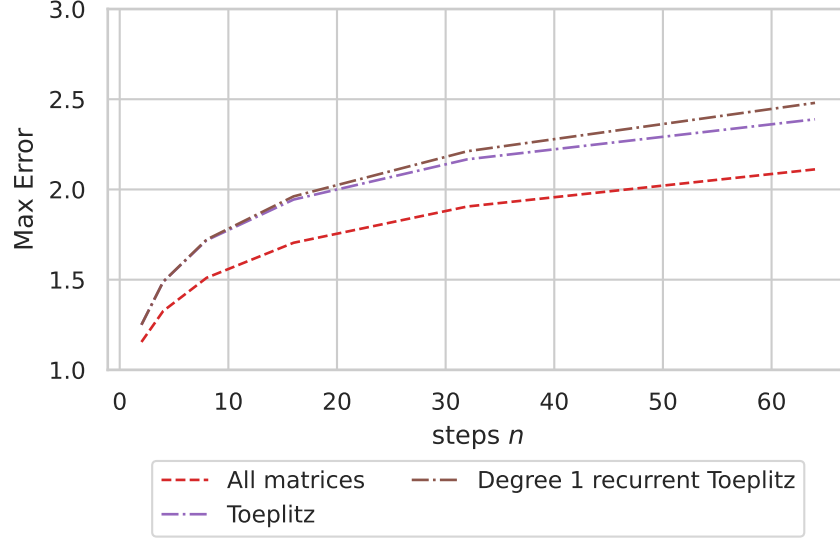


Figure 4: Semidefinite programming based lower bounds on optimal performance for various classes of matrices based on the results from theorem 7.1

*Proof.* Let  $\mathbf{e} \in \mathbb{R}^n$  be an arbitrary vector. Now, consider any positive definite matrix  $M \in \mathbb{S}_+^n$  and eigendecomposition  $M = \sum_i \lambda_i u_i u_i^\top$  for some orthogonal basis  $\{u_i\}$ . Further, expressed in this basis, suppose that  $\mathbf{e} = \sum_i \sigma_i u_i$ .

Then, we have

$$\mathbf{e}^\top M^{-1} \mathbf{e} = \sum_i \frac{1}{\lambda_i} (u_i^\top \mathbf{1})^2 = \sum_i \frac{\sigma_i^2}{\lambda_i}$$

Let  $\langle M, \Gamma \rangle$  denote the inner product in  $\mathbb{S}^n$ , i.e.,  $\langle M, \Gamma \rangle = \text{trace}(M\Gamma)$ . Further, for any matrix  $\Gamma \in \mathbb{S}_+^n$ , we have

$$\begin{aligned} (\mathbf{e}^\top M^{-1} \mathbf{e}) \langle M, \Gamma \rangle &= \left( \sum_i \frac{\sigma_i^2}{\lambda_i} \right) \left( \sum_i \lambda_i u_i^\top \Gamma u_i \right) = \left( \sum_i \frac{\sigma_i^2}{\lambda_i} \right) \left( \sum_i \lambda_i \|\Gamma^{1/2} u_i\|^2 \right) \\ &\geq \left( \sum_i \sigma_i \|\Gamma^{1/2} u_i\| \right)^2 \geq \left\| \sum_i \sigma_i \Gamma^{1/2} u_i \right\|^2 = \mathbf{e}^\top \Gamma \mathbf{e} \end{aligned}$$

where the first inequality is an application of the Cauchy-Schwartz inequality and the second inequality is an application of the triangle inequality.

Thus, for any  $M, \Gamma \in \mathbb{S}_+^n$ , we have

$$\mathbf{e}^\top M^{-1} \mathbf{e} \geq \frac{\mathbf{e}^\top \Gamma \mathbf{e}}{\langle M, \Gamma \rangle}$$

Further, choosing  $\Gamma = (M)^{-1} \mathbf{e} \mathbf{e}^\top (M)^{-1}$ , the inequality is replaced with equality. Hence, we have

$$\mathbf{e}^\top M^{-1} \mathbf{e} = \max_{\Gamma \succeq 0} \frac{\mathbf{e}^\top \Gamma \mathbf{e}}{\langle M, \Gamma \rangle} = \max_{\substack{\Gamma \succeq 0 \\ \mathbf{e}^\top \Gamma \mathbf{e} = 1}} \frac{1}{\langle M, \Gamma \rangle}$$

□

## 8 BLT with $d = 1$ Buffers

In this section, we show that it is possible to achieve  $\text{MaxErr}(B, C) = O(n^{1/6})$  error for a degree  $d = 1$  constant recurrent sequence, without appealing to the sophisticated machinery of rational function approximation that build the crux of the paper. Unfortunately, this approach does not naturally extend to degrees  $d > 1$ . In comparison, adding independent noise (that is when  $d = 0$ ) results in an  $\ell_\infty$  error of  $O(n)$ .

We consider the factorization  $A = BC$ , where  $B = AC^{-1}$  and  $C$  and  $C^{-1}$  are parameterized as follows. Let  $a, \lambda \in [-1, 1]$  with  $|\lambda - a^2| \leq 1$ . Define

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ a^2 & 1 & 0 & 0 & \cdots & 0 \\ a^2\lambda & a^2 & 1 & 0 & \cdots & 0 \\ a^2\lambda^2 & a^2\lambda & a^2 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix}, \quad C^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ -a^2 & 1 & 0 & 0 & \cdots & 0 \\ -a^2(\lambda - a^2) & -a^2 & 1 & 0 & \cdots & 0 \\ -a^2(\lambda - a^2)^2 & -a^2(\lambda - a^2) & -a^2 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \end{pmatrix}. \quad (8.1)$$

That is,  $C$  corresponds to the generating function  $c(x) = 1 + \frac{a^2x}{1-\lambda x}$ , which clearly has degree 1. It can be checked that the expression for  $C^{-1}$  is correct (via Lemma 3.1) as  $1/c(x) = 1 - \frac{a^2x}{1-(\lambda-a^2)x}$ .

**Theorem 8.1.** *Setting  $\lambda = 1 - \frac{1}{n^{2/3}}$ , and  $a^2 = \frac{1}{n^{1/3}} (1 - \frac{1}{n^{1/3}})$  in Equation 8.1 gives*

$$\|C\|_{1 \rightarrow 2}^2 \leq O(1), \quad (8.2)$$

$$\|B\|_{2 \rightarrow \infty} = \|AC^{-1}\|_{2 \rightarrow \infty} \leq O(n^{1/6}) \quad (8.3)$$

*Proof.* The maximum squared column norm of  $C$  is

$$\|C\|_{1 \rightarrow 2}^2 = 1 + a^4 (1 + \lambda^2 + \lambda^4 + \cdots \lambda^{2(n-2)}) \leq 1 + \frac{a^4}{1 - \lambda^2}. \quad (8.4)$$

Since  $1 - \lambda^2 = 2n^{-2/3} - n^{-4/3} = \Theta(n^{-2/3})$  and  $a^4 = n^{-2/3}(1 - n^{-1/3})^2 = \Theta(n^{-2/3})$ , we have  $\|C\|_{1 \rightarrow 2} = \Theta(1)$ .

Let  $b = [b_{n-1}, \dots, b_0]$  be the last row of  $B = AC^{-1}$ . This is the longest row, so  $\|B\|_{2 \rightarrow \infty}^2 = \|b\|_2^2$ . In the following, we bound each element of the vector  $b$ . We have  $b_0 = 1$  and, for any  $i \in \{1, \dots, n-1\}$ ,

$$b_i = 1 - a^2 (1 + (\lambda - a^2) + \cdots + (\lambda - a^2)^{i-2}) = 1 - \frac{a^2(1 - (\lambda - a^2)^{i-1})}{1 + a^2 - \lambda}. \quad (8.5)$$

Thus

$$\begin{aligned}
\|B\|_{2 \rightarrow \infty}^2 &= 1 + \sum_{i=1}^{n-1} \left( 1 - \frac{a^2(1 - (\lambda - a^2)^{i-1})}{1 + a^2 - \lambda} \right)^2 \\
&= 1 + \sum_{i=1}^{n-1} \left( \frac{(1 - \lambda) + a^2(\lambda - a^2)^{i-1}}{1 + a^2 - \lambda} \right)^2 \\
&= 1 + \frac{1}{(1 + a^2 - \lambda)^2} \sum_{i=1}^{n-1} (1 - \lambda)^2 + 2(1 - \lambda)a^2(\lambda - a^2)^{i-1} + a^4(\lambda - a^2)^{2(i-1)} \\
&= 1 + \frac{(n-1)(1 - \lambda)^2 + 2(1 - \lambda)a^2 \frac{1 - (\lambda - a^2)^{n-1}}{1 + a^2 - \lambda} + a^4 \frac{1 - (\lambda - a^2)^{2(n-1)}}{1 - (\lambda - a^2)^2}}{(1 + a^2 - \lambda)^2} \tag{8.6}
\end{aligned}$$

$$\leq 1 + \frac{(n-1)(1 - \lambda)^2 + 2(1 - \lambda)a^2 \frac{1}{1 + a^2 - \lambda} + a^4 \frac{1}{1 - (\lambda - a^2)^2}}{(1 + a^2 - \lambda)^2}. \tag{8.7}$$

Now  $a^2 = n^{-1/3} - n^{-2/3}$ ,  $1 - \lambda = -n^{-2/3}$ ,  $1 - \lambda + a^2 = n^{-1/3}$  and  $1 - (\lambda - a^2)^2 = 2n^{-1/3} - n^{-2/3} = \Theta(n^{-1/3})$ . Substituting these into Equation 8.7 gives

$$\begin{aligned}
\|B\|_{2 \rightarrow \infty}^2 &\leq 1 + \frac{(n-1)n^{-4/3} - 2n^{-2/3}(n^{-1/3} - n^{-2/3})/n^{-1/3} + (n^{-1/3} - n^{-2/3})^2/(2n^{-1/3} - n^{-2/3})}{n^{-2/3}} \\
&= 1 + \frac{n-1}{n^{2/3}} - 2 + \frac{2}{n^{1/3}} + \frac{n^{-2/3} - 2n^{-1} + n^{-4/3}}{2n^{-1} - n^{-4/3}} \\
&= 1 + \frac{n-1}{n^{2/3}} - 2 + \frac{2}{n^{1/3}} + \frac{n^{1/3} - 2 + n^{-1/3}}{2 - n^{-1/3}} \\
&= \Theta(n^{1/3})
\end{aligned}$$

as required. □

## 9 Conjecture

We give the following conjecture which improves the results in Section 4. If true, this would imply that BLTs can work with  $d = O(\log n)$  space complexity.

**Conjecture 9.1.** *For all  $n \in \mathbb{N}$  there exists a rational function  $r : \mathbb{C} \rightarrow \mathbb{C}$  of degree  $d = O(\log n)$  such that the following hold.*

$$\begin{aligned}
\frac{1}{2\pi} \int_{-\pi}^{\pi} \left| \frac{1}{r(x)} - \frac{1}{\sqrt{1-x}} \right|^2 d\theta &\leq \frac{1}{n} \quad \text{where } x = \exp(\sqrt{-1}\theta - 1/n). \\
\frac{1}{2\pi} \int_{-\pi}^{\pi} \left| \frac{r(x)}{1-x} - \frac{1}{\sqrt{1-x}} \right|^2 d\theta &\leq \frac{1}{n} \quad \text{where } x = \exp(\sqrt{-1}\theta - 1/n).
\end{aligned}$$

Compared to what we prove in Sections 4.1 and 4.2, this conjecture is quantitatively stronger in that  $d = O(\log n)$  instead of  $d = O(\log^2 n)$  but qualitatively weaker in that (i) we prove an approximation guarantee for  $|x| \leq \exp(-1/n)$  rather than  $|x| \leq 1$  and, (ii) rather than a uniform bound on  $|r(x) - \sqrt{1-x}|$ , we bound the integrals directly.

## Acknowledgements

We thank Andreas Terzis, Shuang Song, and Arun Ganesh for comments on the draft. We thank Aleksandar Nikolov and Jalaj Upadhyay for helpful discussions about prior work. We thank Mark Bun for making us aware of the results of Newman [New64].

## References

- [ACGMMTZ16] M. Abadi, A. Chu, I. J. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. “Deep Learning with Differential Privacy”. In: *Proc. of the 2016 ACM SIGSAC Conf. on Computer and Communications Security (CCS’16)*. 2016, pp. 308–318 (cit. on p. 24).
- [AFKT23] H. Asi, V. Feldman, T. Koren, and K. Talwar. “Near-optimal algorithms for private online optimization in the realizable regime”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 1107–1120 (cit. on p. 24).
- [ALMM19] N. Alon, R. Livni, M. Malliaris, and S. Moran. “Private PAC learning implies finite Littlestone dimension”. In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 2019, pp. 852–860 (cit. on pp. 3, 26).
- [AP24] J. D. Andersson and R. Pagh. “A smooth binary mechanism for efficient private continual observation”. In: *Advances in Neural Information Processing Systems* 36 (2024) (cit. on pp. 3, 16).
- [AS17] N. Agarwal and K. Singh. “The price of differential privacy for online learning”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 32–40 (cit. on p. 24).
- [BDRS18] M. Bun, C. Dwork, G. N. Rothblum, and T. Steinke. “Composable and versatile privacy via truncated cdp”. In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*. 2018, pp. 74–86 (cit. on pp. 3, 26).
- [BNS13] A. Beimel, K. Nissim, and U. Stemmer. “Private learning and sanitization: Pure vs. approximate differential privacy”. In: *International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer. 2013, pp. 363–378. URL: <https://arxiv.org/abs/1407.2674> (cit. on pp. 3, 26).
- [BNSV15] M. Bun, K. Nissim, U. Stemmer, and S. Vadhan. “Differentially private release and learning of threshold functions”. In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. IEEE. 2015, pp. 634–649 (cit. on pp. 3, 26).

- [Bra+18] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: <http://github.com/google/jax> (cit. on p. 53).
- [BS16] M. Bun and T. Steinke. “Concentrated differential privacy: Simplifications, extensions, and lower bounds”. In: *Theory of Cryptography Conference*. Springer. 2016, pp. 635–658 (cit. on p. 25).
- [BST14] R. Bassily, A. Smith, and A. Thakurta. “Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds”. In: *Proc. of the 2014 IEEE 55th Annual Symp. on Foundations of Computer Science (FOCS)*. 2014, pp. 464–473 (cit. on p. 24).
- [CCDPGST24] C. A. Choquette-Choo, K. Dvijotham, K. Pillutla, A. Ganesh, T. Steinke, and A. Thakurta. “Correlated Noise Provably Beats Independent Noise for Differentially Private Learning”. In: *ICLR*. 2024 (cit. on p. 24).
- [CCGMMRGTX24] C. A. Choquette-Choo, A. Ganesh, R. McKenna, H. B. McMahan, J. Rush, A. Guha Thakurta, and Z. Xu. “(Amplified) Banded Matrix Factorization: A unified approach to private training”. In: *Advances in Neural Information Processing Systems* 36 (2024) (cit. on pp. 3, 24).
- [CCMRT22] C. A. Choquette-Choo, H. B. McMahan, K. Rush, and A. Thakurta. *Multi-Epoch Matrix Factorization Mechanisms for Private Machine Learning*. 2022. URL: <https://arxiv.org/abs/2211.06530> (cit. on pp. 3, 24).
- [CKS20] C. L. Canonne, G. Kamath, and T. Steinke. “The discrete gaussian for differential privacy”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 15676–15688. URL: <https://arxiv.org/abs/2004.00010> (cit. on p. 5).
- [CLNSS23] E. Cohen, X. Lyu, J. Nelson, T. Sarlós, and U. Stemmer. “Optimal differentially private learning of thresholds and quasi-concave optimization”. In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*. 2023, pp. 472–482. URL: <https://arxiv.org/abs/2211.06387> (cit. on pp. 3, 26).
- [CSS11] T.-H. H. Chan, E. Shi, and D. Song. “Private and Continual Release of Statistics”. In: *ACM Trans. on Information Systems Security* 14.3 (Nov. 2011), 26:1–26:24 (cit. on pp. 3, 11, 16, 26, 56).
- [DMNS06] C. Dwork, F. McSherry, K. Nissim, and A. Smith. “Calibrating Noise to Sensitivity in Private Data Analysis”. In: *Proc. of the Third Conf. on Theory of Cryptography (TCC)*. 2006, pp. 265–284. URL: [http://dx.doi.org/10.1007/11681878\\_14](http://dx.doi.org/10.1007/11681878_14) (cit. on p. 3).
- [DMRST22] S. Denisov, B. McMahan, K. Rush, A. Smith, and A. G. Thakurta. *Improved Differential Privacy for SGD via Optimal Private Linear Operators on Adaptive Streams*. 2022. URL: <https://arxiv.org/abs/2202.08312> (cit. on pp. 3, 15, 24).

- [DNPR10] C. Dwork, M. Naor, T. Pitassi, and G. N. Rothblum. “Differential Privacy Under Continual Observation”. In: *Proc. of the Forty-Second ACM Symp. on Theory of Computing (STOC’10)*. 2010, pp. 715–724 (cit. on pp. 3, 11, 16, 26, 56).
- [DNRR15] C. Dwork, M. Naor, O. Reingold, and G. N. Rothblum. “Pure differential privacy for rectangle queries via private partitions”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2015, pp. 735–751. URL: <https://www.iacr.org/archive/asiacrypt2015/94520352/94520352.pdf> (cit. on p. 25).
- [DR14] C. Dwork and A. Roth. “The algorithmic foundations of differential privacy”. In: *Foundations and Trends in Theoretical Computer Science* 9.3–4 (2014), pp. 211–407 (cit. on p. 3).
- [elk] N. D. E. ([https://mathoverflow.net/users/14830/noam-d elkies](https://mathoverflow.net/users/14830/noam-d-elkies)). *Upper limit on the central binomial coefficient*. MathOverflow. URL: <https://mathoverflow.net/q/133752> (version: 2013-06-14). eprint: <https://mathoverflow.net/q/133752>. URL: <https://mathoverflow.net/q/133752> (cit. on pp. 16, 19).
- [FHU22] H. Fichtenberger, M. Henzinger, and J. Upadhyay. “Constant matters: Fine-grained Complexity of Differentially Private Continual Observation”. In: *arXiv preprint arXiv:2202.11205* (2022). URL: <https://arxiv.org/abs/2202.11205> (cit. on pp. 3, 6–12, 16, 17, 19, 27–29).
- [GJK21] J. Gillenwater, M. Joseph, and A. Kulesza. “Differentially private quantiles”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 3713–3722. URL: <https://arxiv.org/abs/2102.08244> (cit. on pp. 3, 26).
- [GT19] A. Gopal and L. Trefethen. “Representation of conformal maps by rational functions.” In: *Numer. Math.* 142 (2019), pp. 359–382. URL: <https://doi.org/10.1007/s00211-019-01023-z> (cit. on pp. 9, 40).
- [Hon15] J. Honaker. “Efficient use of differentially private binary trees”. In: *Theory and Practice of Differential Privacy (TPDP 2015), London, UK 2* (2015), pp. 26–27 (cit. on p. 16).
- [HUU23] M. Henzinger, J. Upadhyay, and S. Upadhyay. “Almost tight error bounds on differentially private continual counting”. In: *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. 2023, pp. 5003–5039. URL: <https://arxiv.org/abs/2211.05006> (cit. on pp. 3, 25).
- [HUU24] M. Henzinger, J. Upadhyay, and S. Upadhyay. “A unifying framework for differentially private sums under continual observation”. In: *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. 2024, pp. 995–1018 (cit. on p. 3).

- [KLMNS20] H. Kaplan, K. Ligett, Y. Mansour, M. Naor, and U. Stemmer. “Privately learning thresholds: Closing the exponential gap”. In: *Conference on Learning Theory*. PMLR. 2020, pp. 2263–2285 (cit. on pp. 3, 26).
- [KLS21] P. Kairouz, Z. Liu, and T. Steinke. “The distributed discrete gaussian mechanism for federated learning with secure aggregation”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 5201–5212. URL: <https://arxiv.org/abs/2102.06387> (cit. on p. 5).
- [KMSTTX21] P. Kairouz, B. McMahan, S. Song, O. Thakkar, A. Thakurta, and Z. Xu. “Practical and private (deep) learning without sampling or shuffling”. In: *ICML*. 2021 (cit. on pp. 3, 24).
- [KP11] M. Kauers and P. Paule. *The Concrete Tetrahedron: Symbolic Sums, Recurrence Equations, Generating Functions, Asymptotic Estimates*. 1st. Springer Publishing Company, Incorporated, 2011. ISBN: 3709104440 (cit. on pp. 9, 29, 30, 32).
- [LMHMR15] C. Li, G. Miklau, M. Hay, A. McGregor, and V. Rastogi. “The matrix mechanism: optimizing linear counting queries under differential privacy”. In: *The VLDB journal* 24 (2015), pp. 757–781 (cit. on p. 26).
- [LSŠ08] T. Lee, A. Shraibman, and R. Špalek. “A direct product theorem for discrepancy”. In: *2008 23rd Annual IEEE Conference on Computational Complexity*. IEEE. 2008, pp. 71–80. URL: <https://web.archive.org/web/20230109145533/https://www2.mta.ac.il/~adish/Pubs/Papers/DPTDiscrepancy.pdf> (cit. on p. 17).
- [Mat93] R. Mathias. “The Hadamard operator norm of a circulant and applications”. In: *SIAM journal on matrix analysis and applications* 14.4 (1993), pp. 1152–1167 (cit. on pp. 3, 4, 17, 26, 65).
- [Mir12] I. Mironov. “On significance of the least significant bits for differential privacy”. In: *Proceedings of the 2012 ACM conference on Computer and communications security*. 2012, pp. 650–661 (cit. on p. 5).
- [MMHM21] R. McKenna, G. Miklau, M. Hay, and A. Machanavajjhala. “Hdmm: Optimizing error of high-dimensional statistical queries under differential privacy”. In: *arXiv preprint arXiv:2106.12118* (2021) (cit. on p. 26).
- [MNT20] J. Matoušek, A. Nikolov, and K. Talwar. “Factorization norms and hereditary discrepancy”. In: *International Mathematics Research Notices* 2020.3 (2020), pp. 751–780. URL: <https://arxiv.org/abs/1408.1376> (cit. on pp. 3, 4, 17, 23, 65).
- [Neu28] J v. Neumann. “Zur theorie der gesellschaftsspiele”. In: *Mathematische annalen* 100.1 (1928), pp. 295–320 (cit. on p. 67).
- [New64] D. J. Newman. “Rational approximation to  $|x|$ .” In: *Michigan Mathematical Journal* 11.1 (1964), pp. 11–14. URL: <https://doi.org/10.1307/mmj/1028999029> (cit. on pp. 9, 10, 40, 72).



- [OWN97] A. V. Oppenheim, A. S. Willsky, and Nawab. *Signals and Systems*. Vol. 2. 1997 (cit. on p. 29).
- [Par12] P. A. Parrilo. “Chapter 3: Polynomial optimization, sums of squares, and applications”. In: *Semidefinite optimization and convex algebraic geometry*. SIAM, 2012, pp. 47–157 (cit. on p. 68).
- [PT07] I. Pólik and T. Terlaky. “A survey of the S-lemma”. In: *SIAM review* 49.3 (2007), pp. 371–418 (cit. on p. 68).
- [QYL13] W. Qardaji, W. Yang, and N. Li. “Understanding hierarchical methods for differentially private histograms”. In: *Proceedings of the VLDB Endowment* 6.14 (2013), pp. 1954–1965 (cit. on p. 16).
- [SCS13] S. Song, K. Chaudhuri, and A. D. Sarwate. “Stochastic gradient descent with differentially private updates”. In: *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 2013, pp. 245–248 (cit. on p. 24).
- [spe] D. E. S. ([https://mathoverflow.net/users/297/david-e speyer](https://mathoverflow.net/users/297/david-e-speyer)). *Upper limit on the central binomial coefficient*. MathOverflow. URL:<https://mathoverflow.net/q/246875> (version: 2023-11-08). eprint: <https://mathoverflow.net/q/246875>. URL: <https://mathoverflow.net/q/246875> (cit. on p. 19).
- [ST13] A. Smith and A. Thakurta. “(Nearly) optimal algorithms for private online learning in full-information and bandit settings”. In: *Advances in Neural Information Processing Systems*. 2013, pp. 2733–2741 (cit. on p. 24).
- [Sta93] H. Stahl. “Best uniform rational approximation of  $x^\alpha$  on  $[0, 1]$ ”. In: *Bulletin of the American Mathematical Society* 28.1 (1993), pp. 116–122. URL: <https://www.ams.org/journals/bull/1993-28-01/S0273-0979-1993-00351-3/> (cit. on p. 40).
- [Ste22] T. Steinke. “Composition of differential privacy & privacy amplification by subsampling”. In: *arXiv preprint arXiv:2210.00597* (2022). URL: <https://arxiv.org/abs/2210.00597> (cit. on p. 3).
- [SW] J. Sondow and E. W. Weisstein. *Harmonic Number*. MathWorld—A Wolfram Web Resource. URL: <https://mathworld.wolfram.com/HarmonicNumber.html> (cit. on pp. 16, 19).
- [TJ89] N. Tomczak-Jaegermann. “Banach-Mazur distances and finite-dimensional operator ideals”. In: *(No Title)* (1989) (cit. on p. 16).
- [use14] user148848. *Fourier transform of  $1/\cosh$* . Mathematics Stack Exchange. URL:<https://math.stackexchange.com/q/799616> (version: 2019-04-23). 2014. eprint: <https://math.stackexchange.com/q/799616>. URL: <https://math.stackexchange.com/q/799616> (cit. on p. 43).
- [Wik24] Wikipedia contributors. *Companion matrix* — *Wikipedia, The Free Encyclopedia*. [https://en.wikipedia.org/w/index.php?title=Companion\\_matrix&oldid=1201009369](https://en.wikipedia.org/w/index.php?title=Companion_matrix&oldid=1201009369). [Online; accessed 18-March-2024]. 2024 (cit. on p. 33).



- [XWG10] X. Xiao, G. Wang, and J. Gehrke. “Differential privacy via wavelet transforms”. In: *IEEE Transactions on knowledge and data engineering* 23.8 (2010), pp. 1200–1214 (cit. on p. 26).
- [XZACCKMRZ23] Z. Xu, Y. Zhang, G. Andrew, C. A. Choquette-Choo, P. Kairouz, H. B. McMahan, J. Rosenstock, and Y. Zhang. “Federated learning of gboard language models with differential privacy”. In: *arXiv preprint arXiv:2305.18465* (2023) (cit. on pp. 3, 24).
- [ZTC22] Q. Zhang, H. Tran, and A. Cutkosky. “Differentially Private Online-to-batch for Smooth Losses.” In: *NeurIPS*. 2022 (cit. on pp. 3, 24).