The Bit Complexity of Dynamic Algebraic Formulas and Their Determinants

Emile Anand
Caltech, Pasadena, CA, USA

Jan van den Brand
Georgia Tech, Atlanta, GA, USA

Mehrdad Ghadiri
MIT, Cambridge, MA, USA

Daniel J. Zhang
Georgia Tech, Atlanta, GA, USA

Abstract

Many iterative algorithms in computer science require repeated computation of some algebraic expression whose input varies slightly from one iteration to the next. Although efficient data structures have been proposed for maintaining the solution of such algebraic expressions under low-rank updates, most of these results are only analyzed under exact arithmetic (real-RAM model and finite fields) which may not accurately reflect the more limited complexity guarantees of real computers. In this paper, we analyze the stability and bit complexity of such data structures for expressions that involve the inversion, multiplication, addition, and subtraction of matrices under the word-RAM model. We show that the bit complexity only increases linearly in the number of matrix operations in the expression. In addition, we consider the bit complexity of maintaining the determinant of a matrix expression. We show that the required bit complexity depends on the logarithm of the condition number of matrices instead of the logarithm of their determinant. Finally, we discuss rank maintenance and its connections to determinant maintenance. Our results have wide applications ranging from computational geometry (e.g., computing the volume of a polytope) to optimization (e.g., solving linear programs using the simplex algorithm).

2012 ACM Subject Classification Theory of computation → Data structures design and analysis

Keywords and phrases Data Structures, Online Algorithms, Bit Complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.10

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2401.11127`

Funding *Jan van den Brand*: Supported by NSF award CCF-2338816.
Daniel J. Zhang: Supported by NSF award CCF-2338816.

Acknowledgements The authors would like to thank Richard Peng for advice and comments.

1 Introduction

Computing algebraic expressions is a workhorse of many iterative algorithms in modern optimization, computational geometry, and dynamic algorithms. Examples include but are not limited to interior point methods for solving linear programs [30, 15, 29, 8, 10], iterative refinement for solving p-norm regression problems [13, 1, 2, 3, 28], semi-definite programming [26, 27], and many algorithmic graph theory problems [33, 11, 6, 14].

Such algebraic expressions are usually represented as matrix formulas involving matrices and operations such as inversion, multiplication, and addition/subtraction. In many iterative algorithms, the algebraic expression does not change over the course of the algorithm, and

only low-rank updates occur to the corresponding matrices from one iteration to the next. For example, for $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$, if we have $\mathbf{AB}$ from a previous iteration and one column of $\mathbf{B}$ changes in the next iteration, we can update $\mathbf{AB}$ in $O(n^2)$ time which is much faster than computing $\mathbf{AB}$ from scratch again. This has been exploited in many iterative algorithms to reduce the amortized cost of iteration and, therefore, the total running time of the algorithms.

A main component of this approach is the Sherman-Morrison-Woodbury (SMW) identity (see (1)), which informally states that the inverse of a rank-k perturbation of a matrix $\mathbf{A}$ can be obtained by a rank-k perturbation of $\mathbf{A}^{-1}$. Although this identity (also called *inverse maintenance*) has been used from the early days of optimization and control theory [31, 32], it was recently shown that *any matrix formula* involving only inversion, multiplication, and addition/subtraction operations can be maintained under low-rank updates with the Sherman-Morrison-Woodbury identity [9]. The main idea is to inductively construct a large matrix whose inverse contains a block that is precisely the output of the formula.

The result of [9] is under the *real-RAM model*, which assumes each arithmetic operation is carried over to infinite precision in constant time. Although this is a valid assumption for finite fields, it does not hold over real numbers. For example, in modern computers, floating-points are the number system of choice that only has a finite precision. Then, it is unclear whether such inverse maintenance techniques are sufficiently stable so that the downstream iterative algorithm outputs the correct solution (e.g., whether the iterative optimization algorithm converges).

Very recently, [24] analyzed the SMW identity over fixed-point arithmetic and showed that a bit complexity proportional to the logarithm of the condition number (ratio of the largest singular value to the smallest singular value) of the corresponding matrix is sufficient to guarantee the stability of the inverse maintenance over the *word-RAM model* in which the running time of arithmetic operations is proportional to the number of bits of corresponding numbers and only finite precision is guaranteed (and the precision itself depends on the number of utilized bits).

This implies that in order to show that the techniques of [9] also hold over the word-RAM model, we need to bound the condition number of the inductively constructed matrix for arbitrary matrix formulas. Indeed, we affirmatively show that the condition number of the constructed matrix is $\kappa^{O(s)}$, where κ is an upper bound for the condition numbers of the input matrices and s is the number of input matrices. This implies that a bit complexity of $O(s \log \kappa)$ is sufficient to guarantee the stability of dynamically updating the matrix formulas. We point out that a naive analysis would give a bound of $O(2^s \log \kappa)$ for the bit complexity, and our bound on the condition number is asymptotically tight since one can easily see that the product of s matrices each with condition number κ results in a matrix with condition number κ^s, e.g., consider $\mathbf{A}^s$.

In addition, we consider the stability and bit complexity of maintaining the determinant and rank of matrix formulas with inversion, multiplication, and addition/subtraction. An application of maintaining the determinant is in the faster computation of the volume of a polytope [22], and an application of the rank maintenance is in dynamic maximum matching [34].

To maintain the determinant of a matrix formula up to a multiplicative error of $(1 \pm \epsilon)$ for $0 < \epsilon < 1$, in addition to the inductively constructed matrix $\mathbf{N}$ of [9], we construct another matrix $\widehat{\mathbf{N}}$ and show that the determinant of the matrix formula is the ratio of $\det(\widehat{\mathbf{N}})$ to $\det(\mathbf{N})$. This then allows us to use the *matrix determinant lemma* (see (3)) to maintain the determinant. Although one might expect that we would require $\log \det(\mathbf{N})$ number of bits for determinant maintenance, we show that $O(s \log(\kappa/\epsilon))$ bits are sufficient. Note that

$\log \kappa$ is preferable to $\log \det$ since, for random matrices, the condition number is polynomial in the dimension of the matrix with high probability [19, 20] while the determinant is exponential [35].

We also consider rank maintenance over finite fields. This is because, under fixed-point arithmetic, we can multiply our matrices by a large number to obtain integer matrices and then perform all operations modulo a sufficiently large prime number ($\text{poly}(n)$ is sufficient). Then the rank of such matrix formula over $\mathbb{Z}_p$ is the same as the rank of the original matrix formula with high probability.

We believe optimizers and algorithm designers can use our results as black boxes to analyze their algorithms under the word-RAM model. The only additional part on their side is to analyze what error bounds can be tolerated in the corresponding algorithm while guaranteeing the returned outputs are correct. Then, our results provide the corresponding running time and bit complexity bounds for the required errors.

Our algorithmic results are presented as dynamic data structures in the next Section 1.1. They cover the most common update schemes occurring in iterative algorithms, such as updating one entry of the matrix and querying one entry or updating a column and querying a row.

Finally, to illustrate the effectiveness of our approach and results, we discuss two example applications. The first one, discussed in the full version, considers finding a basic solution of a set of linear constraints in the standard form $\mathbf{Ax} = \mathbf{b}$, $\mathbf{x} \geq 0$ where $\mathbf{A} \in \mathbb{R}^{d \times n}$ and $n \geq d$. The operations involved in this algorithm are similar to the simplex algorithm. Beling and Megiddo [5] presented two algorithms, a simple one with $O(d^2 n) = O(n^3)$ time, and a more complicated one with $O(d^{1.528} n) = O(n^{2.528})$ time. Both these algorithms assumed the real-RAM model ($O(1)$ time per arithmetic operation with infinite precision). We show that for $n = O(d)$, by simply plugging our data structures into the simple algorithm, the time complexity becomes $\widetilde{O}(n^{2.528} \log(\kappa \cdot \max \det))$ in bit complexity (i.e., number of bit operations). Here $\max \det$ is the maximum determinant over each square $d \times d$ submatrix, and κ is the maximum condition number over each $d \times d$ submatrix. In particular, for matrices with $\log \max \det = \text{poly} \log(n)$ (as is the case when modeling many combinatorial problems as linear programs), our worst-case running time (i.e., number of bit operations) is $\widetilde{O}(n^{2.528} \log \kappa)$. Thus, not only does the simple algorithm become competitive with the more complicated algorithm, but we also show that it can be efficiently implemented without the real-RAM assumptions.

The second example application is for dynamically maintaining the size of the maximum matching of a graph that goes through edge deletion, edge insertion, turning vertices on and off, and merging vertices. We show in the full-version that our rank maintenance data structure can be used for this purpose with a cost of $O(n^{1.405})$ arithmetic operations per update.

1.1 Our Results

Our first result is the following generic data structure that can maintain the value of any matrix formula. Here a matrix formula is any expression that can be written using the basic matrix operations of addition, subtraction, multiplication, and inversion.

▶ **Theorem 1.** *Suppose we are given a matrix formula $f(\mathbf{M}_1, \ldots, \mathbf{M}_s)$ with respective input matrices $\mathbf{M}_1, \ldots \mathbf{M}_s$, where $\|\mathbf{M}_i\|_{\mathrm{F}} \leq \kappa$ for all $i \in [s]$. Let n denote the sum of the number of rows and columns of all $\mathbf{M}_1, \ldots, \mathbf{M}_s$. We further assume that the result of each inversion within f also has Frobenius-norm bounded by κ: in other words, we assume that every*

internal inversion-node of the computation tree has a bounded condition number. Then, for $\epsilon > 0, \kappa > n$*, there exists data structures that are each initialized in time* $\widetilde{O}(n^\omega s \log(\kappa/\epsilon))$ *and have the following operations.*

The data structures have the following update and query operations (where each bullet is a different data structure)

- *Support entry updates and entry queries in* $\widetilde{O}(n^{1.405} s \log(\kappa/\epsilon))$ *time.*
- *Support entry updates in* $\widetilde{O}(n^{1.528} s \log(\kappa/\epsilon))$ *time and entry queries in* $O(n^{0.528} s \log(\kappa/\epsilon))$ *time.*
- *Support column updates and row queries in* $\widetilde{O}(n^{1.528} s \log(\kappa/\epsilon))$ *time.*
- *Support rank-1 updates and returning all entries of* $f(\mathbf{M}_1, ..., \mathbf{M}_s)$ *in* $\widetilde{O}(n^2 s \log(\kappa/\epsilon))$ *time.*
- *Support column updates and row queries in the offline model (the entire sequence of column indices and row queries is given at the start) in* $\widetilde{O}(n^{\omega-1} s \log(\kappa/\epsilon))$ *update and query time.*

The outputs are all ϵ*-approximate, i.e. each entry is off by at most an additive* ϵ*. The stated time complexities depend on current bounds on fast matrix multiplication [37]. The precise dependencies are stated in Theorem 4.*

A similar result was previously proven in [9] using data structures from [33, 12], assuming $O(1)$ time per arithmetic operation and infinite precision. We extend this to the word-RAM model by analyzing the stability of this data structure under the fixed-point arithmetic.

In addition, we show that we can also maintain other properties of $f(\mathbf{M}_1, ..., \mathbf{M}_s)$ while receiving updates to the input matrices. We can maintain the determinant and the rank of $f(\mathbf{M}_1, ..., \mathbf{M}_s)$. The following Theorem 2 and Theorem 3 are proven in the full version.

▶ **Theorem 2.** *Let* $\mathbf{M}_1 \in \mathbb{R}^{n_1 \times d_1}, ..., \mathbf{M}_s \in \mathbb{R}^{n_s \times d_s}$ *and* $n = \sum_{i=1}^{s} n_i + d_i$*. Then, there exists a dynamic determinant data structure that initializes in* $\widetilde{O}(n^\omega s \log(\kappa/\epsilon))$ *time on given accuracy parameters* $\epsilon > 0, \kappa > 2n$*, matrix formula* $f(\mathbf{M}_1, ..., \mathbf{M}_s)$*, and respective input matrices* $\mathbf{M}_1, \ldots, \mathbf{M}_s$*.*

The data structures support the maintenance of $\det(f(\mathbf{M}_1, ..., \mathbf{M}_s))$ *up to a multiplicative factor of* $1 \pm \epsilon$*. They have the following update operations (each bullet is a different data structure)*

- *Support entry updates to any* $\mathbf{M}_i$ *in* $\widetilde{O}(n^{1.405} s \log(\kappa/\epsilon))$ *time.*
- *Support column updates to any* $\mathbf{M}_i$ *in* $\widetilde{O}(n^{1.528} s \log(\kappa/\epsilon))$ *time.*
- *Support rank-1 updates to any* $\mathbf{M}_i$ *in* $\widetilde{O}(n^2 s \log(\kappa/\epsilon))$ *time.*

We assume that throughout all updates, $\|f(\mathbf{M}_1, ..., \mathbf{M}_s)\|_{\mathrm{F}} \leq \kappa$*,* $\|(f(\mathbf{M}_1, ..., \mathbf{M}_s))^{-1}\|_{\mathrm{F}} \leq \kappa$*, and* $\|\mathbf{M}_i\|_{\mathrm{F}} \leq \kappa$ *for all* i*, and the result of each inversion within* f *also has the Frobenius norm bounded by* κ*.*

▶ **Theorem 3.** *There exists a dynamic rank data structure that initializes in* $O(n^\omega)$ *arithmetic operations on given matrix formula* $f(\mathbf{M}_1, ..., \mathbf{M}_s)$*, and respective input matrices. Here,* n *is the sum of the number of rows and columns of all* $\mathbf{M}_1, ..., \mathbf{M}_s$*. The data structure maintains* $\mathrm{rank}(f(\mathbf{M}_1, ..., \mathbf{M}_s))$ *subject to entry updates to any* $\mathbf{M}_i$ *in* $O(n^{1.405})$ *arithmetic operations per update.*

This implies, for example, maintaining the size of the maximum matching in a dynamic graph undergoing edge insertions and deletions, turning vertices on/off, and also merging of vertices (see full version for details). Each such update to the graph takes $O(n^{1.405})$ time. This was previously achieved for edge insertions/deletions only [34, 12].

Theorem 3 gives bounds for rank maintenance over finite fields. It is usually assumed that operations over finite fields take $O(1)$ time. However, a more realistic running time is $\text{poly}\log(|\mathbb{F}|)$ if an isomorphism to polynomials of degree less than d over $\mathbb{Z}_p$ is given, where $|\mathbb{F}| = p^d$ is the size of the field. One approach to go beyond the finite fields for rank maintenance is to perform the operations modulo a random prime, which preserves the rank with constant probability. This has been leveraged in communication complexity literature [36, 23]. See Lemma 4.1 on [36].

1.2 Preliminaries

Notation. We denote matrices with bold uppercase letters and vectors with bold lowercase letters. We denote the Frobenius norm and the operator norm by $\|\cdot\|_{\text{F}}$ and $\|\cdot\|_2$, respectively. We define the condition number of an invertible matrix $\mathbf{M}$ as $\kappa(\mathbf{M}) := \|\mathbf{M}\|_2 \cdot \left\|\mathbf{M}^{-1}\right\|_2$. For simplicity of presentation, we use κ as an upper bound for the Frobenius norm of matrices and their inverses. However, since the condition number of matrices is scale-free, up to polynomial factors such an upper bound is equal to the condition number. When the corresponding matrix is clear from the context, we drop the argument and simply write κ. We denote entry (i, j) of $\mathbf{M}$ by $\mathbf{M}_{i,j}$, row i of $\mathbf{M}$ by $\mathbf{M}_{i:}$ and column j of $\mathbf{M}$ by $\mathbf{M}_{:j}$. For sets I and J, we write $(\mathbf{A})_{I,}$ to denote the rows with indices in I, $(\mathbf{A})_{,J}$ to denote the column with indices in J, and $(\mathbf{A})_{I,J}$ to denote the submatrix with rows with indices in I and columns with indices in J. We denote the $n \times n$ identity matrix by $\mathbf{I}^{(n)}$, and use $\mathbf{0}^{(i,j)}$ to denote the $i \times j$ all-zeros matrix. We denote the transposition of matrix $\mathbf{M}$ by $\mathbf{M}^\top$. We use $\widetilde{O}$ notation to omit polylogarithmic factors in n and polyloglog factors in κ/ϵ from the complexity, i.e., for function f, $\widetilde{O}(f) := O(f \cdot (\log n \cdot \log\log \frac{\kappa}{\epsilon})^c)$, where c is a constant. Further, we denote the set $\{1, \ldots, n\}$ by $[n]$. We denote the number of operations for multiplying an $n^a \times n^b$ matrix with an $n^b \times n^c$ matrix by $O(n^{\omega(a,b,c)})$ and use $O(n^\omega)$ as shorthand for $O(n^{\omega(1,1,1)})$. Finally, for $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $i \in [\min(m, n)]$, let σ_i denote the i'th largest singular value of $\mathbf{A}$.

Sherman-Morrison-Woodbury Identity [38]. Consider an invertible matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$, and matrices $\mathbf{U} \in \mathbb{R}^{n\times r}, \mathbf{D} \in \mathbb{R}^{r\times r}, \mathbf{V} \in \mathbb{R}^{r\times n}$. If $\mathbf{D}$ and $(\mathbf{M} + \mathbf{UDV})^{-1}$ are invertible, then:

$$(\mathbf{M} + \mathbf{UDV})^{-1} = \mathbf{M}^{-1} - \mathbf{M}^{-1}\mathbf{U}(\mathbf{D}^{-1} + \mathbf{VM}^{-1}\mathbf{U})^{-1}\mathbf{VM}^{-1} \tag{1}$$

Schur complement. Consider the block matrix $\mathbf{M}$ given by:

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix},$$

where $\mathbf{A}$ and $\mathbf{D}$ are square matrices. Then if $\mathbf{D}$ is invertible, $\mathbf{M}/\mathbf{D} := \mathbf{A} - \mathbf{BD}^{-1}\mathbf{C}$ is called the *Schur complement* of block $\mathbf{D}$ of matrix $\mathbf{M}$. Similarly, if $\mathbf{A}$ is invertible, $\mathbf{M}/\mathbf{A} := \mathbf{D} - \mathbf{CA}^{-1}\mathbf{B}$ is the Schur complement of block $\mathbf{A}$ of $\mathbf{M}$. The Schur complement gives an inversion formula for block matrices. Particularly, we have the following fact:

Fact. If $\mathbf{A}$ and $\mathbf{M}/\mathbf{A}$ are invertible, then $\mathbf{M}$ is invertible and

$$\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}(\mathbf{M}/\mathbf{A})^{-1}\mathbf{CA}^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{M}/\mathbf{A})^{-1} \\ (\mathbf{M}/\mathbf{A})^{-1}\mathbf{CA}^{-1} & (\mathbf{M}/\mathbf{A})^{-1} \end{bmatrix}.$$

This can be easily verified by multiplication with $\mathbf{M}$.

The Frobenius norm over $\mathbb{R}$ satisfies non-negativity, homogeneity, and the triangle inequality. Specifically, we have that $\|\mathbf{A}\|_{\mathrm{F}} \geq 0$, $\|\beta\mathbf{A}\|_{\mathrm{F}} = |\beta| \|\mathbf{A}\|_{\mathrm{F}}$ and $\|\mathbf{A}+\mathbf{B}\|_{\mathrm{F}} \leq \|\mathbf{A}\|_{\mathrm{F}} + \|\mathbf{B}\|_{\mathrm{F}}$. Finally, when the product $\mathbf{AB}$ is defined, the Frobenius norm obeys the following sub-multiplicative property: $\|\mathbf{AB}\|_{\mathrm{F}} \leq \|\mathbf{A}\|_{\mathrm{F}} \|\mathbf{B}\|_{\mathrm{F}}$.

Our Computational Model. Our algorithms and analysis are under the fixed-point arithmetic. We present all of our analysis under fixed-point arithmetic except for the result of [17] for QR decomposition which is under floating-point arithmetic but we only use that result in a black-box way. In fixed-point arithmetic, each number is represented with a fixed number of bits before and after the decimal point, e.g., under fixed-point arithmetic with 6 bits, we can only present integer numbers less than 64. Addition/subtraction and multiplication of two numbers with n bits can be done in $\widetilde{O}(n)$ time in this model by using fast Fourier transform (FFT) – see [16, Chapter 30]. Division to an additive error of ϵ can also be performed in $\widetilde{O}(n + \log(1/\epsilon))$ time again with the help of FFT. In general, when we mention running time, we mean the number of bit operations. Otherwise, we specify the complexity is about the number of arithmetic operations.

Matrix Formula. Intuitively, a matrix formula is any formula involving matrices and the basic matrix operations of adding, subtracting, multiplying, or inverting matrices. E.g., $f(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}) = (\mathbf{AB}+\mathbf{C})^{-1}\mathbf{D}$ is a matrix formula.

Formally, a matrix formula $f(\mathbf{M}_1,...,\mathbf{M}_s)$ is a directed tree, where each input $\mathbf{M}_i$ is a leaf, and each matrix operation (addition, subtraction, multiplication, inversion) is an internal node. Nodes that represent addition, subtraction, or multiplication have two children, i.e. the two terms that are being added, subtracted, or multiplied. Inversion has only one child, the term being inverted. For example, for node v labeled "+", the subtree rooted at the left child and the subtree rooted at the right child represent formulas $g_{\text{left}}(\mathbf{M}_1,...,\mathbf{M}_k), g_{\text{right}}(\mathbf{M}_{k+1},...,\mathbf{M}_s)$, and the tree rooted at v represents $f(\mathbf{M}_1,...,\mathbf{M}_s) = g_{\text{left}}(\mathbf{M}_1,...,\mathbf{M}_k) + g_{\text{right}}(\mathbf{M}_{k+1},...,\mathbf{M}_s)$.

Note that since a formula is a tree, and not a DAG, and because there is no point in inverting something twice in succession, a formula (i.e., tree) with s input matrices (i.e., leaves) has at most $O(s)$ operations (i.e., internal nodes). Further, note that by formulas being trees, a leaf (input) can be used only once. For example, $(\mathbf{A}+\mathbf{B})\mathbf{A}$ is a formula with 3 inputs.

2 Technical Overview

Here we outline how we obtain our three main results Theorems 1–3. [9] proved a variant of Theorem 1 that assumed the "real-RAM model", i.e., exact arithmetic in only $O(1)$ time per operation. Modern computers do not provide this guarantee unless one uses up to $\Omega(n)$ bit, substantially slowing down each arithmetic operation and the overall algorithm. We show that the techniques by [9] also work with bounded accuracy and much smaller bit complexity. In particular, only $\widetilde{O}(s \log \kappa)$ bits are enough, as stated in Theorem 1.

Before we can outline how to obtain these results, we need to give a brief recap of [9]. This is done in Section 2.1. We outline in Section 2.2 how to prove that $\widetilde{O}(s \log \kappa)$-bit accuracy suffice, resulting in Theorem 1. At last, Section 2.3 outlines how to extend Theorem 1 to maintain determinant and rank of a matrix formula, i.e., prove Theorems 2 and 3.

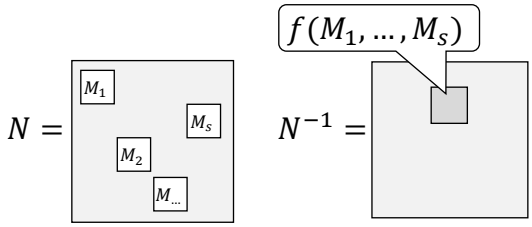

Figure 1 Maintaining $\mathbf{N}^{-1}$ allows us to maintain $f(\mathbf{M}_1, ..., \mathbf{M}_s)$.

2.1 Setting the Stage: How to Maintain Dynamic Algebraic Formulas

Dynamic Matrix Formula is the following data structures task: We are given a formula $f(\mathbf{M}_1, ..., \mathbf{M}_s)$ and respective input matrices $\mathbf{M}_1, ..., \mathbf{M}_s$. The entries of these matrices change over time, and the data structure must maintain $f(\mathbf{M}_1, ..., \mathbf{M}_s)$ (see Theorem 1). *Dynamic Matrix Inverse* is the special case $f(\mathbf{M}) = \mathbf{M}^{-1}$, i.e., given a matrix that changes over time, we must maintain information about its inverse. The latter problem has been studied in, e.g., [33, 12] and there exist several data structures for this task (see Theorem 7).

Previously, [9] showed that the dynamic matrix formula for any formula can be reduced to the special case of matrix inversion, i.e., dynamic matrix inverse. In particular, this means all the previous data structures for dynamic matrix inverse [33, 12] or simple application of the Sherman-Morrison-Woodbury identity (1), can also be used to maintain any general formula $f(\mathbf{M}_1, ..., \mathbf{M}_s)$. [9] shows that for any formula $f(\mathbf{M}_1, ..., \mathbf{M}_s)$, there is a large block matrix $\mathbf{N}$, where $\mathbf{M}_1, ..., \mathbf{M}_s$ occur as subblocks. Further, $\mathbf{N}^{-1}$ contains a block[1] that is precisely $f(\mathbf{M}_1, ..., \mathbf{M}_s)$. See Figure 1. When $\mathbf{M}_1, ..., \mathbf{M}_s$ change over time, matrix $\mathbf{N}$ changes over time, and running a dynamic matrix inverse data structures on this $\mathbf{N}$ allows us to maintain $\mathbf{N}^{-1}$ and its subblock containing $f(\mathbf{M}_1, ...\mathbf{M}_s)$.

The issue of the reduction is that we do not know if $\mathbf{N}$ is well-conditioned. Under which conditions to f and $\mathbf{M}_1, ..., \mathbf{M}_s$ can we guarantee that matrix $\mathbf{N}$ is well-conditioned? Once we can guarantee that $\mathbf{N}$ is well-conditioned, we can give good error guarantees for the dynamic matrix inverse data structures that maintain $\mathbf{N}^{-1}$ via the result of [24] regarding the numerical stability of SMW identity (see full version for details).

We will bound both $\|\mathbf{N}\|_{\mathrm{F}}$ and $\|\mathbf{N}^{-1}\|_{\mathrm{F}}$, which gives a bound on the condition number. In particular, we show that under reasonable assumptions on formula f and input $\mathbf{M}_1, ..., \mathbf{M}_s$, both Frobenius-norms are bounded by $\kappa^{O(s)}$ (Lemma 6). With the dynamic matrix inverse data structures' complexities scaling in the log of these Frobenius norms (see Theorem 7), this leads to the $\widetilde{O}(s \log \kappa)$ factors in Theorem 1.

2.2 Bounding the Frobenius Norms

As outlined in the previous subsection, when given a matrix formula $f(\mathbf{M}_1, ..., \mathbf{M}_s)$, [9] constructs a matrix $\mathbf{N}$ with the following properties. Matrix $\mathbf{N}$ contains $\mathbf{M}_1, ..., \mathbf{M}_s$ as subblock, and the inverse $\mathbf{N}^{-1}$ contains a subblock that is precisely $f(\mathbf{M}_1, ..., \mathbf{M}_s)$, see Figure 1. Our task is to bound the Frobenius-norm of both $\mathbf{N}$ and $\mathbf{N}^{-1}$, which then implies a bound on the condition number of $\mathbf{N}$. Further, data structures for maintaining $\mathbf{N}^{-1}$ have a runtime that scales in those norms (Theorem 7).

[1] The proof is constructive. Given $f, \mathbf{M}_1, ..., \mathbf{M}_s$, we know $\mathbf{N}$ and we know which submatrix of $\mathbf{N}^{-1}$ contains $f(\mathbf{M}_1, ..., \mathbf{M}_s)$, i.e., we do not have to search for the subblock.

Let us briefly recap how matrix $\mathbf{N}$ is constructed, so we can then analyze the Frobenius-norms of $\mathbf{N}$ and $\mathbf{N}^{-1}$.

Construction of N. The given formula $f(\mathbf{M}_1, ..., \mathbf{M}_s)$ can be represented as a tree (where the operations like matrix product, sum, or inversion are nodes, the input matrices are leaves, and the output is the root). The construction of $\mathbf{N}$ follows by induction over the size of the tree: E.g., given some $f(\mathbf{M}_1, ..., \mathbf{M}_s) = g_1(\mathbf{M}_1, ..., \mathbf{M}_k) \cdot g_2(\mathbf{M}_{k+1}, ..., \mathbf{M}_s)$, by induction hypothesis there are matrices $\mathbf{N}_1, \mathbf{N}_2$ where $\mathbf{N}_i$ contains the input matrices of g_i as blocks, and a block of $\mathbf{N}_i^{-1}$ contains the evaluation of f_i. These two matrices are then combined into one larger matrix $\mathbf{N}$ (i.e., $\mathbf{N}$ contains $\mathbf{N}_1, \mathbf{N}_2$ as sub-blocks and thus $\mathbf{N}$ contains $\mathbf{M}_1, ..., \mathbf{M}_s$ as sub-blocks) with the property that $\mathbf{N}^{-1}$ contains a sub-block that is precisely $g_1(\mathbf{M}_1, ..., \mathbf{M}_k) \cdot g_2(\mathbf{M}_{k+1}, ...\mathbf{M}_s)$.

Here we will not go into the precise construction of $\mathbf{N}$ for the different arithmetic operations $f(\mathbf{M}_1, ..., \mathbf{M}_s) = g_1(\mathbf{M}_1, ..., \mathbf{M}_k) \circ g_2(\mathbf{M}_{k+1}, ..., \mathbf{M}_s)$. To follow the outline, it is only important to know that we perform induction by splitting the formula $f = g_1 \circ g_2$ at its root into g_1 and g_2 to obtain two smaller matrices $\mathbf{N}_1, \mathbf{N}_2$. If the root is an inversion, i.e., $f(\mathbf{M}_1, ..., \mathbf{M}_s) = (g(\mathbf{M}_1, ..., \mathbf{M}_s))^{-1}$, then we only have one matrix $\mathbf{N}'$ where $\mathbf{N}'^{-1}$ contains a sub-block that is $g(\mathbf{M}_1, ..., \mathbf{M}_s)$.

Bounding the Frobenius Norm (Section 3.1). For each possible operation $\circ \in \{+, -, \cdot\}$ at the root: $f(\mathbf{M}_1, ..., \mathbf{M}_s) = g_1(\mathbf{M}_1, ..., \mathbf{M}_k) \circ g_2(\mathbf{M}_{k+1}, ..., \mathbf{M}_s)$ there is a different construction for how to combine $\mathbf{N}_1, \mathbf{N}_2$ into a single $\mathbf{N}$, such that $\mathbf{N}^{-1}$ contains $f(\mathbf{M}_1, ..., \mathbf{M}_s)$ as a submatrix. We bound the Frobenius-norm of $\mathbf{N}$ and $\mathbf{N}^{-1}$, with respect to the Frobenius-norms of $\mathbf{N}_1, \mathbf{N}_2$. This then implies a bound by induction. Since the construction of $\mathbf{N}$ differs depending on the operation $\circ \in \{+, -, \cdot\}$, we need slightly different proof for each operation. The proofs will all follow via simple applications of triangle inequalities. Since $\mathbf{N}$ is constructed so that $\mathbf{N}_1, \mathbf{N}_2$ are submatrices of $\mathbf{N}$, simple arguments via triangle inequality suffice. However, for the special case of $f(\mathbf{M}_1, ..., \mathbf{M}_s) = (g(\mathbf{M}_1, ..., \mathbf{M}_s))^{-1}$, a more careful analysis is required, which we outline below.

For $\mathbf{N}'$ being the matrix constructed for formula $g(\mathbf{M}_1, ..., \mathbf{M}_s)$, we have sets $I', J' \subset \mathbb{N}$ where $(\mathbf{N}'^{-1})_{I',J'} = g(\mathbf{M}_1, ..., \mathbf{M}_s)$ (i.e., this is the subblock that contains the evaluation of $g(\mathbf{M}_1, ..., \mathbf{M}_s)$). The reduction by [9] then constructs $\mathbf{N}$ as follows, and we state its inverse:

$$\mathbf{N} = \begin{bmatrix} \mathbf{N}' & -\mathbf{I}^{(n_{N'})}_{,J'} \\ \mathbf{I}^{(n_{N'})}_{I',} & \mathbf{0}^{(n_w, n_w)} \end{bmatrix}$$

$$\mathbf{N}^{-1} = \begin{bmatrix} \mathbf{N}'^{-1} - (\mathbf{N}'^{-1})_{,J'}(\mathbf{N}'^{-1})^{-1}_{I',J'}(\mathbf{N}'^{-1})_{I',} & (\mathbf{N}'^{-1})_{,J'}(\mathbf{N}'^{-1})^{-1}_{I',J'} \\ -(\mathbf{N}'^{-1})^{-1}_{I',J'}(\mathbf{N}'^{-1})_{I',} & ((\mathbf{N}'^{-1})_{I',J'})^{-1} \end{bmatrix}$$

Note that the bottom right block of $\mathbf{N}^{-1}$ is precisely $g(\mathbf{M}_1, ..., \mathbf{M}_s)^{-1}$ since $(\mathbf{N}'^{-1})_{I',J'} = g(\mathbf{M}_1, ..., \mathbf{M}_s)$. To bound the Frobenius-norm of $\mathbf{N}$ and $\mathbf{N}^{-1}$, we apply the triangle inequality to

$$\begin{aligned} \|\mathbf{N}\|_{\mathrm{F}} &\leq \left\| \begin{bmatrix} \mathbf{N}' & 0 \\ 0 & 0 \end{bmatrix} \right\|_{\mathrm{F}} + \left\| \begin{bmatrix} 0 & -\mathbf{I}^{(n_{N'})}_{,J'} \\ \mathbf{I}^{(n_{N'})}_{I',} & 0 \end{bmatrix} \right\|_{\mathrm{F}} \\ &\leq \|\mathbf{N}'\|_F + \sqrt{|J'| + |I'|} \end{aligned}$$

We will have an upper bound on $\|\mathbf{N}'\|_{\mathrm{F}}$ by the inductive hypothesis, so this yields an upper bound on $\|\mathbf{N}\|_{\mathrm{F}}$ as well.

Using the triangle inequality similarly, we also can also bound $\|\mathbf{N}^{-1}\|_{\mathrm{F}}$ by

$$\begin{aligned}\|\mathbf{N}^{-1}\|_{\mathrm{F}} \leq &\left\|\mathbf{N}'^{-1} - (\mathbf{N}'^{-1})_{,J'}(\mathbf{N}'^{-1})^{-1}_{I',J'}(\mathbf{N}'^{-1})_{I',}\right\|_{\mathrm{F}} \\ &+ \left\|(\mathbf{N}'^{-1})_{,J'}(\mathbf{N}'^{-1})^{-1}_{I',J'}\right\|_{\mathrm{F}} + \left\|(\mathbf{N}'^{-1})^{-1}_{I',J'}(\mathbf{N}'^{-1})_{I',}\right\|_{\mathrm{F}} + \left\|(\mathbf{N}'^{-1})^{-1}_{I',J'}\right\|_{\mathrm{F}} \end{aligned} \tag{2}$$

We can split the sum by the triangle inequality, and products using $\|\mathbf{AB}\|_{\mathrm{F}} \leq \|\mathbf{A}\|_{\mathrm{F}}\|\mathbf{B}\|_{\mathrm{F}}$. If we naively upper bound $\|(\mathbf{N}'^{-1})_{I',}\|_{\mathrm{F}}$ and $\|(\mathbf{N}'^{-1})_{,J'}\|_{\mathrm{F}}$ using $\|(\mathbf{N}'^{-1})\|_{\mathrm{F}}$, we will get

$$\begin{aligned}&\|\mathbf{N}'^{-1} - (\mathbf{N}'^{-1})_{,J'}(\mathbf{N}'^{-1})^{-1}_{I',J'}(\mathbf{N}'^{-1})_{I',}\|_{\mathrm{F}} \\ &\quad\leq \|\mathbf{N}'^{-1}\|_{\mathrm{F}} + \|(\mathbf{N}'^{-1})_{,J'}\|_{\mathrm{F}}\|(\mathbf{N}'^{-1})^{-1}_{I',J'}\|_{\mathrm{F}}\|(\mathbf{N}'^{-1})_{I',}\|_{\mathrm{F}} \\ &\quad\leq \|\mathbf{N}'^{-1}\|_{\mathrm{F}} + \|(\mathbf{N}'^{-1})^{-1}_{I',J'}\|_{\mathrm{F}}\|(\mathbf{N}'^{-1})\|^2_{\mathrm{F}}\end{aligned}$$

and similarly for the other three terms of (2). This will yield an upper bound for $\|\mathbf{N}^{-1}\|_{\mathrm{F}}$, but it involves $\|(\mathbf{N}'^{-1})\|^2_{\mathrm{F}}$. In particular, the upper bound gets squared for every nested inverse gate, which will yield a bound that is in the order of κ^{2^s} (with $O(s)$ being a bound on the number of gates).

To improve this, we bound $\|(\mathbf{N}'^{-1})_{I',}\|_{\mathrm{F}}$ and $\|(\mathbf{N}'^{-1})_{,J'}\|_{\mathrm{F}}$ inductively as well. This removes the dependence on $\|(\mathbf{N}'^{-1})\|^2_{\mathrm{F}}$, so the upper bound no longer gets squared in every iteration, and becomes $\kappa^{O(s)}$ instead.

2.3 Dynamic Rank and Determinant of Matrix Formulas

So far, we outlined how to maintain $f(\mathbf{M}_1, ...\mathbf{M}_s)$ within finite precision. This is based on a reduction by [9] from the dynamic matrix formula to the dynamic matrix inverse. We now explain how to extend the reduction, allowing us to also maintain $\det(f(\mathbf{M}_1, ..., \mathbf{M}_s))$ and $\mathrm{rank}(f(\mathbf{M}_1, ..., \mathbf{M}_s))$.

Maintaining the Determinant. First, note that given a block matrix, we can represent its determinant as follows

$$\text{For } \mathbf{M} = \begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C}^\top & \mathbf{D}\end{bmatrix} \text{ we have } \det(\mathbf{M}) = \det(\mathbf{A}) \cdot \det(\mathbf{D} - \mathbf{C}^\top\mathbf{A}^{-1}\mathbf{B}).$$

This allows for the following observation: Given $n \times n$ matrix $\mathbf{N}$ and sets $I, J \subset \mathbb{N}$ with $(\mathbf{N}^{-1})_{I,J} = f(\mathbf{M}_1, ..., \mathbf{M}_s)$, we have $(\mathbf{N}^{-1})_{I,J} = \mathbf{I}^{(n)}_{I,[n]}\mathbf{N}^{-1}\mathbf{I}^{(n)}_{[n],J}$ and thus

$$\widehat{\mathbf{N}} = \begin{bmatrix}\mathbf{N} & \mathbf{I}^{(n)}_{[n],J}\\ \mathbf{I}^{(n)}_{I,[n]} & \mathbf{0}\end{bmatrix} \text{ with } \det(f(\mathbf{M}_1, ..., \mathbf{M}_s)) = \det(\mathbf{I}^{(n)}_{I,[n]}\mathbf{N}^{-1}\mathbf{I}^{(n)}_{[n],J}) = \det(\widehat{\mathbf{N}})/\det(\mathbf{N}).$$

Thus, to maintain $\det(f(\mathbf{M}_1, ..., \mathbf{M}_s))$, we just need to maintain $\det(\widehat{\mathbf{N}})$ and $\det(\mathbf{N})$. Maintaining these determinants can be done via the determinant lemma, which states:

$$\det(\mathbf{N} + \mathbf{uv}^\top) = \det(\mathbf{N})(1 + \mathbf{v}^\top\mathbf{N}^{-1}\mathbf{u}). \tag{3}$$

Here adding $\mathbf{uv}^\top$ is a rank-1 update and can capture updates such as changing an entry of $\mathbf{N}$ (when $\mathbf{u}, \mathbf{v}$ have only 1 nonzero entry each) or changing a column of $\mathbf{N}$ (when v has only 1 nonzero entry). In particular, the task of maintaining $\det(\mathbf{N})$ reduces to the task of repeatedly computing $\mathbf{v}^\top\mathbf{N}^{-1}\mathbf{u}$. This is a dynamic matrix formula (since $\mathbf{u}, \mathbf{v}, \mathbf{N}$ change over time). For example, maintaining $\det(\mathbf{N})$ while $\mathbf{N}$ receives entry updates, requires us to

maintain $f(\mathbf{u}, \mathbf{v}, \mathbf{N}) = \mathbf{v}^\top \mathbf{N}^{-1}\mathbf{u}$ while $\mathbf{u}, \mathbf{v}, \mathbf{N}$ receive entry updates. Thus data structures for maintaining $\det(f(\mathbf{M}_1, ..., \mathbf{M}_s))$ (Theorem 2) are implied by data structures for dynamic matrix formula (Theorem 7), together with some additional error analysis performed in the full-version. A key observation in the error analysis is that the determinant is the product of the eigenvalues, and therefore, if we guarantee (with a sufficient number of bits) that the eigenvalues are preserved up to a multiplicative error factor of $1 \pm \frac{\epsilon}{10n}$, then we have determinant computation up to a multiplicative error factor of $1 \pm \epsilon$. We formalize this idea by bounding the determinant of a matrix perturbed by a small amount – see full version for details.

Maintaining the Rank. Let us assume that $\mathbf{M}_1, ..., \mathbf{M}_s$ are integer matrices, so $\mathbf{N}$ is an integer matrix as well. Note that w.h.p. $\text{rank}(\mathbf{N})$ is the same over $\mathbb{Z}$ and $\mathbb{Z}_p$ for prime $p \sim n^c$ and large enough constant c. So for the rank, we do not need to worry about rounding errors and can just focus on finite fields.

Sankowski [34] proved the following statement about matrix ranks. For any $n \times n$ matrix $\mathbf{M}$, and random $n \times n$ matrices $\mathbf{X}$ and $\mathbf{Y}$ (each entry is chosen uniformly at random from $\mathbb{Z}_p$), and $\mathbf{I}_k$ being a partial identity (the first k diagonal entries are 1, the remaining diagonal entries are 0), let

$$\overline{\mathbf{M}} = \begin{bmatrix} \mathbf{M} & \mathbf{X} & \mathbf{0} \\ \mathbf{Y} & \mathbf{0} & \mathbf{I}_n \\ \mathbf{0} & \mathbf{I}_n & \mathbf{I}_k \end{bmatrix}$$

Then with high probability, $\det(\overline{\mathbf{M}}) \neq 0 \iff \text{rank}(\mathbf{M}) \geq n - k$. In [34], this was used to maintain the rank of $\mathbf{M}$. We now generalize this to maintaining the rank of $f(\mathbf{M}_1, ..., \mathbf{M}_s)$.

Given a formula $f(\mathbf{M}_1, ..., \mathbf{M}_s)$, let $g(\mathbf{M}_1, ..., \mathbf{M}_s, \mathbf{P}, \mathbf{Q}, \mathbf{R}_k) = \mathbf{P} f(\mathbf{M}_1, ..., \mathbf{M}_s)\mathbf{Q} + \mathbf{R}_k$ where

$$\mathbf{P} = \mathbf{Q} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathbf{R}_k = \begin{bmatrix} \mathbf{0} & \mathbf{X} & \mathbf{0} \\ \mathbf{Y} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} & \mathbf{I}_k \end{bmatrix},$$

we have

$$g(\mathbf{M}_1, ..., \mathbf{M}_s, \mathbf{P}, \mathbf{Q}, \mathbf{R}_k) = \begin{bmatrix} f(\mathbf{M}_1, ..., \mathbf{M}_s) & \mathbf{X} & \mathbf{0} \\ \mathbf{Y} & \mathbf{0} & \mathbf{I}_n \\ \mathbf{0} & \mathbf{I}_n & \mathbf{I}_k \end{bmatrix}$$

Thus, $\det(g(\mathbf{M}_1, ..., \mathbf{M}_s, \mathbf{P}, \mathbf{Q}, \mathbf{R}_k)) \neq 0 \iff \text{rank}(f(\mathbf{M}_1, ..., \mathbf{M}_s)) \geq n - k$. So we can track the rank of $f(\mathbf{M}_1, ..., \mathbf{M}_s)$ by finding and maintaining the smallest k where $\det(g(\mathbf{M}_1, ..., \mathbf{M}_s, \mathbf{P}, \mathbf{Q}, \mathbf{R}_k)) \neq 0$. Note that with each changed entry to any $\mathbf{M}_i$, the rank can change by at most 1. So we can simply try increasing/decreasing k by performing a single entry update to $\mathbf{R}_k$ (and potentially reverting the update) to check if the $\det(g(\mathbf{M}_1, ..., \mathbf{M}_s, \mathbf{P}, \mathbf{Q}, \mathbf{R}_k))$ becomes 0. Note that the determinant lemma (3) breaks once the matrix is no longer invertible.

Thus we must increase k whenever $\det(g(\mathbf{M}_1, ..., \mathbf{M}_s, \mathbf{P}, \mathbf{Q}, \mathbf{R}_k)) = 0$. If an update causes the determinant to become 0, we must revert that update by reverting any internal changes of the data structure, then increase k, and then perform the reverted update again. This way, we always guarantee $\det(g(\mathbf{M}_1, ..., \mathbf{M}_s, \mathbf{P}, \mathbf{Q}, \mathbf{R}_k)) \neq 0$ and that (3) never breaks. By always maintaining the largest k where $\det(g(\mathbf{M}_1, ..., \mathbf{M}_s, \mathbf{P}, \mathbf{Q}, \mathbf{R}_k)) \neq 0$, we know $n - k$ is the rank of $f(\mathbf{M}_1, ..., \mathbf{M}_s)$.

3 Dynamic Matrix Formula

In this section, we prove the first main result: a generic data structure that can maintain the evaluation of any matrix formula $f(\mathbf{M}_1, ..., \mathbf{M}_s)$ while supporting updates to the input matrices.

▶ **Theorem 4** (Detailed variant of Theorem 1). *Suppose we are given a matrix formula $f(\mathbf{M}_1, \ldots, \mathbf{M}_s)$ with respective input matrices $\mathbf{M}_1, \ldots \mathbf{M}_s$, where $\|\mathbf{M}_i\|_{\mathrm{F}} \leq \kappa$ for all $i \in [s]$. Let n denote the sum of the number of rows and columns of all $\mathbf{M}_1, \ldots, \mathbf{M}_s$. We further assume that the result of each inversion within f also has Frobenius-norm bounded by κ: in other words, we assume that every internal inversion-node of the computation tree has a bounded condition number. Then, for $\epsilon > 0, \kappa > n$ and any $0 \leq \nu \leq \mu \leq 1$, there exists data structures that are each initialized in time $\widetilde{O}(n^{\omega} s \log(\kappa/\epsilon))$ and have the following operations.*

The data structures have the following update and query operations (where each bullet is a different data structure)

- *Support entry updates and entry queries in $\widetilde{O}((n^{\omega(1,1,\mu)-\mu} + n^{\omega(1,\mu,\nu)-\nu} + n^{\mu+\nu}) s \log(\kappa/\epsilon))$ time. This is $\widetilde{O}(n^{1.405} s \log(\kappa/\epsilon))$ for $\nu \approx 0.543, \mu \approx 0.8612$.*
- *Support entry updates in $\widetilde{O}((n^{\omega(1,1,\mu)-\mu} + n^{1+\mu}) s \log(\kappa/\epsilon))$ time and entry queries in $O(n^{\mu} s \log(\kappa/\epsilon))$ time. This is $\widetilde{O}(n^{1.528} s \log(\kappa/\epsilon))$ and $O(n^{0.528} s \log(\kappa/\epsilon))$ for $\mu \approx 0.528$.*
- *Support column updates and row queries in $\widetilde{O}((n^{\omega(1,1,\mu)-\mu} + n^{1+\mu}) s \log(\kappa/\epsilon))$ time. This is $\widetilde{O}(n^{1.528} s \log(\kappa/\epsilon))$ for $\mu \approx 0.528$.*
- *Support rank-1 updates and returning all entries of $f(\mathbf{M}_1, ..., \mathbf{M}_s)$ in $\widetilde{O}(n^2 s \log(\kappa/\epsilon))$ time.*
- *Support column updates and row queries in the offline model (the entire sequence of column indices and row queries is given at the start) in $\widetilde{O}(n^{\omega-1} s \log(\kappa/\epsilon))$ update and query time.*

The outputs are all ϵ-approximate, i.e. each entry is off by at most an additive ϵ.

The explicit upper bound exponents were obtained via the tool of [7] and use the upper bounds on fast matrix multiplication by [37].

This result is obtained by reducing the task to the special case $g(\mathbf{N}) = \mathbf{N}^{-1}$, where the structure of matrix $\mathbf{N}$ depends on the formula f and its inputs $\mathbf{M}_1, ..., \mathbf{M}_s$.

We then run data structures (Theorem 7) that can maintain $\mathbf{N}^{-1}$ while supporting updates to $\mathbf{N}$. The accuracy of these data structures depends on $\|\mathbf{N}\|_F$ and $\|\mathbf{N}^{-1}\|_F$, so we must bound these Frobenius-norms. These bounds are given in Section 3.1. We then combine the bounds with the previous reduction to obtain Theorem 1 in Section 3.2.

3.1 Norm Bounds on Construction

In this section, we bound the Frobenius norm of the matrix produced by the reduction of [8], as well as its inverse. Formally, [8] has proven the following.

▶ **Theorem 5** (Theorem 3.1 of [9]). *Given a matrix formula $f(\mathbf{A}_1, ..., \mathbf{A}_p)$ over field $\mathbb{F}$, define $n := \sum_{i \in V} n_i + m_i$ where $n_i \times m_i$ is the dimension of $\mathbf{A}_i$.*

Then there exists a square matrix $\mathbf{N}$ of size at most $O(n) \times O(n)$, where each $\mathbf{A}_i$ is a block of $\mathbf{N}$. Further, if $f(\mathbf{A}_1, ..., \mathbf{A}_p)$ does not attempt to invert a non-invertible matrix then $(\mathbf{N}^{-1})_{I,J} = f(\mathbf{A}_1, \ldots, \mathbf{A}_p)$. Constructing $\mathbf{N}$ and I, J takes $O(n^2)$ time.

In the following Lemma 6 we retread the construction of matrix $\mathbf{N}$ and bound the Frobenius-norm.

▶ **Lemma 6.** *Let $f(\mathbf{A}_1, \ldots, \mathbf{A}_p)$ be a matrix formula over $\mathbb{R}$ using s gates. Suppose matrix $\mathbf{N}$, and index sets I, and J are constructed as in Theorem 5 so that $(\mathbf{N}^{-1})_{I,J} = f(\mathbf{A}_1, \ldots, \mathbf{A}_p)$. Let $\kappa \geq \max_i n_i + m_i \geq 2$, where $n_i \times m_i$ are the dimensions of $\mathbf{A}_i$.*

Then, if $\|\mathbf{A}_1\|_{\mathrm{F}}, \ldots, \|\mathbf{A}_p\|_{\mathrm{F}} \leq \kappa$, and the Frobenius norms of outputs of intermediate inverse gates are also bounded by κ, we have

$$\begin{aligned} \|\mathbf{N}\|_{\mathrm{F}} &\leq \kappa^s \\ \|\mathbf{N}^{-1}\|_{\mathrm{F}} &\leq (10\kappa)^{2s+1}. \end{aligned}$$

Proof. We bound $\|\mathbf{N}\|_{\mathrm{F}}$ and $\|\mathbf{N}^{-1}\|_{\mathrm{F}}$ by induction on the number of gates s. Note that the given formula f can be represented as a tree, where the input matrices are leaves and each operation is an internal node. For example $f(\mathbf{M}_1 ... \mathbf{M}_p) = g(\mathbf{M}_1, ..., \mathbf{M}_q) + h(\mathbf{M}_{q+1}, ..., \mathbf{M}_p)$ can be seen as a tree where the root node is a "+" with two subtrees for the formulas g and h. Each node that represents an operation has 2 children (or 1 child in case of inversion). We call the nodes also gates, e.g., inversion gate or addition gate, depending on what operation they represent.

Theorem 5 ([8]) constructs the matrix $\mathbf{N}$ by induction over the number of gates, i.e., for each gate w some matrix $\mathbf{N}_w$ is constructed. This $\mathbf{N}_w$ is constructed as a block matrix where some blocks are $\mathbf{N}_u, \mathbf{N}_v$ where u, v are the child gates of w. We also say that "$\mathbf{N}_w$ is returned by gate w".

Suppose matrix $\mathbf{N}$, and index sets I, and J are constructed as in Theorem 5 so that $(\mathbf{N}^{-1})_{I,J} = f(\mathbf{A}_1, \ldots, \mathbf{A}_p)$. We will show by induction on $s \geq 1$ (the number of gates in f) that:

$$\begin{aligned} \|\mathbf{N}\|_{\mathrm{F}} &\leq \kappa^s \\ \|(\mathbf{N}^{-1})_{I,J}\|_{\mathrm{F}} &\leq 2s\kappa \\ \|(\mathbf{N}^{-1})_{I,}\|_{\mathrm{F}} &\leq (5\kappa)^s \\ \|(\mathbf{N}^{-1})_{,J}\|_{\mathrm{F}} &\leq (5\kappa)^s \\ \|\mathbf{N}^{-1}\|_{\mathrm{F}} &\leq (10\kappa)^{2s+1} \end{aligned}$$

We now prove bounds on the output of our gates by assuming the induction hypothesis that their inputs have bounded norms. We start with the base case.

Input gate. The base case is when $s = 1$, in which case the formula $f(\mathbf{M}_v) = \mathbf{M}_v$ consists of a single input gate. The construction by [8] for Theorem 5 defines $\mathbf{N}$ as

$$\mathbf{N}^{-1} = \mathbf{N} = \begin{bmatrix} \mathbf{I}^{(n_v)} & \mathbf{M}_v \\ \mathbf{0}^{(m_v, n_v)} & -\mathbf{I}^{(m_v)} \end{bmatrix}$$

where $\mathbf{M}_v$ is the input matrix to the formula f and $n_v \times m_v$ are its dimensions. Selecting rows with indices in $I = \{1, ..., m_v\}$, and columns with indices in $J = \{n_v + 1, ..., n_v + m_v\}$, we get

$$\begin{aligned} (\mathbf{N}^{-1})_{,J} &= \begin{bmatrix} \mathbf{M}_v \\ -\mathbf{I}^{(m_v)} \end{bmatrix}, \\ (\mathbf{N}^{-1})_{I,} &= \begin{bmatrix} \mathbf{I}^{(n_v)} & \mathbf{M}_v \end{bmatrix}, \\ (\mathbf{N}^{-1})_{I,J} &= \begin{bmatrix} \mathbf{M}_v \end{bmatrix} \end{aligned}$$

Applying the triangle inequality to

$$\mathbf{N} = \begin{bmatrix} \mathbf{I}^{(n_v)} & \mathbf{0}^{(n_v,m_v)} \\ \mathbf{0}^{(m_v,n_v)} & -\mathbf{I}^{(m_v)} \end{bmatrix} + \begin{bmatrix} \mathbf{0}^{(n_v,n_v)} & \mathbf{M}_v \\ \mathbf{0}^{(m_v,n_v)} & \mathbf{0}^{(m_v,m_v)} \end{bmatrix}$$

and using the fact that $\sqrt{n_v + m_v} \leq \kappa$ and $\|\mathbf{M}_v\|_{\mathrm{F}} \leq \kappa$, we get

$$\begin{aligned} \|\mathbf{N}\|_{\mathrm{F}} &\leq \sqrt{n_v + m_v} + \|\mathbf{M}_v\|_{\mathrm{F}} \\ &\leq 2\kappa \end{aligned}$$

Similarly, we have that

$$\begin{aligned} \|(\mathbf{N}^{-1})_{I,J}\| &\leq \|\mathbf{M}_v\|_{\mathrm{F}} \leq \kappa^1 \\ \|(\mathbf{N}^{-1})_{I,}\| &\leq \|\mathbf{M}_v\|_{\mathrm{F}} + \sqrt{n_v} \leq 2\kappa \leq (5\kappa)^1 \\ \|(\mathbf{N}^{-1})_{,J}\| &\leq \|\mathbf{M}_v\|_{\mathrm{F}} + \sqrt{m_v} \leq 2\kappa \leq (5\kappa)^1 \\ \|\mathbf{N}^{-1}\|_{\mathrm{F}} &\leq \sqrt{n_v + m_v} + \|\mathbf{M}_v\|_{\mathrm{F}} \leq 2\kappa \leq (10\kappa)^3 \end{aligned}$$

We now consider the operation gates for the inductive step.

Inversion. Suppose the root gate is an inversion gate. Suppose $\mathbf{N}'$ is a $n_{N'} \times n_{N'}$ matrix and $I', J' \subset \mathbb{Z}$ are sets, such that $(\mathbf{N}'^{-1})_{I',J'}$ is the $n_w \times n_w$ matrix that the child gate w returns. The child gate is the root of a subtree with $a = s - 1 \geq 1$ gates, which implies bounds on the Frobenius-norm of $\mathbf{N}'$. The construction of Theorem 5 defines

$$\mathbf{N} = \begin{bmatrix} \mathbf{N}' & -\mathbf{I}^{(n_{N'})}_{,J'} \\ \mathbf{I}^{(n_{N'})}_{I',} & \mathbf{0}^{(n_w,n_w)} \end{bmatrix}$$

By block matrix inversion (Section 1.2), we have

$$\mathbf{N}^{-1} = \begin{bmatrix} \mathbf{N}'^{-1} - (\mathbf{N}'^{-1})_{,J'}(\mathbf{N}'^{-1})^{-1}_{I',J'}(\mathbf{N}'^{-1})_{I',} & (\mathbf{N}'^{-1})_{,J'}(\mathbf{N}'^{-1})^{-1}_{I',J'} \\ -(\mathbf{N}'^{-1})^{-1}_{I',J'}(\mathbf{N}'^{-1})_{I',} & (\mathbf{N}'^{-1})^{-1}_{I',J'} \end{bmatrix}$$

Selecting the rows with indices in I, and columns with indices in J, we get

$$\begin{aligned} (\mathbf{N}^{-1})_{I,} &= \begin{bmatrix} -(\mathbf{N}'^{-1})^{-1}_{I',J'}(\mathbf{N}'^{-1})_{I',} & (\mathbf{N}'^{-1})^{-1}_{I',J'} \end{bmatrix} \\ (\mathbf{N}^{-1})_{,J} &= \begin{bmatrix} (\mathbf{N}'^{-1})_{,J'}(\mathbf{N}'^{-1})^{-1}_{I',J'} \\ (\mathbf{N}'^{-1})^{-1}_{I',J'} \end{bmatrix} \\ (\mathbf{N}^{-1})_{I,J} &= \begin{bmatrix} (\mathbf{N}'^{-1})^{-1}_{I',J'} \end{bmatrix} \end{aligned}$$

By the triangle inequality and the assumption that $\kappa \geq n_w \geq 1$,

$$\begin{aligned} \|\mathbf{N}\|_{\mathrm{F}} &\leq \|\mathbf{N}'\|_{\mathrm{F}} + \sqrt{2n_w} \\ &\leq 2a\kappa + 2\kappa \leq 2s\kappa \end{aligned}$$

By the assumption that the output of each inversion gate has Frobenius norm at most κ,

$$\|(\mathbf{N}^{-1})_{I,J}\|_{\mathrm{F}} = \|(\mathbf{N}'^{-1})^{-1}_{I',J'}\|_{\mathrm{F}} \leq \kappa \leq \kappa^s$$

For the remaining matrices, we can bound their Frobenius norms by the sum of Frobenius norms of their blocks, use the triangle inequality to split sums, and $\|\mathbf{AB}\|_{\mathrm{F}} \leq \|\mathbf{A}\|_{\mathrm{F}}\|\mathbf{B}\|_{\mathrm{F}}$ to split products, and then bound the resulting terms by the inductive hypothesis:

$$\begin{aligned}
\|(\mathbf{N}^{-1})_{I,}\|_{\mathrm{F}} &\leq \|(\mathbf{N}'^{-1})^{-1}_{I',J'}\|_{\mathrm{F}}(1 + \|(\mathbf{N}'^{-1})_{I',}\|_{\mathrm{F}}) \\
&\leq \kappa(1 + (5\kappa)^a) \leq \kappa \cdot 2(5\kappa)^a \leq (5\kappa)^{a+1} = (5\kappa)^s \\
\|(\mathbf{N}^{-1})_{,J}\|_{\mathrm{F}} &\leq \|(\mathbf{N}'^{-1})^{-1}_{I',J'}\|_{\mathrm{F}}(1 + \|(\mathbf{N}'^{-1})_{,J'}\|_{\mathrm{F}}) \\
&\leq \kappa(1 + (5\kappa)^a) \leq \kappa \cdot 2(5\kappa)^a \leq (5\kappa)^{a+1} = (5\kappa)^s \\
\|\mathbf{N}^{-1}\|_{\mathrm{F}} &\leq \|\mathbf{N}'^{-1}\|_{\mathrm{F}} + \|(\mathbf{N}'^{-1})^{-1}_{I',J'}\|_{\mathrm{F}}(\|(\mathbf{N}'^{-1})_{I',}\|_{\mathrm{F}} + 1)(\|(\mathbf{N}'^{-1})_{,J'}\|_{\mathrm{F}} + 1) \\
&\leq (10\kappa)^{2a+1} + \kappa((5\kappa)^a + 1)^2 \leq (10\kappa)^{2a+1} + \kappa(10\kappa)^{2a} \\
&\leq (10\kappa)^{2a+3} = (10\kappa)^{2s+1}
\end{aligned}$$

Addition and Subtraction. Suppose the root gate w is an addition gate, adding two $n_w \times m_w$ matrices. Suppose the subtree of the left child has $a \geq 1$ gates and the subtree of the right child has $b \geq 1$ gates, where $s = a + b + 1$. Let $\mathbf{L}$ be the $n_L \times n_L$ matrix and $\mathbf{R}$ be the a $n_R \times n_R$ matrix returned by the child gates, where $(\mathbf{L}^{-1})_{I_L,J_L}$ and $(\mathbf{R}^{-1})_{I_R,J_R}$ are $n_w \times m_w$ matrices. The matrix $\mathbf{N}$ for parent (addition) gate w is defined as

$$\mathbf{N} = \begin{bmatrix} \mathbf{L} & \mathbf{0} & \mathbf{I}^{(n_L)}_{,J_L} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} & \mathbf{I}^{(n_R)}_{,J_R} & \mathbf{0} \\ \mathbf{I}^{(n_L)}_{I_L,} & \mathbf{I}^{(n_R)}_{I_R,} & \mathbf{0} & \mathbf{I}^{(n_w)} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}^{(m_w)} & \mathbf{0} \end{bmatrix}$$

$$\mathbf{N}^{-1} = \begin{bmatrix} \mathbf{L}^{-1} & \mathbf{0} & \mathbf{0} & -(\mathbf{L}^{-1})_{,J_L} \\ \mathbf{0} & \mathbf{R}^{-1} & \mathbf{0} & -(\mathbf{R}^{-1})_{,J_R} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}^{(m_w)} \\ -(\mathbf{L}^{-1})_{I_L,} & -(\mathbf{R}^{-1})_{I_R,} & \mathbf{I}^{(n_w)} & (\mathbf{L}^{-1})_{I_L,J_L} + (\mathbf{R}^{-1})_{I_R,J_R} \end{bmatrix}$$

Selecting the rows of $\mathbf{N}^{-1}$ with indices in I and columns with indices in J, we get

$$(\mathbf{N}^{-1})_{I,} = \begin{bmatrix} -(\mathbf{L}^{-1})_{I_L,} & -(\mathbf{R}^{-1})_{I_R,} & \mathbf{I}^{(n_w)} & (\mathbf{L}^{-1})_{I_L,J_L} + (\mathbf{R}^{-1})_{I_R,J_R} \end{bmatrix}$$

$$(\mathbf{N}^{-1})_{,J} = \begin{bmatrix} -(\mathbf{L}^{-1})_{,J_L} \\ -(\mathbf{R}^{-1})_{,J_R} \\ \mathbf{I}^{(m_w)} \\ (\mathbf{L}^{-1})_{I_L,J_L} + (\mathbf{R}^{-1})_{I_R,J_R} \end{bmatrix}$$

$$(\mathbf{N}^{-1})_{I,J} = \begin{bmatrix} (\mathbf{L}^{-1})_{I_L,J_L} + (\mathbf{R}^{-1})_{I_R,J_R} \end{bmatrix}$$

We now bound the Frobenius norms of these matrices using the bounds on $\|\mathbf{L}\|_{\mathrm{F}}$, $\|\mathbf{L}^{-1}\|_{\mathrm{F}}$, $\|(\mathbf{L}^{-1})_{I_L,}\|_{\mathrm{F}}$, $\|(\mathbf{L}^{-1})_{,J_L}\|_{\mathrm{F}}$, $\|(\mathbf{L}^{-1})_{I_L,J_L}\|_{\mathrm{F}}$, and similarly for $\mathbf{R}^{-1}$, that we get from the inductive hypothesis. Using the triangle inequality and the assumption that $\kappa \geq n_w + m_w$, and that $\|\mathbf{L}\|_{\mathrm{F}} \leq 2a\kappa$ and $\|\mathbf{R}\|_{\mathrm{F}} \leq 2b\kappa$ by the inductive hypothesis,

$$\begin{aligned}
\|\mathbf{N}\|_{\mathrm{F}} &\leq \|\mathbf{L}\|_{\mathrm{F}} + \|\mathbf{R}\|_{\mathrm{F}} + \sqrt{3n_w + 3m_w} \\
&\leq 2a\kappa + 2b\kappa + 2\kappa \leq 2(a + b + 1)\kappa = 2s\kappa
\end{aligned}$$

Similarly, we get:

$$\begin{aligned}
\|(\mathbf{N}^{-1})_{I,J}\|_{\mathrm{F}} &\le \|(\mathbf{L}^{-1})_{I_L,J_L}\|_{\mathrm{F}} + \|(\mathbf{R}^{-1})_{I_R,J_R}\|_{\mathrm{F}} \\
&\le \kappa^a + \kappa^b \le 2\kappa^{a+b} \le \kappa^{a+b+1} = \kappa^s \\
\|(\mathbf{N}^{-1})_{I,}\|_{\mathrm{F}} &\le \|(\mathbf{L}^{-1})_{I_L,}\|_{\mathrm{F}} + \|(\mathbf{R}^{-1})_{I_R,}\|_{\mathrm{F}} + \|(\mathbf{L}^{-1})_{I_L,J_L}\|_{\mathrm{F}} + \|(\mathbf{R}^{-1})_{I_R,J_R}\|_{\mathrm{F}} + \sqrt{n_w} \\
&\le (5\kappa)^a + (5\kappa)^b + \kappa^a + \kappa^b + \kappa \le 5(5\kappa)^{a+b} \le (5\kappa)^{a+b+1} = (5\kappa)^s \\
\|(\mathbf{N}^{-1})_{,J}\|_{\mathrm{F}} &\le \|(\mathbf{L}^{-1})_{,J_L}\|_{\mathrm{F}} + \|(\mathbf{R}^{-1})_{,J_R}\|_{\mathrm{F}} + \|(\mathbf{L}^{-1})_{I_L,J_L}\|_{\mathrm{F}} + \|(\mathbf{R}^{-1})_{I_R,J_R}\|_{\mathrm{F}} + \sqrt{m_w} \\
&\le (5\kappa)^a + (5\kappa)^b + \kappa^a + \kappa^b + \kappa \le 5(5\kappa)^{a+b} \le (5\kappa)^{a+b+1} = (5\kappa)^s
\end{aligned}$$

$$\begin{aligned}
\|\mathbf{N}^{-1}\|_{\mathrm{F}} &\le \|\mathbf{L}^{-1}\|_{\mathrm{F}} + \|\mathbf{R}^{-1}\|_{\mathrm{F}} + \|(\mathbf{L}^{-1})_{I_L,}\|_{\mathrm{F}} + \|(\mathbf{L}^{-1})_{,J_L}\|_{\mathrm{F}} \\
&\qquad + \|(\mathbf{R}^{-1})_{I_R,}\|_{\mathrm{F}} + \|(\mathbf{R}^{-1})_{,J_R}\|_{\mathrm{F}} + \sqrt{n_w + m_w} \\
&\le (10\kappa)^{2a+1} + (10\kappa)^{2b+1} + 2(5\kappa)^a + 2(5\kappa)^b + \kappa \\
&\le 7(10\kappa)^{2a+2b+1} \le (10\kappa)^{2a+2b+3} = (10\kappa)^{2s+1}
\end{aligned}$$

Subtraction gates are the same as addition gates except that the $\mathbf{I}^{(n_R)}_{,J_R}$ in the second row, third column of $\mathbf{N}$ is replaced by $-\mathbf{I}^{(n_R)}_{,J_R}$. The norm-bounding computations are then the same except for irrelevant sign changes.

Multiplication. Suppose the root gate is a multiplication gate. Suppose the left child has $a \ge 1$ gates and the right child has $b \ge 1$ gates, where $s = a + b + 1$. Let $\mathbf{L}$ be the $n_L \times n_L$ matrix and $\mathbf{R}$ be the $n_R \times n_R$ matrix such that the outputs of the child gates are $(\mathbf{L}^{-1})_{I_L,J_L}$ and $(\mathbf{R}^{-1})_{I_R,J_R}$.

$$\begin{aligned}
\mathbf{N} &= \begin{bmatrix} \mathbf{L} & -\mathbf{I}^{(n_L)}_{[n_L],J_L}\mathbf{I}^{(n_R)}_{I_R,[n_R]} \\ \mathbf{0}^{(n_R,n_L)} & \mathbf{R} \end{bmatrix} \\
\mathbf{N}^{-1} &= \begin{bmatrix} \mathbf{L}^{-1} & (\mathbf{L}^{-1})_{,J_L}(\mathbf{R}^{-1})_{I_R,} \\ \mathbf{0} & \mathbf{R}^{-1} \end{bmatrix}
\end{aligned}$$

Selecting the rows of $\mathbf{N}^{-1}$ with indices in I is the same as taking the first row of blocks and left-multiplying by $\mathbf{I}^{(n_L)}_{I_L,}$. Selecting columns with indices in J is the same as taking the second column of blocks and right-multiplying by $\mathbf{I}^{(n_R)}_{,J_R}$. Hence,

$$\begin{aligned}
(\mathbf{N}^{-1})_{I,} &= \begin{bmatrix} (\mathbf{L}^{-1})_{I_L,} & (\mathbf{L}^{-1})_{I_L,J_L}(\mathbf{R}^{-1})_{I_R,} \end{bmatrix} \\
(\mathbf{N}^{-1})_{,J} &= \begin{bmatrix} (\mathbf{L}^{-1})_{,J_L}(\mathbf{R}^{-1})_{I_R,J_R} \\ (\mathbf{R}^{-1})_{,J_R} \end{bmatrix} \\
(\mathbf{N}^{-1})_{I,J} &= \begin{bmatrix} (\mathbf{L}^{-1})_{I_L,J_L}(\mathbf{R}^{-1})_{I_R,J_R} \end{bmatrix}
\end{aligned}$$

We again bound the Frobenius norms of these matrices using the bounds on $\|\mathbf{L}\|_{\mathrm{F}}$, $\|\mathbf{L}^{-1}\|_{\mathrm{F}}$, $\|(\mathbf{L}^{-1})_{I_L,}\|_{\mathrm{F}}$, $\|(\mathbf{L}^{-1})_{,J_L}\|_{\mathrm{F}}$, $\|(\mathbf{L}^{-1})_{I_L,J_L}\|_{\mathrm{F}}$, and similarly for $\mathbf{R}^{-1}$, that we get from the inductive hypothesis. The Frobenius norm of each block matrix is bounded by the sum of the Frobenius norms of its blocks. Using this together with the fact that $\|\mathbf{AB}\|_{\mathrm{F}} \le \|\mathbf{A}\|_{\mathrm{F}}\|\mathbf{B}\|_{\mathrm{F}}$, we get

$$\begin{aligned}
\|\mathbf{N}\|_{\mathrm{F}} &\leq \|\mathbf{L}\|_{\mathrm{F}} + \|\mathbf{R}\|_{\mathrm{F}} + \sqrt{\min(n_L, n_R)} \\
&\leq 5a\kappa + 5b\kappa + \kappa \leq 5(a+b+1)\kappa = 5s\kappa \\
\|(\mathbf{N}^{-1})_{I,J}\|_{\mathrm{F}} &\leq \|(\mathbf{L}^{-1})_{I_L,J_L}\|_{\mathrm{F}}\|(\mathbf{R}^{-1})_{I_R,J_R}\|_{\mathrm{F}} \\
&\leq \kappa^a\kappa^b \leq \kappa^{a+b+1} = \kappa^s \\
\|(\mathbf{N}^{-1})_{I,}\|_{\mathrm{F}} &\leq \|(\mathbf{L}^{-1})_{I_L,}\|_{\mathrm{F}} + \|(\mathbf{L}^{-1})_{I_L,J_L}\|_{\mathrm{F}}\|(\mathbf{R}^{-1})_{I_R,}\|_{\mathrm{F}} \\
&\leq (5\kappa)^a + \kappa^a(5\kappa)^b \leq (5\kappa)^{a+b} + (5\kappa)^{a+b} \leq (5\kappa)^{a+b+1} = (5\kappa)^s \\
\|(\mathbf{N}^{-1})_{,J}\|_{\mathrm{F}} &\leq \|(\mathbf{R}^{-1})_{,J_R}\|_{\mathrm{F}} + \|(\mathbf{R}^{-1})_{I_R,J_R}\|_{\mathrm{F}}\|(\mathbf{L}^{-1})_{,J_L}\|_{\mathrm{F}} \\
&\leq (5\kappa)^b + \kappa^b(5\kappa)^a \leq (5\kappa)^{a+b} + (5\kappa)^{a+b} \leq (5\kappa)^{a+b+1} = (5\kappa)^s \\
\|\mathbf{N}^{-1}\|_{\mathrm{F}} &\leq \|\mathbf{L}^{-1}\|_{\mathrm{F}} + \|\mathbf{R}^{-1}\|_{\mathrm{F}} + \|(\mathbf{L}^{-1})_{,J_L}\|_{\mathrm{F}}\|(\mathbf{R}^{-1})_{I_R,}\|_{\mathrm{F}} \\
&\leq (10\kappa)^{2a+1} + (10\kappa)^{2b+1} + (5\kappa)^a(5\kappa)^b \\
&\leq 3(10\kappa)^{2a+2b+1} \leq (10\kappa)^{2a+2b+3} = (10\kappa)^{2s+1}
\end{aligned}$$

Then by the inductive hypothesis, the claim is proven. ◀

3.2 Proof of Theorem 1

To obtain Theorem 1, we will use the following data structures (Theorem 7) by Sankowski [33], v.d.Brand, Nanongkai and Saranurak [12]. This previous work only considered finite fields or the real-RAM model, i.e., infinite precision with $O(1)$ time per arithmetic operation. In the full version, we prove that these data structures also work with finite precision and $\widetilde{O}(\log(\kappa/\epsilon))$-bit fixed-point arithmetic, as stated in Theorem 7.

▶ **Theorem 7.** *There exist several dynamic matrix inverse algorithms with the following operations. For any update vs. query time trade-off parameters $0 \leq \nu \leq \mu \leq 1$, each data structure initializes in $O(n^\omega \log \kappa/\epsilon)$ time on given accuracy parameters $\epsilon > 0, \kappa > n$, and dynamic matrix $\mathbf{Z} \in \mathbb{R}^{n\times n}$ that is promised to stay invertible throughout all updates with $\|\mathbf{Z}\|_F, \|\mathbf{Z}^{-1}\|_F \leq \kappa$.*

The data structures have the following update/query operations

1. *Support entry updates and entry queries in $\widetilde{O}((n^{\omega(1,1,\mu)-\mu} + n^{\omega(1,\mu,\nu)-\nu} + n^{\mu+\nu})s\log(\kappa/\epsilon))$ time. This is $\widetilde{O}(n^{1.405}s\log(\kappa/\epsilon))$ for $\nu \approx 0.543, \mu \approx 0.8612$.*
2. *Support entry updates in $\widetilde{O}((n^{\omega(1,1,\mu)-\mu} + n^{1+\mu})s\log(\kappa/\epsilon))$ time and entry queries in $O(n^\mu s\log(\kappa/\epsilon))$ time. This is $\widetilde{O}(n^{1.528}s\log(\kappa/\epsilon))$ and $O(n^{0.528}s\log(\kappa/\epsilon))$ for $\mu \approx 0.528$.*
3. *Support rank-1 updates and returning all entries of $f(\mathbf{M}_1, ..., \mathbf{M}_s)$ in $\widetilde{O}(n^2 s\log(\kappa/\epsilon))$ time.*
4. *Support column updates and row queries in the offline model (the entire sequence of column indices and row queries is given at the start) in $\widetilde{O}(n^{\omega-1}s\log(\kappa/\epsilon))$ update and query time.*

The outputs are all ϵ-approximate, i.e., each entry is off by at most an additive $\pm\epsilon$.

By running the data structures of Theorem 7 on the matrix obtained from Theorem 5, we obtain Theorem 1.

Proof of Theorem 4 (Theorem 1). Given the formula $f(\mathbf{M}_1, ..., \mathbf{M}_s)$ and its input matrices $\mathbf{M}_1, ..., \mathbf{M}_s$, where each $\mathbf{M}_i$ is of size $n_i \times m_i$, let $n = \sum_s n_i + m_i$.

Initialization. By Theorem 5, we can construct in $O(n^2)$ time a square $O(n) \times O(n)$ matrix, where each $\mathbf{M}_i$ is a sub-block of $\mathbf{N}$, and two sets $I, J \subset \mathbb{Z}$ with $(\mathbf{N}^{-1})_{I,J} = f(\mathbf{M}_1, ..., \mathbf{M}_s)$.

The assumption of Theorem 4 states that each $\|\mathbf{M}_i\|_F \leq \kappa$ and each result of an inversion gate within f also has Frobenius norm bounded by κ. Thus, by Lemma 6, we have $\log \|\mathbf{N}\|_F$ and $\log \|\mathbf{N}^{-1}\|_F$ bounded by $O(s \log \kappa)$.

Depending on the type of update and query that we want (i.e., entry, column, row, etc), we run the respective data structure from Theorem 7 on $\mathbf{N}$.

In total, the initialization takes $\widetilde{O}(n^\omega s \log(\kappa/\epsilon))$ time, dominated by the initialization of the data structure from Theorem 7.

Updates and Queries. Since each $\mathbf{M}_i$ is a submatrix of $\mathbf{N}$, entry, column, row, or rank-1 updates to any $\mathbf{M}_i$ can be modeled by an entry, column, row, or rank-1 update to $\mathbf{N}$.

Likewise, querying an entry, a row, or column of $f(\mathbf{M}_1, ..., \mathbf{M}_s)$ can be performed by querying an entry, or row, or column of $\mathbf{N}^{-1}$, because submatrix $(\mathbf{N}^{-1})_{I,J} = f(\mathbf{M}_1, ..., \mathbf{M}_s)$. Further, the queries all have accuracy ϵ by Theorem 7.

Thus, the update and query complexity of Theorem 1 is exactly as stated in Theorem 7. ◀

4 Conclusion

We discussed the bit complexity and stability of maintaining arbitrary matrix formulas (with inversion, multiplication, and addition/subtraction) and their determinants. In addition, we provided data structures for maintaining the rank of matrices under finite fields and discussed a few applications for these. We believe our data structures and analysis would provide a useful and easy-to-use toolbox for designing iterative algorithms under the word-RAM model. For example, to extend optimization algorithms to the word-RAM model, one is only required to provide an analysis of what amount of errors in each step can be tolerated without affecting the convergence. Some other applications are in computational geometry and computer algebra problems.

A compelling future direction is to analyze the bit complexity of more complex algorithms that use algebraic and matrix formulas and the required error bounds for these algorithms. One interesting example is the Gram-Schmidt walk introduced for discrepancy minimization [4] that has many applications including experimental design [25]. It is not clear how many bits are required to guarantee such a random walk constructs a good distribution.

Another compelling direction is to investigate whether our bit complexity bounds can be improved for certain problems. For example, in the basic solution application, we presented bounds that depend on both the maximum determinant and maximum condition number over all d-by-d submatrices. We know that the maximum determinant is small for combinatorial problems and the maximum condition number is small for random matrices with high probability. Therefore it would be interesting to investigate whether one of these terms can be eliminated from the bit complexity bound. Another case of special problems is in tensor Tucker decomposition, where linear regression problems with Kronecker structure are solved [18, 21]. Although the matrices involved have a condition number exponential in the order of the tensor, such matrices are usually not constructed explicitly.

Finally, our results hint that inverse maintenance approaches might not be as unstable as previously assumed. It would be interesting to investigate their performance in practice. This might require modifications to plain vanilla Sherman-Morrison-Woodbury identity.

References

1. Deeksha Adil, Rasmus Kyng, Richard Peng, and Sushant Sachdeva. Iterative Refinement for ℓ_p-norm Regression. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1405–1424. SIAM, 2019. doi:10.1137/1.9781611975482.86.
2. Deeksha Adil, Richard Peng, and Sushant Sachdeva. Fast, Provably convergent IRLS algorithm for p-norm Linear Regression. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 14166–14177, 2019. URL: http://papers.nips.cc/paper/9565-fast-provably-convergent-irls-algorithm-for-p-norm-linear-regression.
3. Deeksha Adil and Sushant Sachdeva. Faster p-norm minimizing flows, via smoothed q-norm problems. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 892–910. SIAM, 2020. doi:10.1137/1.9781611975994.54.
4. Nikhil Bansal, Daniel Dadush, Shashwat Garg, and Shachar Lovett. The Gram-Schmidt Walk: A Cure for the Banaszczyk Blues. In *Proceedings of the 50th annual acm sigact symposium on theory of computing*, pages 587–597, 2018. URL: https://theoryofcomputing.org/articles/v015a021/v015a021.pdf.
5. Peter A. Beling and Nimrod Megiddo. Using fast matrix multiplication to find basic solutions. *Theoretical Computer Science*, 205(1):307–316, 1998. doi:10.1016/S0304-3975(98)00003-6.
6. Thiago Bergamaschi, Monika Henzinger, Maximilian Probst Gutenberg, Virginia Vassilevska Williams, and Nicole Wein. New techniques and fine-grained hardness for dynamic near-additive spanners. In *SODA*, pages 1836–1855. SIAM, 2021. URL: https://epubs.siam.org/doi/10.1137/1.9781611976465.110.
7. Jan van den Brand. Complexity term balancer. URL: https://www.ocf.berkeley.edu/~vdbrand/complexity/. Tool to balance complexity terms depending on fast matrix multiplication.
8. Jan van den Brand. A deterministic linear program solver in current matrix multiplication time. In *SODA*, pages 259–278. SIAM, 2020. URL: https://dl.acm.org/doi/abs/10.5555/3381089.3381105.
9. Jan van den Brand. Unifying matrix data structures: Simplifying and speeding up iterative algorithms. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 1–13. SIAM, 2021. arXiv:2010.13888.
10. Jan van den Brand, Yin Tat Lee, Aaron Sidford, and Zhao Song. Solving tall dense linear programs in nearly linear time. In *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 775–788. ACM, 2020. doi:10.1145/3357713.3384309.
11. Jan van den Brand and Danupon Nanongkai. Dynamic approximate shortest paths and beyond: Subquadratic and worst-case update time. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 436–455. IEEE, 2019. arXiv:1909.10850.
12. Jan van den Brand, Danupon Nanongkai, and Thatchaphol Saranurak. Dynamic matrix inverse: Improved algorithms and matching conditional lower bounds. In *FOCS*, pages 456–480. IEEE Computer Society, 2019. arXiv:1905.05067.
13. Sébastien Bubeck, Michael B Cohen, Yin Tat Lee, and Yuanzhi Li. An homotopy method for lp regression provably beyond self-concordance and in input-sparsity time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1130–1137, 2018. arXiv:1711.01328.
14. Li Chen, Rasmus Kyng, Yang P Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–623. IEEE, 2022. URL: https://ieeexplore.ieee.org/document/9996881.

15 Michael B. Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. *J. ACM*, 68(1):3:1–3:39, 2021. doi:10.1145/3424305.

16 Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, third edition edition, 2009. URL: https://mitpress.mit.edu/9780262046305/introduction-to-algorithms/.

17 James Demmel, Ioana Dumitriu, and Olga Holtz. Fast linear algebra is stable. *Numerische Mathematik*, 108(1):59–91, 2007. doi:10.1007/s00211-007-0114-x.

18 Huaian Diao, Rajesh Jayaram, Zhao Song, Wen Sun, and David Woodruff. Optimal sketching for kronecker product regression and low rank approximation. *Advances in neural information processing systems*, 32, 2019.

19 Alan Edelman. Eigenvalues and condition numbers of random matrices. *SIAM journal on matrix analysis and applications*, 9(4):543–560, 1988.

20 Alan Edelman. *Eigenvalues and condition numbers of random matrices*. PhD thesis, Massachusetts Institute of Technology, 1989. URL: https://math.mit.edu/~edelman/homepage/papers/Eig.pdf.

21 Matthew Fahrbach, Gang Fu, and Mehrdad Ghadiri. Subquadratic kronecker regression with applications to tensor decomposition. In *Advances in Neural Information Processing Systems*, 2022.

22 Vissarion Fisikopoulos and Luis Mariano Peñaranda. Faster geometric algorithms via dynamic determinant computation. *Comput. Geom.*, 54:1–16, 2016. URL: https://www.sciencedirect.com/science/article/pii/S0925772115001261.

23 Mehrdad Ghadiri, Yin Tat Lee, Swati Padmanabhan, William Swartworth, David Woodruff, and Guanghao Ye. Improving the bit complexity of communication for distributed convex optimization. In *56th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, 2024.

24 Mehrdad Ghadiri, Richard Peng, and Santosh Vempala. The bit complexity of efficient continuous optimization. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 2023. URL: https://www.computer.org/csdl/proceedings-article/focs/2023/189400c059/1T9796LmQ80.

25 Christopher Harshaw, Fredrik Sävje, Daniel Spielman, and Peng Zhang. Balancing covariates in randomized experiments with the gram-schmidt walk design. *arXiv preprint*, 2019. arXiv:1911.03071.

26 Baihe Huang, Shunhua Jiang, Zhao Song, Runzhou Tao, and Ruizhe Zhang. Solving SDP faster: A robust ipm framework and efficient implementation. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 233–244. IEEE, 2022. URL: https://www.computer.org/csdl/proceedings-article/focs/2022/551900a233/1JtvWgBUn8A.

27 Haotian Jiang, Tarun Kathuria, Yin Tat Lee, Swati Padmanabhan, and Zhao Song. A faster interior point method for semidefinite programming. In *2020 IEEE 61st annual symposium on foundations of computer science (FOCS)*, pages 910–918. IEEE, 2020. URL: https://ieeexplore.ieee.org/document/9317892.

28 Shunhua Jiang, Binghui Peng, and Omri Weinstein. The complexity of dynamic least-squares regression. In *FOCS*, 2023. URL: https://www.computer.org/csdl/proceedings-article/focs/2023/189400b605/1T972gjnp4I.

29 Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. A faster algorithm for solving general LPs. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 823–832, 2021.

30 Yin Tat Lee and Aaron Sidford. Path finding I: Solving linear programs with $\widetilde{O}(\sqrt{\text{rank}})$ linear system solves. *arXiv preprint*, 2013. URL: https://ieeexplore.ieee.org/document/6979027, arXiv:1312.6677.

31 Yiheng Lin, James A Preiss, Emile Timothy Anand, Yingying Li, Yisong Yue, and Adam Wierman. Learning-augmented control via online adaptive policy selection: No regret via contractive perturbations. In *ACM SIGMETRICS, Workshop on Learning-augmented Algorithms: Theory and Applications 2023*, 2023. URL: https://learning-augmented-algorithms.github.io/papers/sigmetrics23-lata-posters-paper5.pdf.

32 Yiheng Lin, James A Preiss, Emile Timothy Anand, Yingying Li, Yisong Yue, and Adam Wierman. Online adaptive policy selection in time-varying systems: No-regret via contractive perturbations. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL: https://openreview.net/forum?id=hDajsofjRM.

33 Piotr Sankowski. Dynamic transitive closure via dynamic matrix inverse (extended abstract). In *FOCS*, pages 509–517. IEEE Computer Society, 2004. URL: https://ieeexplore.ieee.org/document/1366271.

34 Piotr Sankowski. Faster dynamic matchings and vertex connectivity. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 118–126, USA, 2007. Society for Industrial and Applied Mathematics. URL: https://dl.acm.org/doi/10.5555/1283383.1283397.

35 Terence Tao and Van Vu. On random ±1 matrices: singularity and determinant. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 431–440, 2005. URL: https://dl.acm.org/doi/10.1145/1060590.1060655.

36 Santosh S Vempala, Ruosong Wang, and David P Woodruff. The communication complexity of optimization. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1733–1752. SIAM, 2020.

37 Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. In *SODA*, pages 3792–3835. SIAM, 2024.

38 Max A Woodbury. *Inverting modified matrices*. Statistical Research Group, 1950. URL: https://books.google.com/books/about/Inverting_Modified_Matrices.html?id=_zAnzgEACAAJ.