Optimal Fine-Grained Hardness of Approximation of Linear Equations

Mitali Bafna
Harvard University, Cambridge, MA, USA

Nikhil Vyas
MIT, Cambridge, MA, USA

Abstract

The problem of solving linear systems is one of the most fundamental problems in computer science, where given a satisfiable linear system (A, b), for $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$, we wish to find a vector $x \in \mathbb{R}^n$ such that $Ax = b$. The current best algorithms for solving dense linear systems reduce the problem to matrix multiplication, and run in time $O(n^\omega)$. We consider the problem of finding ε-approximate solutions to linear systems with respect to the L_2-norm, that is, given a satisfiable linear system $(A \in \mathbb{R}^{n \times n}, b \in \mathbb{R}^n)$, find an $x \in \mathbb{R}^n$ such that $||Ax - b||_2 \leq \varepsilon ||b||_2$. Our main result is a fine-grained reduction from computing the rank of a matrix to finding ε-approximate solutions to linear systems. In particular, if the best known $\tilde{O}(n^\omega)$ time algorithm for computing the rank of $n \times O(n)$ matrices is optimal (which we conjecture is true), then finding an ε-approximate solution to a dense linear system also requires $\tilde{\Omega}(n^\omega)$ time, even for ε as large as $(1 - 1/\text{poly}(n))$. We also prove (under some modified conjectures for the rank-finding problem) optimal hardness of approximation for sparse linear systems, linear systems over positive semidefinite matrices and well-conditioned linear systems. At the heart of our results is a novel reduction from the rank problem to a decision version of the approximate linear systems problem. This reduction preserves properties such as matrix sparsity and bit complexity.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Linear Equations, Fine-Grained Complexity, Hardness of Approximation

Digital Object Identifier 10.4230/LIPIcs.ICALP.2021.20

Category Track A: Algorithms, Complexity and Games

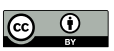
Related Version *Full Version*: `https://arxiv.org/abs/2106.13210`

Funding *Mitali Bafna*: Supported in part by a Simons Investigator Award and NSF Award CCF 1715187.
Nikhil Vyas: Supported by NSF CCF-1909429.

1 Introduction

Algorithms for solving linear equations are one of the most fundamental primitives in computer science. Formally this is the problem where, given a linear system (A, b), where $A \in \mathbb{R}^{m \times n}$ is a real matrix and $b \in \mathbb{R}^m$ is a vector in the column space of A, we need to find a vector $x \in \mathbb{R}^n$ such that $Ax = b$. Gaussian elimination running in time[1] $O(n^3)$ was one of the first algorithms for this problem. Hopcraft and Bunch [9] reduced solving linear equations to fast matrix multiplication of two $n \times n$ matrices [35, 14, 34, 37, 17, 3] which can be done in $m(n) = n^\omega$, where ω is the matrix multiplication constant. The current best known upper bound on ω is approximately 2.372.. [3]. The best known algorithms for solving linear systems reduce the problem to matrix multiplication, but there is no known reduction in the other direction. We study the complexity of finding approximate solutions to linear systems (defined more precisely later) under the following conjecture:

[1] Here we are discussing algorithms and hardness over the Real RAM, unless stated otherwise. We discuss the Word RAM in more detail in Section 1.3.



▶ **Conjecture 1.1** (Rank-Finding Conjecture over RealRAM). *Finding the rank of a matrix $A \in \mathbb{R}^{m\times n}$ with $m = O(n)$ in RealRAM is $\Omega(n^\omega)$-hard.*

The problem of finding the rank of a matrix is a central problem in linear algebra. It is known that this problem can be reduced to fast matrix multiplication [9, 20], hence there exist algorithms for the rank-finding algorithm that run in time $O(n^\omega)$. There are known faster algorithms for restricted classes of matrices. For sparse matrices we know of $O(n^2)$ algorithms [36] and for low-rank matrices the $O(n^2+\text{rank}(A)^\omega)$-time algorithms of [21, 15, 11] run in time $n^{\omega-\Omega(1)}$ when $\text{rank}(A) = n^{1-\Omega(1)}$. No improvement over the $O(n^\omega)$-runtime is known for general matrices though. We conjecture that this run time is in fact *optimal* for general matrices. The rank-finding problem is equivalent to checking whether the determinant of a matrix is 0. We get some evidence towards the truth of our conjecture by considering the computational model of arithmetic circuits[2]: In a seminal work Baur and Strassen [7] linearly reduced the the problem of matrix multiplication to the problem of computing the determinant, thus showing that the latter problem requires arithmetic circuits of size as large as those required for matrix multiplication. Furthermore, this is a central conjecture because falsifying it (getting faster algorithms for the rank-finding problem) would yield better algorithms for important problems like finding the size of a maximum matching in a graph [26, 27]. [3] For some direct evidence: there has been a line of work by Musco et al [28] that gives algorithms to approximate the Schatten p-norms in time better that $O(n^\omega)$ when $p > 0$. But at $p = 0$, the problem of finding the Schatten p-norm is the same as finding the rank of the matrix, and their algorithms run in time $\Omega(n^\omega)$. Hence they are not able to beat the runtime of $O(n^\omega)$ to approximate the rank of a matrix, let alone determine it exactly.

Conjecture 1.1 allows us to study the hardness of linear system solving and related linear algebraic problems in the style of fine-grained complexity [38]. We show that the conjecture directly implies $\tilde{\Omega}(n^\omega)$-hardness of finding exact solutions to linear systems. One could hope to get faster algorithms though when allowed to find an approximate solution to the linear system. In this paper, we consider the problem of finding approximate solutions to linear systems. Specifically, we consider the following notion of approximation:

▶ **Definition 1.2** ($\varepsilon(n)$-Approximate Linear Search). *For a function $\varepsilon : \mathbb{N} \to [0, 1]$, the ε-Approximate Linear Search problem is defined as, given a satisfiable[4] linear system $(A \in \mathbb{R}^{O(n)\times n}, b)$, find an $x \in \mathbb{R}^n$ such that $||Ax - b||_2 \leq \varepsilon(n)||b||_2$.*

Note that the all 0's vector $x = 0^n$ is a 1-approximate solution to any linear system as $||A0^n - b|| = ||b||_2$. Our main result shows that doing barely better than the trivial approximation is hard: $(1 - 1/n^{100})$-Approximate Linear Search i.e. finding an x such that $||Ax - b||_2 \leq (1 - 1/n^{100})||b||_2$. is $\tilde{\Omega}(n^\omega)$ hard under Conjecture 1.1.

Spielman and Teng [32] gave nearly linear-time algorithms ($O(n^2 \log(1/\varepsilon(n)))$-time) for finding $\varepsilon(n)$-approximate solutions to Laplacian linear-systems and this result was built upon by many works [23, 12] to give such algorithms for other restricted classes of linear systems. Our result shows that under the hardness of the rank-finding problem, these algorithms cannot be extended to general linear systems. As mentioned above, we conditionally rule out $\tilde{O}(n^2)$-time algorithms for finding $\varepsilon(n)$-approximate solutions to general linear systems even for $\varepsilon(n) = 1 - 1/n^{100}$.

---

[2] Such a reduction is unknown in the RealRAM model.

[3] This is because maximum matching algorithms has a randomized reduction to the rank-finding problem.

[4] Keeping with the convention of promise problems, we will assume that when given an unsatisfiable instance the algorithm is allowed to output an arbitrary vector.

We also extend our results to give optimal conditional hardness of approximation (under analogous conjectures for the rank-finding problem) for restricted classes of linear systems: sparse linear systems, linear systems over positive-semidefinite matrices, and well-conditioned linear systems.

Recently there has been a lot of progress in relating the exact time-complexities of various problems that have polynomial running times. Although there has been success in a variety of graph-theoretic, geometric and string problems [31, 2, 6], there are very few fine grained reductions from the assumptions therein to linear algebraic problems, an example being, the work of Musco et al [28] that showed conditional lower bounds for spectrum approximation.

The theory of probabilistically checkable proofs [5, 18] was instrumental in proving a host of NP-hardness of approximation results. Though this theory was very successful in settling the time-complexity for approximation problems in NP, there are inherent limitations to extend these techniques to problems in P. Towards this, there has been recent progress for establishing hardness of approximation results for problems in P [1, 10, 22]. Our paper makes further progress in this direction.

1.1 Our results

The table below gives a summary of our hardness results over the RealRAM.

Table 1 ε-ALS refers to ε-Approximate Linear Search Problem. Our hardness results are under different conjectures, see statements for more details. All hardness results are for $\varepsilon = 1 - 1/n^{100}$, while all algorithms are for the much (apriori) harder exact search problem. For small values of condition number better algorithms are known [19] but for our regime of either unbounded or poly(n) condition number the above algorithms are the best known.

Problem	Hardness	Algorithm
ε-ALS	$\tilde{\Omega}(n^\omega)$ (Corollary 4.2)	$O(n^\omega)$ ([9])
Sparse ε-ALS	$\tilde{\Omega}(n^2)$ (Corollary 4.4)	$O(n^2)$([19])
Well-conditioned ε-ALS (Definition 4.13)	$\tilde{\Omega}(n^\omega)$ (Corollary 4.17)	$O(n^\omega)$ ([9])
ε-ALS with PSD matrix	$\tilde{\Omega}(n^\omega)$ (Corollary 4.10)	

We consider the question: Can one get fast approximate linear system solvers for general linear systems that run in time $\tilde{o}(n^\omega)$? We answer this question in the negative, under Conjecture 1.1. In this section we discuss the hardness result for solving general linear systems approximately (first row of Table 1). We discuss the other results of the table in Section 1.2.

We refer to the decision version of the ε-approximate linear search problem as Approximate Linear Decision problem (formally defined in Definition 1.4). We prove all our hardness results by showing a reduction from the rank-finding problem to the Approximate Linear Decision problem. We now state the lemma that proves our main reduction:

▶ **Theorem 1.3** (Main reduction: Informal). *There exists a randomized Turing reduction from the rank-finding problem on $A \in \mathbb{R}^{m\times n}$ to the $(1 - 1/n^{100})$-Approximate linear decision problem*[5] *on square matrices with sparsity $\tilde{O}(nnz(A))$ and dimension $O(\max(m, n))$. The reduction runs in time $\tilde{O}(nnz(A))$ and works with high probability.*

[5] The constant 100 here is arbitrary and in fact our reduction works for all constants.

Our reduction

At the heart of our results lies an "exact to approximate" reduction for *deciding* the satisfiability of linear systems. For showing hardness of finding approximate solutions, we are able to use this philosophy of "increasing the gap" between the YES and NO instances. We consider the following natural decision analogue of the ε-approximate search problem:

▶ **Definition 1.4** ($\varepsilon(n)$-Approximate Linear Decision problem [25]). *For a function $\varepsilon : \mathbb{N} \to [0, 1]$, given a linear system $(A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m)$, with $m = O(n)$ and the promise that it falls into one of the following two sets of instances:*
1. *YES instance: There exists an $x \in \mathbb{R}^n$ such that $Ax = b$.*
2. *NO instances: For all $x \in \mathbb{R}^n$, $||Ax - b||_2 > \varepsilon(n)||b||_2$,*

decide whether (A, b) is a YES instance or a NO instance. We will refer to $\varepsilon(n)$ as the "gap" of the instance.

We show that the rank-finding problem reduces to the $(1 - 1/n^{100})$-approximate linear decision problem described above. We also show that the rank-finding problem is equivalent to the *exact* linear decision problem i.e. the problem of deciding satisfiability of linear systems. Hence our reduction can be interpreted as increasing the gap between the YES/NO cases from almost 0 (can be arbitrarily small as we are working over the RealRAM) to $1 - 1/n^{100}$. We increase this gap in two stages, first to $\varepsilon(n) = 1/n^{O(1)}$ and then to $1 - 1/n^{100}$. This gives conditional $\tilde{\Omega}(n^\omega)$-hardness of the $(1 - 1/n^{100})$-approximate linear decision problem.

Even though the main reduction discussed here is from the rank-finding problem, we are also able to give a search to search reduction from the $1/n^{O(1)}$-approximate linear search problem to the $(1 - 1/n^{100})$-approximate linear search (see Corollary 1.6).

We will now discuss the corollaries of the main reduction outlined above. Given Theorem 1.3, we perform a standard decision to search reduction (Lemma 4.1) to get optimal hardness of approximation for the $(1-1/n^{100})$-approximate *search* problem, under Conjecture 1.1. Thus under the rank finding conjecture, this reduction rules out all $(1-1/n^{100})$-approximation algorithms that run in time $\tilde{o}(n^\omega)$.

▶ **Corollary 1.5** (Informal). *Under Conjecture 1.1, for all constants $c > 0$, the $(1 - 1/n^{100})$-approximate linear search problem is $\tilde{\Omega}(n^\omega)$-hard in the RealRAM model of computation.*

The second step of our reduction can also be used to increase the gap of the $1/n^{O(1)}$-Approximate Linear *Search* problem from $1/n^{O(1)}$ to $1 - 1/n^{100}$. This gives us the following corollary:

▶ **Corollary 1.6** (Search to Search reduction: Informal). *If for any constant a there exists an $\tilde{O}(n^a)$-time algorithm for $(1 - 1/n^{100})$-approximate linear search problem then there exists an $\tilde{O}(n^a)$-time algorithm for the $1/n^{100}$-approximate linear search problem.*

Our hardness result is tight because there exist algorithms which solve the $(1 - 1/n^c)$-Approximate Linear Search problem in $O(n^\omega)$ over the RealRAM. In fact, one can solve the more general problem of linear regression, i.e. given a (possibly unsatisfiable) linear system (A, b), find an x that minimizes $||Ax - b||_2$, in time $O(n^\omega)$.

1.2 Extensions

We prove several extensions of our main theorem using the reduction discussed above. We can modify our main reduction so that it preserves the sparsity and condition number of the original instance to get the results for sparse and well-conditioned linear systems. To get hardness for PSD linear systems we need additional ideas beyond this reduction. We get the following results:

Sparse Linear Systems

Linear equation solving has also been studied in the case of sparse linear systems. We know of $O(\text{nnz}(A)n)$ time algorithms for solving a linear system $(A \in \mathbb{R}^{O(n)\times n}, b)$ where $\text{nnz}(A)$ denotes the number of non-zero entries of A, that use Conjugate Gradient Descent [19], so that when the sparsity of A is $\tilde{O}(n)$, these algorithms run in time $\tilde{O}(n^2)$. Our reduction (discussed above), preserves the sparsity of the original matrix A and thus reduces the exact problem over sparse linear systems to the approximate problem over sparse ones. We start with an analogous conjecture to Conjecture 1.1 for finding the rank of a sparse matrix:

▶ **Conjecture 1.7** (Rank-finding Conjecture for Sparse matrices over RealRAM)**.** *Finding the rank of a matrix $A \in \mathbb{R}^{m\times n}$ with $m = O(n)$ and $nnz(A) = \tilde{O}(n)$ in RealRAM is $\Omega(n^2)$-hard.*

Under the above conjecture we show (see Corollary 4.4) that solving $(1 - 1/n^{100})$-Approximate Linear Search problems on sparse linear systems is $\tilde{\Omega}(n^2)$-hard, which is optimal up to poly-logarithmic factors.

Positive Semi-Definite Linear systems

We give optimal hardness for the $(1 - 1/n^{100})$-approximate linear search problem when the matrix A is restricted to be positive semidefinite (see Corollary 4.10). Recently there has been a lot of work for getting nearly linear-time approximation algorithms for restricted classes of matrices. For a slightly more restricted class of linear systems than PSD ones, called Strongly Diagonally Dominant (SDD) systems, Spielman and Teng gave near-linear time approximate solvers [32], leaving near-linear time approximation algorithms for PSD linear systems as the next open problem. In fact, resolving the time-complexity for PSD linear systems was in mentioned as an open problem in [4], where they gave unconditional hardness for PSD linear systems for sublinear-time algorithms. Interestingly, we show that under Conjecture 4.6, such solvers are not possible for PSD linear systems, thus giving a conditional separation of the time complexity required for approximately solving SDD linear systems versus PSD ones.

Well-conditioned linear systems

We now turn our attention to the problem of solving well-conditioned (polynomially bounded condition number) linear systems approximately. In Section 4.4, we give optimal conditional hardness of approximation for linear systems over matrices with polynomially bounded condition number under the $\tilde{\Omega}(n^\omega)$-hardness of the well-conditioned rank-finding problem.

We also prove the following analogue of Corollary 1.6 which amplifies the gap for the *search* problem on well-conditioned matrices:

▶ **Corollary 1.8.** *If there exists as $\tilde{O}(n^a)$ time algorithm for Well-conditioned $(1 - 1/n)$-Approximate Linear Search then there exists a $\tilde{O}(n^a)$-time algorithm for Well-conditioned $(1/n)$-Approximate Linear Search problem.*

1.3 Reductions over the WordRAM

Our main reduction can be modified to preserve the bit-complexity of the original matrix. In Section 5, we show analogous results for all the problems considered above over the WordRAM. We show that under analogous conjectures for the rank-finding problem over

the WordRAM, the problem of finding $(1 - 1/n^{100})$-approximate solutions to linear systems with bit-complexity $O(\log n)$ is $\tilde{\Omega}(n^\omega)$-hard over the WordRAM. Our hardness result is tight up to polylogarithmic factors because there exist $\tilde{O}(n^\omega)$-time algorithms for exactly solving linear systems on the WordRAM [33, 29, 8].

1.4 Further applications and related work

Recently, there has been a lot of progress on the algorithmic front for finding approximate solutions to restricted classes of linear systems (A, b). In a breakthrough work, Spielman and Teng [32] obtained $\tilde{O}(\text{nnz}(A)\log(1/\varepsilon(n)))$-time algorithms for finding $\varepsilon(n)$-approximate solution to Laplacian systems and Strongly Diagonally Dominant (SDD) systems. This result was followed up by algorithms for more general classes of linear systems such as Connection Laplacians [23] and Directed Laplacian systems [13]. This raised the hope that such approximation algorithms could be obtained for more general classes of matrices such as truss stiffness matrices and total variation matrices. Kyng and Zhang [25] showed that such algorithms for these slightly more general classes would imply approximation algorithms for general linear systems. Therefore, by composing our reduction with theirs, one immediate corollary we get is that solving approximately for these classes of restricted linear systems is as hard as the rank-finding problem.

In [24] the authors prove conditional hardness for the problems of Packing/Covering Linear Programs based on the hardness of approximately solving general linear equations. Prior to our work there was no evidence of hardness for approximately solving linear equations. Our results therefore imply hardness for these problems under the rank-finding problem, which is a more well-studied problem in our opinion.

Organization

In section 2 we introduce notation and basic definitions that will be used throughout the paper. In Section 3 we give a proof of our main reduction (Theorem 3.1). In Section 4, we show conditional hardness for finding approximate solutions to linear systems over the Real RAM. We then extend these conditional hardness results to restricted classes of linear systems: Section 4.2 considers sparse linear systems, Section 4.3 considers positive semidefinite linear systems, and finally Sec 4.4 considers well-conditioned linear systems. In Section 5 we show the analogues of these results over the WordRAM model of computation.

All the proofs are deferred to the full version of the paper.

2 Preliminaries

Below is some notation that will be used throughout:

Notation

We will use $\text{nnz}(A)$ for to denote the sparsity of a matrix A and we will assume that $\text{nnz}(A) \geq \max(n, m)$ for $A \in \mathbb{R}^{m\times n}$. We will call a matrix $A \in \mathbb{R}^{m\times n}$ sparse if $\text{nnz}(A) = \tilde{O}(\max(m, n))$. We will use $\kappa(A)$ to denote the condition number of a matrix A. By bit complexity of A or $\mathcal{B}(A)$ we will refer to maximum bit complexity of any entry in the matrix/vector A. We will use $\leq_T$ to denote Turing reductions. Most of our Turing reductions run in quasi-time linear in the input size. We say $\langle a, b\rangle$ to denote the inner product of a and b i.e. $\sum_i a_i b_i$. We denote $W^\perp$ to denote the subspace orthogonal to the subspace W. By $P_W(b)$ we denote the projection of b on the vector space W. For a matrix M we denote its column space by

colspace(M). By $A^\dagger$ we mean the pseudoinverse of a matrix A. For a matrix $A \in \mathbb{R}^{m\times n}$ by $\Pi_A = A(AA^T)^\dagger A^T$ we mean the linear operator such that for all $x \in \mathbb{R}^m$, $\Pi_A(x)$ is the projection of x on colspace(A). By $g = \tilde{O}(f)$ we mean $g = O(f \cdot \text{polylog}(f))$. By $g = \tilde{\Omega}(f)$ we mean $g = \Omega(f/\text{polylog}(f))$. Whenever not specified by algorithms we mean randomized algorithms. We use w.h.p. to denote a probability of $1 - 1/n^{\log n}$, where n is the input size under consideration.

We refer to the exact version of the $\varepsilon(n)$-Approximate Linear Search problem as the Linear Search Problem. Similarly we refer to the exact version of the $\varepsilon(n)$-Approximate Linear Decision problem as the Linear Decision problem.

3 Proof of Main Reduction (Theorem 1.3)

In this section we will prove the reduction from the rank-finding problem to the approximate version of the linear decision problem. For simplicity, throughout this section we work on the RealRAM model of computation and wherever we do not state it we assume that this is the case, so we *do not* discuss the bit complexity of the reductions. Our reduction can be modified to work on the WordRAM which we do in Section 5.

▶ **Theorem 3.1** (Restatement of Theorem 1.3). *For all constants $c > 0$, there exists a randomized Turing reduction in the RealRAM model of computation, from the rank-finding problem on $A \in \mathbb{R}^{m\times n}$ to the $(1 - 1/n^c)$-Approximate linear decision problem on $(A' \in \mathbb{R}^{n'\times n'}, \mathbf{1}^{n'})$, with dimension $n' = O(\max(m, n))$ and sparsity $\tilde{O}(nnz(A))$, where in the YES case we have the additional property that the matrices produced have full rank. The reduction runs in time $\tilde{O}(nnz(A))$, produces polylog(mn) instances of the approximate linear decision problem and works with high probability.*

We will prove this reduction in three steps:

1. Lemma 3.3: Rank-Finding Problem $\leq_T$ Full Rank problem (see Definition 3.2)
2. Lemma 3.4: Full Rank problem $\leq_T$ $(1/n^{O(1)})$-Approximate linear decision problem
3. Lemma 3.5: $1/n^{O(1)}$-Approximate linear decision problem $\leq$ $(1 - 1/n^c)$-Approximate linear decision problem

Given the lemmas, it will be straightforward to combine these reductions to get that the linear decision problem reduces to the $(1 - 1/n^c)$-Approximate linear decision problem. Let us now show each of the steps stated above. We will prove the first reduction from the rank-finding problem to the full-rank problem. This is similar to a reduction by Wiedemann [36] for finite fields. Let us formally introduce the Full rank problem.

▶ **Definition 3.2** (Full Rank Problem.). *Given a matrix $A \in \mathbb{R}^{n\times n}$, is rank$(A) = n$?*

The intuition behind the following reduction from the rank-finding problem to the full-rank problem is the following: For a matrix $A \in \mathbb{R}^{m\times n}$ with $m > n$ and rank$(A) = k < n$, adding t "random" columns will give a full column rank matrix if and only if $t \geq n - k$. Hence we can binary search for the first value of t which gives us a full column rank matrix, yielding k.

▶ **Lemma 3.3.** *There exists a randomized Turing reduction which works w.h.p. from the Rank-Finding problem on $A \in \mathbb{R}^{m\times n}$ to the Full rank problem, that runs in time $\tilde{O}(nnz(A))$ and produces $O(\log m)$ instances of the Full rank problem, such that all instances of the matrices produced have dimension $O(\max(m, n)) \times O(\max(m, n))$ and sparsity $\tilde{O}(nnz(A))$.*

We will now prove that the full rank problem reduces to the $(1/n^{O(1)})$-Approximate linear decision problem. The idea behind the proof is that for a full-rank square matrix $A \in \mathbb{R}^{n\times n}$ every vector $b \in \mathbb{R}^n$ belongs in the column space of A and hence the linear system (A, b) is satisfiable. On the other hand if $\text{rank}(A) < n$ we expect a random vector b to be outside the column space of A and hence we expect the linear system (A, b) to be unsatisfiable. We show a strengthened version of the previous statement by proving that w.h.p. for a random Gaussian vector b, it holds that for all x, $||Ax - b|| \geq \varepsilon(n)||b||$ for some $\varepsilon(n) = 1/n^{O(1)}$. We also show that we can rescale the rows of the linear system (A, b) to get the system $(A', \mathbf{1}^n)$ while maintaining the property that $||A'x - \mathbf{1}^n|| \geq \varepsilon(n)||\mathbf{1}^n||$. We perform the rescaling to obtain $b = \mathbf{1}^n$ as that will simplify our later reductions.

▶ **Lemma 3.4.** *Consider a matrix $M \in \mathbb{R}^{n\times n}$. There exists a randomized Turing reduction from the problem of checking whether M has full rank to the Linear Decision $(1/n^{O(1)})$-Approximation problem. The reduction runs in time $\tilde{O}(nnz(M))$, produces polylog(n) instances of the form $(M' \in \mathbb{R}^{n\times n}, \mathbf{1}^n)$ where in the YES case M' is a full rank matrix, and works w.h.p.*

We will now show that one can amplify the error in the NO case from $1/n^{O(1)}$ to $1 - 1/n^c$ for all constants c.

The following lemma works for all $\varepsilon(n), \delta(n)$, but we only state it for $\varepsilon(n) = 1/n^{O(1)}, \delta(n) = 1/n^c$ to maintain consistency with the later sections in which we will work over WordRAM.

▶ **Lemma 3.5.** *For all constants c and $\varepsilon(n) = 1/n^{O(1)}, \delta(n) = 1/n^c$, there exists a deterministic many-one reduction from the $\varepsilon(n)$-Approximate linear search problem on the linear system $(A \in \mathbb{R}^{n\times n}, \mathbf{1}^n)$ to the $(1 - \delta(n))$-Approximate linear search problem on the linear system $(A' \in \mathbb{R}^{n\times n}, \mathbf{1}^n)$, with $nnz(A') = O(nnz(A))$. The reduction runs in time $\tilde{O}(nnz(A))$. Additionally if A is full rank then the matrix A' produced is also full rank.*

As this is a deterministic many-one reduction we also get a gap-amplifying reduction for the $\varepsilon(n)$-Approximate linear decision problem with the same parameters.

We can now combine all the lemmas above to get the proof of Theorem 3.1.

Proof of Theorem 3.1. We have proved the following sequence of reductions which preserve sparsity:

1. Lemma 3.3: Linear decision problem $\leq_T$ Full rank problem
2. Lemma 3.4: Full rank problem $\leq_T$ $1/n^{O(1)}$-Approximate linear decision problem
3. Lemma 3.5: $1/n^{O(1)}$-Approximate decision problem $\leq_T$ $(1 - 1/n^c)$-Approximate linear decision problem.

Note that in the YES case of Full-Rank problem we have a square full rank matrix, and this is propagated through Lemma 3.4 and Lemma 3.5 hence the final instance we produce has a full rank matrix in the YES case. Since each of these reductions work whp, one can compose them to get the theorem statement. ◀

4 Hardness of finding L_2-Approximate solutions on Real RAM

In this section, we elaborate on the implications of our main reduction from the previous section (Theorem 3.1). Below is the map of reductions we showed in the previous sections. We will introduce conjectures for the Rank-finding problem in this section and given the reductions, the conjectures will imply conditional hardness of the approximate linear search problem.

All the results in this section are for the RealRAM but can be obtained over the WordRAM too. In Section 5, we prove the conditional hardness of the approximate linear search problem over general matrices in the WordRAM model.

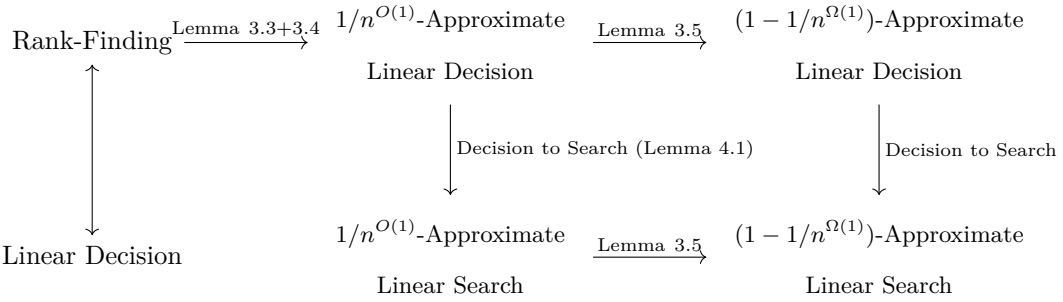


Figure 1 Reductions on RealRAM preserving sparsity (up to polylog(n) factors) and dimension (up to constant factors).

Note that all the lemmas pointed out here (except for the decision to search reduction) were shown in the previous section. The decision to search reduction is straightforward to carry out in the RealRAM and we formally prove it in Lemma 4.1.

In the sections below we discuss the hardness of the approximate linear search problem over general matrices, and then the case of restricted classes of linear systems - sparse linear systems and linear systems given by positive-semidefinite matrices. Our results give tight conditional hardness for all the problems considered.

4.1 General linear systems

We base the hardness result for the approximate linear search problem under the following conjecture for the rank-finding problem:

We now prove the search to decision reduction.

▶ **Reminder of Conjecture 1.1** *Finding the rank of a matrix $A \in \mathbb{R}^{m\times n}$ with $m = O(n)$ in the RealRAM model of computation is $\tilde{\Omega}(n^{\omega})$-hard.*

▶ **Lemma 4.1.** *If there exists a $O(t(n))$ time algorithm to solve ε-Approximate linear search problem for a linear system with sparsity s then there exists a $O(t(n) + sn)$ time algorithm to solve ε-Approximate linear decision problem for linear system with sparsity s.*

Proof. We will follow the standard decision to search reductions which proceed by solving and then confirming. Suppose we are given a linear system (A, b) with $\text{nnz}(A) = s$ for which we want to solve ε-Approximate linear decision problem.

Suppose it is a YES instance i.e. there exists an exact solution, then by the assumed algorithm for ε-Approximate linear search problem we can find an x' such that $||Ax' - b||_2 \leq \varepsilon||b||_2$. This can be done in $O(t(n))$ time.

Suppose instead we were in the NO case i.e. for all x', $||Ax' - b||_2 > \varepsilon||b||_2$ i.e. there exists no x' such that $||Ax' - b||_2 \leq \varepsilon||b||_2$.

Hence checking whether $||Ax' - b||_2 \leq \varepsilon||b||_2$ is true of not for the x' returned by the assumed algorithm for ε-Approximate linear search problem will let us solve the ε-Approximate linear decision problem. We can check if $||Ax' - b||_2 \leq \varepsilon||b||_2$ in time $O(sn)$. Hence the total running time is $O(t(n) + sn)$. ◀

Combining the main reduction (Theorem 3.1) with a decision to search reduction (Lemma 4.1) we get optimal conditional hardness for the $(1-1/n^c)$-approximate linear search problem under Conjecture 1.1:

▶ **Corollary 4.2** (Corollary 1.5 restated). *Under Conjecture 1.1, for all constants $c > 0$, the $(1-1/n^c)$-approximate linear search problem is $\tilde{\Omega}(n^\omega)$ hard in the RealRAM model of computation. Moreover, this remains true even when the matrix A in the given linear system (A, b) is square and has full rank.*

This conditional hardness result is tight as the best algorithms for (exactly) solving general linear systems run in time $\tilde{O}(n^\omega)$ [9].

The next corollary is a search to search reduction between the approximate linear search problem with small gap to one with a larger gap. This is a direct consequence of Lemma 3.5. Recall that in the proof of Lemma 3.1 we used Lemma 3.5 to amplify the gap of the approximate linear *decision* problem. Lemma 3.5 is in fact more general and can amplify the gap of the approximate linear *search* problem too (as noted in the lemma statement):

▶ **Corollary 4.3** (Corollary 1.6 restated). *If for any constant $c > 0$ and a there exists an $\tilde{O}(n^a)$-time algorithm for $(1-1/n^c)$-Approximate Linear Search problem then for all constants d there exists a $\tilde{O}(n^a)$-time algorithms for $1/n^d$-Approximate Linear Search problem.*

Even though this is stated as a reduction, one could potentially use the above corollary to get better algorithms for the $1/n^{O(1)}$-approximate linear search problem.

4.2 Sparse linear systems

In this section, we give analogous results for sparse linear systems. Here we use the fact that our main reduction (Theorem 3.1) preserves the sparsity of the original matrix. Hence if we assume the hardness of the rank-finding problem over sparse matrices, we get conditional hardness for approximately solving sparse linear systems.

▶ **Reminder of Conjecture 1.7** *Finding the rank of a matrix $A \in \mathbb{R}^{m\times n}$, where $m = O(n)$ and $nnz(A) = \tilde{O}(n)$, in the RealRAM model of computation, is $\tilde{\Omega}(n^2)$-hard.*

Combining the main reduction (Theorem 3.1) with a decision to search reduction we get optimal conditional hardness for the $(1-1/n^c)$-approximate linear search problem on sparse matrices, under Conjecture 1.7:

▶ **Corollary 4.4.** *Under Conjecture 1.7, for all constants $c > 0$, the $(1-1/n^c)$-approximate linear search problem (A, b) with $nnz(A) = \tilde{O}(n)$ is $\tilde{\Omega}(n^2)$ hard, in the RealRAM model of computation. Moreover, this remains true even when the matrix A is square and has full rank.*

Note that here we crucially used the fact that the main reduction preserves the sparsity of the original matrix. This conditional hardness result is tight as the best algorithms for (exactly) solving linear equations run in time $\tilde{O}(\text{nnz}(A) \cdot n)$ [19] which is equal to $\tilde{O}(n^2)$ for sparse matrices.

Now we state the search to search reduction for the approximate linear search problem over sparse matrices which follows from Lemma 3.5. This reduction amplifies a small gap to a large gap.

▶ **Corollary 4.5.** *If for any constant $c > 0$ and a there exists an $\tilde{O}(n^a)$-time algorithm for the $(1-1/n^c)$-Approximate Linear Search problem over (A, b) then for all constants d there exists an $\tilde{O}(n^a)$-time algorithm for the $1/n^d$-Approximate Linear Search problem over (A', b) where $nnz(A') = nnz(A)$.*

4.3 Positive semidefinite linear systems

In this section, we show hardness for dense linear systems over PSD matrices. To do so, we need some additional ideas beyond our main reduction and also a conjecture for solving linear systems on matrices with intermediate sparsities. This conjecture also allows us to show optimal conditional hardness of approximately solving linear systems (A, b) for any sparsity.

▶ **Conjecture 4.6** (Rank-finding conjecture for all sparsities)**.** *Finding the rank of a matrix $A \in \mathbb{R}^{m\times n}$, where $m = O(n)$, is $\min(\tilde{\Omega}(nnz(A) \cdot n), \tilde{\Omega}(n^\omega))$ hard in the RealRAM model of computation.*

The current best known algorithms for the Rank-Finding problem ($A \in \mathbb{R}^{O(n)\times O(n)}$) runs in time $\min(\text{nnz}(A)n, n^\omega)$. The above conjecture assumes that this is optimal. We can prove the following theorem directly by combining Conjecture 4.6 and Lemma 5.5.

▶ **Corollary 4.7.** *Under Conjecture 4.6, for all constants $c > 0$, $(1 - 1/n^c)$-approximate linear search problem (A, b), is $\min(\tilde{\Omega}(nnz(A) \cdot n), \tilde{\Omega}(n^\omega))$ hard in the RealRAM model of computation. Moreover, this remains true even when the matrix A in the given linear system (A, b) has full rank.*

The next lemma reduces the approximate linear search problem for intermediate sparsities to approximate linear search problem on dense PSD matrices. It's proof exploits the fact that the algorithm of Yuster and Zwick [39] does matrix multiplication of two matrices A, A' in $O(\min(n^\omega, z^{.7}n^{1.2} + n^2))$ time where $z \geq \text{nnz}(A)$ and $z > \text{nnz}(A')$. This is faster than the best algorithm for solving linear systems (A, b) which runs in $O(\min(n^\omega, zn))$ where $z = \text{nnz}(A)$ for certain sparsities. Specifically we will use the following result from Yuster and Zwick [39]:

▶ **Theorem 4.8** (Yuster and Zwick [39], See Theorem 3.1 and discussion)**.** *Assuming $\omega > 2$ there exists constants $.1 \geq \gamma, \gamma' > 0$ such that for two matrices $A, B \in \mathbb{R}^{O(n)\times O(n)}$ which satisfy $nnz(A), nnz(B) \leq n^{\frac{\omega+1}{2}-\gamma}$ can be multiplied in time $O(n^{\omega-\gamma'})$.*

This allows to do the following reduction:

▶ **Lemma 4.9.** *Assuming $\omega > 2$, there exists constants $.1 \geq \gamma, \gamma' > 0$ and a reduction running in time $O(\max(n^{\omega-\gamma'}, n^{\omega-\gamma}))$ from $(1 - \delta(n))$-approximate linear search problem $(V \in \mathbb{R}^{n\times n}, b)$ with $nnz(V) \leq n^{\frac{\omega+1}{2}-\gamma}$ to $(1 - \delta(n))$-approximate linear search problem (V', b) such that the matrix V' is PSD.*

We now compose the above reduction with Conjecture 4.6 to get the following tight conditional hardness.

▶ **Corollary 4.10.** *Under Conjecture 4.6, for all constants $c > 0$ $(1 - 1/n^c)$-approximate linear search problem (A, b) where A is restricted to be a PSD matrix, is $\tilde{\Omega}(n^\omega)$ hard in the RealRAM model of computation. Moreover, this remains true even when the matrix A in the given linear system (A, b) has full rank.*

4.4 Well-Conditioned Linear Systems

In this section, we show conditional hardness of approximately solving well-conditioned linear systems. The condition number of a full-rank square matrix is the ratio of its maximum and minimum singular values. If the entries of a matrix are all $O(\log n)$-bits then the condition

number of this matrix is at most exponential in n (this is true even for rectangular full column-rank matrices). Therefore linear systems over matrices with polynomially-bounded condition number could be significantly easier to solve than general linear systems.

For the case of certain restricted classes of matrices such as directed Laplacians, the algorithm of Cohen et al [12] for the $\varepsilon(n)$-approximate linear search problem runs in time $\tilde{O}(\text{nnz}(A)\log(\kappa(A)/\varepsilon(n)))$ which is a near-linear time algorithm for $\kappa(A) = \text{poly}(n)$. This is a significant improvement over algorithms for directed Laplacian systems with no bound on the condition number (which run in time $\tilde{O}(n^{\omega})$).

But for general systems no such improvement is known! Conjugate gradient [19] runs in time $\tilde{O}(\text{nnz}(A))$, when $\kappa(A) = \text{poly}\log n$, whereas when $\kappa(A) = \text{poly}(n)$ this algorithm gives *no improvement* over the algorithm for general matrices.

We show that if we assume that the rank-finding problem is hard over well-conditioned matrices ($\kappa(A) = \text{poly}(n)$), then the approximate linear search problem is hard to solve over well-conditioned linear systems. The proof goes along the same lines as that for general matrices: we show in Lemmas 4.15 and 4.16 that our main reduction (Theorem 3.1) in fact preserves the condition number of our original matrix. Then as a corollary we obtain the conditional hardness for approximately solving well-conditioned linear systems. Note that we show all the results here over the RealRAM but they can be easily modified to work on the WordRAM too.

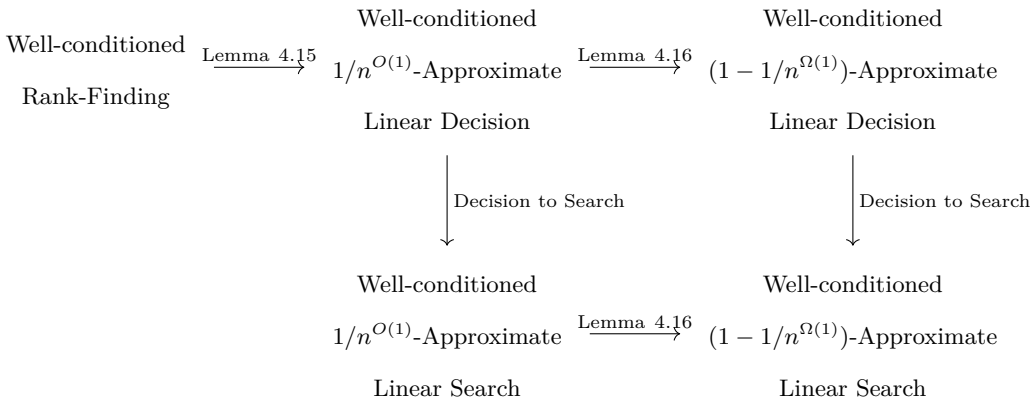


Figure 2 Reductions on the RealRAM preserving sparsity (up to polylog(n) factors), dimension (up to constant factors) and condition-number (upto poly(n) factors). One difference from the results from the previous sections is that we no longer have the equivalence for the Rank-Finding Problem and the Linear Decision problem for Well-conditioned matrices.

Before diving into the reduction, we will formally define all the problems used in the reduction-map above. As discussed in the definition of the condition-number, the condition-number is bounded only when the matrix has full column-rank, therefore in all the definitions below the matrices considered have dimension $m \times n$ with $n \leq m$.

▶ **Definition 4.11** (Well-conditioned Rank-Finding Problem)**.** *Given a matrix $A \in \mathbb{R}^{m\times n}$ where $n \leq m \leq O(n)$, with the promise that it falls into one of the following two sets of instances:*

1. *YES instances: $rank(A) = n$ and $\kappa(A) \leq poly(n)$.*
2. *NO instances: $rank(A) < n$*

decide whether A is a YES instance or a NO instance.

▶ **Definition 4.12** (Well-conditioned Full Column-Rank Problem). *Given a square matrix $A \in \mathbb{R}^{m \times n}$, where $n \leq m$, with the promise that it falls into one of the following two sets of instances:*

1. *YES instances: col-rank$(A) = n$ and $\kappa(A) \leq poly(n)$.*
2. *NO instances: col-rank$(A) < n$*

decide whether A is a YES instance or a NO instance.

Note that the Well-conditioned Rank-finding problem easily reduces to the Well-conditioned Full Column-rank problem. This is because the YES instances of the former always have full column-rank by definition. In fact, we can also reduce the well-conditioned rank-finding problem to the well-conditioned *full-rank* problem on *square* matrices, by adding random columns, since this operation preserves the condition-number [16]. For simplicity of presentation we do not perform this operation and continue to work with full column-rank and (possibly) rectangular matrices throughout.

Next we formally introduce the search and decision problems on well-conditioned linear systems:

▶ **Definition 4.13** (Well-conditioned $\varepsilon(n)$-Approximate Linear Search Problem). *For a function $\varepsilon : \mathbb{N} \to [0, 1]$, given a satisfiable linear system (A, b) with $A \in \mathbb{Z}^{m \times n}$ for $n \leq m$ and $\kappa(A) = poly(n)$ find an assignment x such that $||Ax - b|| \leq \varepsilon(n)||b||$*

▶ **Definition 4.14** (Well-Conditioned $\varepsilon(n)$-Approximate Linear Decision problem). *For a function $\varepsilon : \mathbb{N} \to [0, 1]$, given a linear system $(A \in \mathbb{Z}^{m \times n}, b \in \mathbb{Z}^n)$ for $n \leq m$, with the promise that it falls into one of the following two sets of instances:*

1. *YES instance: There exists an $x \in \mathbb{Q}^n$ such that $Ax = b$ and A is well-conditioned.*
2. *NO instances: For all $x \in \mathbb{R}^n$, $||Ax - b||_2 > \varepsilon(n)||b||_2$,*

decide whether (A, b) is a YES instance or a NO instance.

We will now show our main reduction from the Well-conditioned Rank-finding problem to the Well-conditioned $(1 - 1/n^c)$-approximate linear decision problem. As noted above the Well-conditioned Rank-finding problem reduces to the Well-conditioned Full Column-Rank problem, so we will show a reduction from the latter to the approximate linear decision problem. To do so, we will show that the proofs in Section 3 preserve the condition-number of the original matrix.

Let us start with showing that "well-conditioned" property is preserved in the reduction in Lemma 3.4.

▶ **Lemma 4.15.** *There exists a randomized Turing reduction from the Well-conditioned Full Column-Rank Problem on $M \in \mathbb{Z}^{m \times n}$ to the Well-conditioned $(1/n^{O(1)})$-Approximate Linear Decision problem. The reduction produces polylog(n) instances of the form $(M', \mathbf{1}^n)$ where $M' \in \mathbb{Z}^{m \times n}$ and $\kappa(M') = poly(n)$, runs in time $\tilde{O}(nnz(M))$, and works w.h.p.*

Next let us show that the "well-conditioned" property is preserved in the reduction in Lemma 3.5.

▶ **Lemma 4.16.** *For all constants $c, d > 0$, there exists a deterministic many-one reduction from the Well-conditioned $1/n^d$-Approximate linear search problem on the linear system $(A \in \mathbb{Z}^{m \times n}, \mathbf{1}^n)$ to the Well-conditioned $(1 - 1/n^c)$-Approximate linear search problem on the linear system $(A' \in \mathbb{Z}^{n \times n}, \mathbf{1}^n)$, with $nnz(A') = O(nnz(A))$ and $\kappa(A') = poly(n)$.*

As this is a deterministic many-one reduction we also get a gap-amplifying reduction for the $\varepsilon(n)$-Approximate linear decision problem with the same parameters.

Combining the two lemmas above we get the well-conditioned analogue of the main reduction from the Rank-finding problem to the $(1 - 1/n^c)$-approximate linear decision problem. We can now apply a decision to search reduction to get conditional hardness for approximately solving linear systems over well-conditioned matrices:

▶ **Corollary 4.17.** *For all constants c, assuming $\tilde{\Omega}(n^\omega)$ hardness of the well-conditioned rank-finding problem we get that the well-conditioned $(1 - 1/n^c)$-approximate linear search problem is $\tilde{\Omega}(n^\omega)$-hard.*

We also state the following search to search reduction which follows directly from Lemma 4.16:

▶ **Corollary 4.18.** *For all constants $a, c, d > 0$, if there exists an $\tilde{O}(n^a)$ time algorithm for well-conditioned $(1 - 1/n^c)$-approximate linear search then there exists an $\tilde{O}(n^a)$-time algorithm for the well-conditioned $(1/n^d)$-approximate linear search problem.*

5 Hardness of finding L_2-Approximate solutions on Word RAM

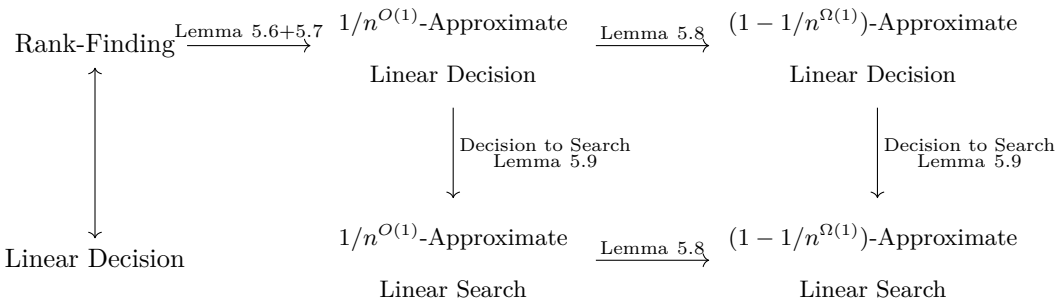


Figure 3 Reductions on WordRAM preserving sparsity (up to polylog(n) factors), dimension (up to constant factors) and bit complexity of entries (up to constant factors).

In Section 4 we gave conditional hardness for solving linear equations in RealRAM. In this section we will give hardness for WordRAM through some modifications of the reduction for RealRAM. We will assume that for a matrix/vector of dimension $m \times n$ in this section that all input entries are from $\mathbb{Z}$ and have $O(\log mn)$ bits in.

The reduction in RealRAM does not directly work for WordRAM as:

1. As we are in WordRAM we need to argue that the final problem instance has bounded bit complexity if the starting problem has bounded bit complexity. To verify this we show that this is true for all the steps of the reduction.
2. We sampled a random gaussian vector in the reduction (Lemma 3.4), this is not possible in the WordRAM. We will get around this issue (in Lemma 5.7) by sampling a random vector whose each entry is a random integer from a predefined range.
3. The decision to search reduction for RealRAM (Lemma 4.1) was nearly trivial. This is not the case for WordRAM as the solution can have large bit complexity and hence given a solution directly substituting to check if it is a good solution or not may require too much time. We give an alternative decision to search reduction in WordRAM in Lemma 5.9.

We start by defining the Linear Decision Problem over Word-RAM:

▶ **Definition 5.1** (Linear Decision Problem over Word-RAM). *Given a linear system (A, b) with $A \in \mathbb{Q}^{m\times n}$ with $\mathcal{B}(A) = O(\log mn)$, distinguish between the following two sets of instances:*

- *YES instances: There exists an x such that $Ax = b$.*
- *NO instances: For all x, $Ax \neq b$.*

We will consider the following conjecture (analogous to Conjecture 1.1) for the Word RAM:

▶ **Conjecture 5.2** (Rank-Finding Conjecture on WordRAM). *There exists no $\tilde{o}(n^\omega)$-time randomized algorithm for finding the rank of a matrix $A \in \mathbb{Z}^{m\times n}$ with $m = O(n)$ and $\mathcal{B}(A) = O(\log n)$, in the WordRAM model of computation.*

The conjecture is tight as using an easy randomized reduction [33] Rank over integers with $O(\log n)$ bit complexity can be reduced to rank over finite fields GF(poly(n)) which gives a $\tilde{O}(n^\omega)$ time randomized algorithm on WordRAM.

We will prove the following main theorem:

▶ **Corollary 5.3** (Hardness of solving linear equation on WordRAM). *For all constant $c > 0$, under Conjecture 1.1, there does not exist a $\tilde{o}(n^\omega)$ time randomized algorithm for the $(1 - 1/n^c)$-approximate linear search problem (A, b) with $\mathcal{B}(A) = O((c + 1)\log n)$ in the WordRAM model of computation. Moreover, this remains true even when the matrix A in the given linear system (A, b) has full rank.*

The conditional hardness in the above theorem is tight as there exists $O(n^\omega)$ time algorithms for exactly solving linear equations over $\mathbb{Z}$ in WordRAM [33, 29, 8].

We are also able to establish the following corollary of one of the intermediate steps in our reduction (Lemma 5.8) which reduces approximately solving linear systems with low error to approximately solving linear systems with barely non-trivial error.

▶ **Corollary 5.4.** *For all constant $c, d > 0$, if over WordRAM there exists as $\tilde{O}(n^a)$ time algorithm for $(1-1/n^c)$-Approximate Linear Search then there exists a $\tilde{O}(n^a)$-time algorithms for $(1/n^d)$-Approximate Linear Search problem.*

Analogous to Theorem 3.1 we will prove a reduction from the rank finding problem to approximate linear decision problem from which Corollary 5.3 will follow by a decision to search reduction (Lemma 5.9). Formally,

▶ **Lemma 5.5** (Reduction from exact to approximate). *For all constant c, there exists a randomized Turing reduction from the rank-finding problem on $A \in \mathbb{Z}^{m\times n}$ to the $(1 - 1/n^c)$-Approximate linear decision problem on $n' \times n'$-square matrices with bit complexity $\mathcal{B}(A) = O((c + 1)\log(n))$, with $n' = O(\max(m, n))$, where in the YES case we have the additional property that the matrices produced have full rank. The reduction runs in time $\tilde{O}(c \cdot nnz(A))$, produces polylog$(mn)$ instances of the approximate linear decision problem and works w.h.p..*

To prove the above Lemma, we will go through each of the steps in the proof of Theorem 1.3 and see that all of them work for WordRAM with small modifications. We start with the reduction the Rank-Finding Problem to the Full rank problem for WordRAM (analogous to Lemma 3.3 for RealRAM).

▶ **Lemma 5.6.** *There exists a randomized Turing reduction which works w.h.p. from the Rank-Finding problem on $A \in \mathbb{Z}^{m\times n}$ with $m = O(n)$ and $\mathcal{B}(A) = O(\log n)$, to the Full rank problem, that runs in time $\tilde{O}(nnz(A))$ and produces $O(\log m)$ instances of the Full rank problem, such that all instances of the matrices produced have dimension $O(\max(m, n)) \times O(\max(m, n))$, sparsity $\tilde{O}(nnz(A))$ and bit complexity $O(\log n)$.*

Analogous to Lemma 3.4, we now reduce the Full-Rank Problem to the $(1/n^{O(1)})$-Linear Decision Approximation problem on WordRAM.

▶ **Lemma 5.7.** *Consider a matrix $M \in \mathbb{Z}^{n\times n}$. There exists a randomized Turing reduction from the problem of checking whether M has full rank to the $(1/n^{12})$-Linear Decision Approximation problem. The reduction runs in time $\tilde{O}(nnz(M))$, produces polylog(n) instances of the form $(M', \mathbf{1}^n)$ where $M' \in \mathbb{Z}^{n\times n}, \mathcal{B}(M') = O(\mathcal{B}(M) + (\log n))$, in the YES case M' is a full rank matrix, and works w.h.p..*

Finally we reduce the $(1/n^{O(1)})$-Linear Decision Approximation problem to the $(1-1/n^c)$-Linear Decision Approximation problem over WordRAM. This is straightforward to work out from the analogous Lemma 3.5 in Section 4.

▶ **Lemma 5.8.** *For all constants c and $\varepsilon(n) = 1/n^{O(1)}, \delta(n) = 1/n^c$, there exists a deterministic many-one reduction from the $\varepsilon(n)$-Approximate linear search problem on the linear system $(A \in \mathbb{R}^{n\times n}, \mathbf{1}^n)$ to the $(1-\delta(n))$-Approximate linear search problem on the linear system $(A' \in \mathbb{R}^{n\times n}, \mathbf{1}^n)$, with $nnz(A') = O(nnz(A))$. The reduction runs in time $\tilde{O}(nnz(A))$. Additionally if A is full rank then the matrix A' produced is also full rank.*

As this is a deterministic many-one reduction we also get a gap-amplifying reduction for the $\varepsilon(n)$-Approximate linear decision problem with the same parameters.

Now we are ready to prove Lemma 5.5.

Proof of Lemma 5.5. Similar to the proof of Theorem 1.3, The proof follows by composing Lemma 5.6, Lemma 5.7 and Lemma 5.8 (for $\varepsilon(n) = 1/n^{O(1)}$ and $\delta(n) = 1/n^c$). The bit complexity of the final matrix is $O(\log n) + O(\log(n/(\varepsilon(n)\delta(n)))) = O((c+1)(\log n))$ and the running time is $\tilde{O}(c \cdot \text{nnz}(A))$ ◀

To prove Corollary 5.3 we need the following decision to search reduction:

▶ **Lemma 5.9.** *Let $\varepsilon(n) = 1/n^{O(1)}$, given an $A \in \mathbb{Z}^{m\times n}, x, b$ where $m = O(n), \mathcal{B}(A), \mathcal{B}(b) =$ polylog$(n), \mathcal{B}(x) = \tilde{O}(n)$ we can distinguish between:*
1. $||Ax - b|| \le \varepsilon(n)||b||/2$.
2. $||Ax - b|| \ge \varepsilon(n)||b||$.

w.h.p. in time $\tilde{O}(nnz(A)n)$.

Combining Lemma 5.5 and Lemma 5.9 give us Corollary 5.3:

Proof of Corollary 5.3. For all constants c, Composing Conjecture 5.2 and Lemma 5.5 gives us $\tilde{\Omega}(n^\omega)$ hardness of $(1-1/n^c)$-Approximate linear decision problem $(A \in \mathbb{Z}^{n\times n}, b)$ with bit complexity $O((c+1)\log(n))$ where in the YES case we have the additional property that A has full rank. The hardness of $(1-1/n^c)$-Approximate linear search problem with the same properties follows from the decision to search reduction from Lemma 5.9. ◀

We now prove Corollary 5.4:

Proof of Corollary 5.4. Note that the input size is $\tilde{O}(n^2)$ and hence $a \ge 2$. The corollary directly follows from noting that Lemma 5.8 applied for $\varepsilon(n) = 1/n^d$ and $\delta(n) = 1/n^c$ reduces $(1/n^d)$-Approximate Linear Search problem to $(1-1/n^d)$-Approximate Linear Search problem in time $\tilde{O}(n^2) = \tilde{O}(n^a)$. ◀

5.1 Sparse Matrices

Starting from the WordRAM version of Conjecture 1.7 and using Lemma 5.5 we can establish that there does not exist a $\tilde{o}(n^2)$ algorithm for $(1 - 1/\text{poly}(n))$-Approximate linear decision problem on sparse matrices in the WordRAM model. By the decision to search reduction in Lemma 5.9 we get that there does not exist a $\tilde{o}(n^2)$ algorithm for $(1 - 1/\text{poly}(n))$-Approximate linear decision search on sparse matrices in the WordRAM model. Note though that this hardness is trivial to obtain since there exist sparse linear systems such that every $(1 - 1/\text{poly}(n))$-approximate solution to the system requires $\Omega(n^2)$ bits to represent.

On the algorithmic side no improvement over the dense case algorithmic runtime of $O(n^\omega)$ was known until the recent result of Peng and Vempala [30] who gave an asymptotically faster algorithm for the $1/\text{poly}(n)$-approximate linear search problem.

References

1 Amir Abboud, Aviad Rubinstein, and R. Ryan Williams. Distributed PCP theorems for hardness of approximation in P. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 25–36. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.12.

2 Josh Alman and Ryan Williams. Probabilistic polynomials and hamming nearest neighbors. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 136–150. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.18.

3 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. *CoRR*, abs/2010.05846, 2020. arXiv:2010.05846.

4 Alexandr Andoni, Robert Krauthgamer, and Yosef Pogrow. On solving linear systems in sublinear time. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 3:1–3:19, 2019.

5 Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *J. ACM*, 45(3):501–555, 1998. doi:10.1145/278298.278306.

6 Arturs Backurs and Piotr Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). *SIAM J. Comput.*, 47(3):1087–1097, 2018. doi:10.1137/15M1053128.

7 Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theor. Comput. Sci.*, 22:317–330, 1983. doi:10.1016/0304-3975(83)90110-X.

8 Stavros Birmpilis, George Labahn, and Arne Storjohann. Deterministic reduction of integer nonsingular linear system solving to matrix multiplication. In *Proceedings of the 2019 on International Symposium on Symbolic and Algebraic Computation, ISSAC 2019, Beijing, China, July 15-18, 2019*, pages 58–65, 2019.

9 James R Bunch and John E Hopcroft. Triangular factorization and inversion by fast matrix multiplication. *Mathematics of Computation*, 28(125):231–236, 1974.

10 Lijie Chen, Shafi Goldwasser, Kaifeng Lyu, Guy N. Rothblum, and Aviad Rubinstein. Fine-grained complexity meets IP = PSPACE. 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 1–20. SIAM, 2019. doi:10.1137/1.9781611975482.1.

11 Ho Yee Cheung, Tsz Chiu Kwok, and Lap Chi Lau. Fast matrix rank algorithms and applications. *J. ACM*, 60(5):31:1–31:25, 2013. doi:10.1145/2528404.

12 Michael B. Cohen, Jonathan A. Kelner, Rasmus Kyng, John Peebles, Richard Peng, Anup B. Rao, and Aaron Sidford. Solving directed laplacian systems in nearly-linear time through sparse LU factorizations. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 898–909, 2018.

13 Michael B. Cohen, Jonathan A. Kelner, John Peebles, Richard Peng, Anup B. Rao, Aaron Sidford, and Adrian Vladu. Almost-linear-time algorithms for markov chains and new spectral primitives for directed graphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 410–419, 2017.

14 Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of symbolic computation*, 9(3):251–280, 1990.

15 Wayne Eberly, Mark Giesbrecht, Pascal Giorgi, Arne Storjohann, and Gilles Villard. Faster inversion and other black box matrix computations using efficient block projections. In *Symbolic and Algebraic Computation, International Symposium, ISSAC 2007, Waterloo, Ontario, Canada, July 28 - August 1, 2007, Proceedings*, pages 143–150, 2007.

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

17 François Le Gall. Powers of tensors and fast matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014*, pages 296–303, 2014.

18 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001. doi:10.1145/502090.502098.

19 Magnus Rudolph Hestenes and Eduard Stiefel. *Methods of conjugate gradients for solving linear systems*, volume 49. DC: NBS, 1952.

20 Oscar H. Ibarra, Shlomo Moran, and Roger Hui. A generalization of the fast LUP matrix decomposition algorithm and applications. *J. Algorithms*, 3(1):45–56, 1982. doi:10.1016/0196-6774(82)90007-4.

21 Erich Kaltofen and B. David Saunders. On wiedemann's method of solving sparse linear systems. In Harold F. Mattson, Teo Mora, and T. R. N. Rao, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 9th International Symposium, AAECC-9, New Orleans, LA, USA, October 7-11, 1991, Proceedings*, volume 539 of *Lecture Notes in Computer Science*, pages 29–38. Springer, 1991. doi:10.1007/3-540-54522-0_93.

22 Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1283–1296. ACM, 2018. doi:10.1145/3188745.3188896.

23 Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A. Spielman. Sparsified cholesky and multigrid solvers for connection laplacians. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 842–850, 2016.

24 Rasmus Kyng, Di Wang, and Peng Zhang. Packing lps are hard to solve accurately, assuming linear equations are hard. 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 279–296. SIAM, 2020. doi:10.1137/1.9781611975994.17.

25 Rasmus Kyng and Peng Zhang. Hardness results for structured linear systems. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 684–695, 2017.

26 László Lovász. On determinants, matchings, and random algorithms. In *Fundamentals of Computation Theory, FCT 1979, Proceedings of the Conference on Algebraic, Arthmetic, and Categorial Methods in Computation Theory, Berlin/Wendisch-Rietz, Germany, September 17-21, 1979*, pages 565–574, 1979.

27 Marcin Mucha and Piotr Sankowski. Maximum matchings in planar graphs via gaussian elimination. *Algorithmica*, 45(1):3–20, 2006. doi:10.1007/s00453-005-1187-5.

28 Cameron Musco, Praneeth Netrapalli, Aaron Sidford, Shashanka Ubaru, and David P. Woodruff. Spectrum approximation beyond fast matrix multiplication: Algorithms and hardness. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, volume 94 of *LIPIcs*, pages 8:1–8:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ITCS.2018.8.

29 Colton Pauderis and Arne Storjohann. Deterministic unimodularity certification. In Joris van der Hoeven and Mark van Hoeij, editors, *International Symposium on Symbolic and Algebraic Computation, ISSAC'12, Grenoble, France - July 22 - 25, 2012*, pages 281–288. ACM, 2012. doi:10.1145/2442829.2442870.

30 Richard Peng and Santosh S. Vempala. Solving sparse linear systems faster than matrix multiplication. In Dániel Marx, editor, *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, pages 504–521. SIAM, 2021. doi:10.1137/1.9781611976465.31.

31 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 515–524. ACM, 2013. doi:10.1145/2488608.2488673.

32 Daniel A. Spielman and Shang-Hua Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM J. Matrix Analysis Applications*, 35(3):835–885, 2014. doi:10.1137/090771430.

33 Arne Storjohann. The shifted number system for fast linear algebra on integer matrices. *J. Complex.*, 21(4):609–650, 2005. doi:10.1016/j.jco.2005.04.002.

34 Andrew James Stothers. On the complexity of matrix multiplication, 2010.

35 Volker Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.

36 Douglas H. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Trans. Inf. Theory*, 32(1):54–62, 1986. doi:10.1109/TIT.1986.1057137.

37 Virginia Vassilevska Williams. Multiplying matrices in o (n2. 373) time, 2014.

38 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the ICM 2018*, pages 3447–3487. World Scientific, 2019.

39 Raphael Yuster and Uri Zwick. Fast sparse matrix multiplication. *ACM Trans. Algorithms*, 1(1):2–13, 2005. doi:10.1145/1077464.1077466.