Solving Sparse Linear Systems Faster than Matrix Multiplication

Richard Peng* Santosh Vempala†

Downloaded 06/27/21 to 143.215.38.55. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Abstract

Can linear systems be solved faster than matrix multiplication? While there has been remarkable progress for the special cases of graph structured linear systems, in the general setting, the bit complexity of solving an $n \times n$ linear system $Ax = b$ is $\tilde{O}(n^\omega)$, where $\omega < 2.372864$ is the matrix multiplication exponent. Improving on this has been an open problem even for sparse linear systems with $\text{poly}(n)$ condition number.

In this paper, we present an algorithm that solves linear systems in sparse matrices asymptotically faster than matrix multiplication for any $\omega > 2$. This speedup holds for any input matrix A with $o(n^{\omega-1}/\log(\kappa(A)))$ non-zeros, where $\kappa(A)$ is the condition number of A. For $\text{poly}(n)$-conditioned matrices with $\tilde{O}(n)$ nonzeros, and the current value of ω, the bit complexity of our algorithm to solve to within any $1/\text{poly}(n)$ error is $O(n^{2.331645})$.

Our algorithm can be viewed as an efficient, randomized implementation of the block Krylov method via recursive low displacement rank factorizations. It is inspired by the algorithm of [Eberly et al. ISSAC '06 '07] for inverting matrices over finite fields. In our analysis of numerical stability, we develop matrix anti-concentration techniques to bound the smallest eigenvalue and the smallest gap in eigenvalues of semi-random matrices.

1 Introduction

Solving a linear system $Ax = b$ is a basic algorithmic problem with direct applications to scientific computing, engineering, and physics, and is at the core of algorithms for many other problems, including optimization [Ye11], data science [BHK20], and computational geometry [EH10]. It has enjoyed an array of elegant approaches, from Cramer's rule and Gaussian elimination to numerically stable iterative methods to more modern randomized variants based on random sampling [ST11, KMP12] and sketching [DMM08, Woo14]. Despite much recent progress on faster solvers for graph-structured linear systems [Vai89, Gre96, ST14, KMP12, Kyn17], progress on the general case has been elusive.

Most of the work in obtaining better running time bounds for linear systems solvers has focused on efficiently computing the inverse of A, or some factorization of it. Such operations are in turn closely related to the cost of matrix multiplication. Matrix inversion can be reduced to matrix multiplication via divide-and-conquer, and this reduction was shown to be stable when the word size for representing numbers[1] is increased by a factor of $O(\log n)$ [DDHK07]. The current best runtime of $O(n^\omega)$ with $\omega < 2.372864$ [LG14] follows a long line of work on faster matrix multiplication algorithms [Str69, Pan84, CW87, Wil12, LG14] and is also the current best running time for solving $Ax = b$: When the input matrix/vector are integers, matrix multiplication based algorithms can obtain the exact rational value solution using $O(n^\omega)$ word operations [Dix82, Sto05].

Methods for Matrix inversion or factorization are often referred to as direct methods in the linear systems literature [DRSL16]. This is in contrast to iterative methods, which gradually converge to the solution. Iterative methods have little space overhead, and therefore are widely used for solving large, sparse, linear systems that arise in scientific computing. Another reason for their popularity is that they are naturally suited to producing approximate solutions of desired accuracy in floating point arithmetic, the de facto method for representing real numbers. Perhaps the most famous iterative method is the Conjugate Gradient (CG) / Lanczos algorithm [HS52, Lan50]. It was introduced as an $O(n \cdot nnz)$ time algorithm under exact arithmetic, where nnz is the number of non-zeros in the matrix. However, this bound only holds under the Real RAM model where the words have with unbounded precision [PS85, BSS+89]. When taking bit sizes into account, it incurs an additional factor of n. Despite much progress in iterative techniques in the intervening decades, obtaining analogous gains over matrix multiplication in the presence of round-off errors has remained an open question.

*Georgia Tech rpeng@cc.gatech.edu

†Georgia Tech vempala@gatech.edu

[1] We will be measuring bit-complexity under fixed-point arithmetic. Here the machine word size is on the order of the maximum number of digits of precision in A, and the total cost is measured by the number of word operations. The need to account for bit-complexity of the numbers naturally led to the notion of condition number [Tur48, Blu04]. The logarithm of the condition number measures the additional number of words needed to store A^{-1} (and thus $A^{-1}b$) compared to A. In particular, matrices with $poly(n)$ condition number can be stored with a constant factor overhead in precision, and are numerically stable under standard floating point number representations.

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 06/27/21 to 143.215.38.55. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

The convergence and stability of iterative methods typically depends on some *condition number* of the input. When all intermediate steps are carried out to precision close to the condition number of A, the running time bounds of the conjugate gradient algorithm, as well as other currently known iterative methods, depend polynomially on the condition number of the input matrix A. Formally, the condition number of a symmetric matrix A, $\kappa(A)$, is the ratio between the maximum and minimum eigenvalues of A. Here the best known rate of convergence when all intermediate operations are restricted to bit-complexity $O(\log(\kappa(A))$ is to an error of ϵ in $O(\sqrt{\kappa(A)}\log(1/\epsilon))$ iterations. This is known to be tight if one restricts to matrix-vector multiplications in the intermediate steps [SV13, MMS18]. This means for moderately conditioned (e.g. with $\kappa = poly(n)$), sparse, systems , the best runtime bounds are still via direct methods, which are stable when $O(\log(1/\kappa))$ words of precision are maintained in intermediate steps [DDHK07].

Many of the algorithms used in scientific computing for solving linear systems involving large, space, matrices are based on combining direct and iterative methods: we will briefly discuss this perspectives in Section 1.3. From the asymptotic complexity perspective, the practical successes of many such methods naturally leads to the question of whether one can provably do better than the $O(\min\{n^\omega, nnz \cdot \sqrt{\kappa(A)}\})$ time corresponding to the faster of direct or iterative methods. Somewhat surprisingly, despite the central role of this question in scientific computing and numerical analysis, as well as extensive studies of linear systems solvers, progress on this question has been elusive. The continued lack of progress on this question has led to its use as a hardness assumption for showing conditional lower bounds for numerical primitives such as linear elasticity problems [KZ17] and positive linear programs [KWZ20]. One formalization of such hardness is the *Sparse Linear Equation Time Hypothesis* (SLTH) from [KWZ20]: SLTH_k^γ denotes the assumption that a sparse linear system with $\kappa \leqslant nnz(A)^k$ cannot be solved in time faster than $nnz(A)^\gamma$ to within relative error $\epsilon = n^{-10k}$. Here improving over the smaller running time of both direct and iterative methods can be succinctly encapsulated as refuting $\text{SLTH}_k^{\min\{1+k/2,\omega\}}$. [2]

In this paper, we provide a faster algorithm for solving sparse linear systems. Our formal result is the following (we use the form defined in [KWZ20] [Linear Equation Approximation Problem, LEA]).

[2]The hardness results in [KWZ20] were based on $\text{SLTH}_{1.5}^{1.99}$ under the Real RAM model in part due to the uncertain status of conjugate gradient in different models of computation.

THEOREM 1.1. *Given a matrix A with max dimension n, $nnz(A)$ non-zeros (whose values fit into a single word), along with a parameter $\kappa(A)$ such that $\kappa(A) \geqslant \sigma_{\max}(A)/\sigma_{\min}(A)$, along with a vector b and error requirement ϵ, we can compute, under fixed point arithmetic, in time*

$$O\left(\max\left\{nnz(A)^{\frac{\omega-2}{\omega-1}}n^2, n^{\frac{5\omega-4}{\omega+1}}\right\}\log^2(\kappa/\epsilon)\right)$$

a vector x such that

$$\|Ax - \Pi_A b\|_2^2 \leqslant \epsilon \|\Pi_A b\|_2^2,$$

where c is a fixed constant and Π_A is the projection operator onto the column space of A.

Note that $\|\Pi_A b\|_2 = \|A^T b\|_{(A^TA)^{-1}}$, and when A is square and full rank, it is just $\|b\|_2$.

The cross-over point for the two bounds is at $nnz(A) = n^{\frac{3(\omega-1)}{\omega+1}}$. In particular, for the sparse case with $nnz(A) = O(n)$, and the current best $\omega \leqslant 2.372864$ [LG14], we get an exponent of

$$\max\left\{2 + \frac{\omega-2}{\omega-1}, \frac{5\omega-4}{\omega+1}\right\} < \max\{2.271595, 2.331645\} = 2.331645.$$

As $n \leqslant nnz$, this also translates to a running time of $O(nnz^{\frac{5\omega-4}{\omega+1}})$, which as $\frac{5\omega-4}{\omega+1} = \omega - \frac{(\omega-2)^2}{\omega+1}$, refutes SLTH_k^ω for constant values of k and any value of $\omega > 2$.

We can parameterize the asymptotic gains over matrix multiplication for moderately sparse instances. Here we use the $\widetilde{O}(\cdot)$ notation to hide lower-order terms, specifically $\widetilde{O}(f(n))$ denotes $O(f(n) \cdot \log^c(f(n)))$ for some absolute constant c.

COROLLARY 1.1. *For any matrix A with dimension at most n, $O(n^{\omega-1-\theta})$ non-zeros, and condition number $n^{O(1)}$, a linear system in A can be solved to accuracy $n^{-O(1)}$ in time $\widetilde{O}(\max\{n^{\frac{5\omega-4}{\omega+1}}, n^{\omega-\frac{\theta(\omega-2)}{\omega-1}}\})$.*

Here the cross-over point happens at $\theta = \frac{(\omega-1)(\omega-2)}{\omega+1}$. Also, because $\frac{5\omega-4}{\omega+1} = \omega - \frac{(\omega-2)^2}{\omega+1}$, we can also infer that for any $0 < \theta \leqslant \omega - 2$ and any $\omega > 2$, the runtime is $o(n^\omega)$, or asymptotically faster than matrix multiplication.

1.1 Idea At a high level, our algorithm follows the block Krylov space method (see e.g. Chapter 6.12 of Saad [Saa03]). This method is a multi-vector extension of the conjugate gradient / Lanczos method, which in the single-vector setting is known to be problematic under round-off errors both in theory [MMS18] and in

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 06/27/21 to 143.215.38.55. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

practice [GO89]. Our algorithm starts with a set of s initial vectors, $B \in \Re^{n \times s}$, and forms a column space by multiplying these vectors by A repeatedly, m times. Formally, the block Krylov space matrix is

$$K = \left[\; B \;\middle|\; AB \;\middle|\; A^2B \;\middle|\; \ldots \;\middle|\; A^{m-1}B \; \right].$$

The core idea of Krylov space methods is to efficiently orthogonalize this column space. For this space to be spanning, block Krylov space methods typically choose s and m so that $sm = n$.

The conjugate gradient algorithm can be viewed as an efficient implementation of the case $s = 1$, $m = n$, and B is set to b, the RHS of the input linear system. The block case with larger values of s was studied by Eberly, Giesbrecht, Giorgi, Storjohann, and Villard [EGG+06, EGG+07] over finite fields, and they gave an $O(n^{2.28})$ time algorithm for computing the inverse of a sparse matrix over a finite field.

Our algorithm also leverages the top-level insight of the Eberly et al. results: the Gram matrix of the Krylov space matrix (which can be used inter-changeably for solving linear systems) is a block Hankel matrix. That is, if we view the Gram matrix $(AK)^T(AK)$ as an m-by-m matrix containing s-by-s sized blocks, then all the blocks along each anti-diagonal are the same:

$$(AK)^T (AK) = \left[\begin{array}{c|c|c|c} B^TA^2B & B^TA^3B & \ldots & B^TA^{m+1}B \\ \hline B^TA^3B & B^TA^4B & \ldots & B^TA^{m+2}B \\ \hline \ldots & \ldots & \ldots & \ldots \\ \hline B^TA^{m+1}B & B^TA^{m+2}B & \ldots & B^TA^{2m}B \end{array}\right]$$

Formally, the s-by-s inner product matrix formed from A^iB and A^jB is $B^TA^{i+j}B$, and only depends on $i + j$. So instead of m^2 blocks each of size $s \times s$, we are able to represent a n-by-n matrix with about m blocks.

Operations involving these m blocks of the Hankel matrix can be handled using $\widetilde{O}(m)$ block operations. This is perhaps easiest seen for computing matrix-vector products using K. If we use $\{i\}$ to denote the ith block of the Hankel matrix, that is

$$H_{\{i,j\}} = M\,(i + j)$$

for a sequence of matrices M, we get that the i^{th} block of the product Hx can be written in block-form as

$$(Hx)_{\{i\}} = \sum_j H_{\{i,j\}} x_{\{j\}} = \sum_j M\,(i + j)\, x_{\{j\}}.$$

Note this is precisely the convolution of (a sub-interval) of M and x, with shifts indicated by i. Therefore, in the forward matrix-vector multiplication direction, a speedup by a factor of about m is possible with fast convolution algorithms. The performance gains of the Eberly et al. algorithms [EGG+06, EGG+07] can be viewed as of similar nature, albeit in the more difficult direction of solving linear systems. Specifically, they utilize algorithms for the Padé problem of computing a polynomial from the result of its convolution [XB90, BL94]. Over finite fields, or under exact arithmetic, such algorithms for matrix Padé problems take $O(m \log m)$ block operations [BL94], for a total of $\widetilde{O}(s^\omega m)$ operations..

The overall time complexity follows from two opposing goals:

1. Quickly generate the Krylov space: repeated multiplication by A allows us to generate A^iB using $O(ms \cdot nnz) = O(n \cdot nnz)$ arithmetic operations. Choosing a sparse B then allows us to compute B^TA^iB in $O(n \cdot s)$ arithmetic operations, for a total overhead of $O(n^2) = O(n \cdot nnz)$.
2. Quickly invert the Hankel matrix. Each operation on an s-by-s block takes $O(s^\omega)$ time. Under the optimistic assumption of $\widetilde{O}(m)$ block operations, the total is $\widetilde{O}(m \cdot s^\omega)$.

Under these assumptions, and the requirement of $n \approx ms$, the total cost becomes about $O(n \cdot nnz + m \cdot s^\omega)$, which is at most $O(n \cdot nnz)$ as long as $m > n^{\frac{\omega-2}{\omega-1}}$. However, this runtime complexity is over finite fields, where numerical stability is not an issue, instead of over reals under round-off errors, where one must contend with numerical errors without blowing up the bit complexity. This is a formidable challenge; indeed, with exact arithmetic, the CG method takes time $O(n \cdot nnz)$, but this is misleading since the computation is effective only the word sizes increase by a factor of n (to about $n \log \kappa$ words), which leads to an overall complexity of $O(n^2 \cdot nnz \cdot \log \kappa)$.

1.2 Our Contributions

Our algorithm can be viewed as the numerical generalization of the algorithms from [EGG+06, EGG+07]. We work with real numbers of bounded precision, instead of entries over a finite field. The core of our approach can be summarized as:

> The block Krylov space method together with fast Hankel solvers can be made numerically stable using $\widetilde{O}(m \log(\kappa))$ words of precision.

Doing so, on the other hand, requires developing tools for two topics that have been extensively studied in mathematics, but separately.

1. Obtain low numerical cost solvers for block Hankel/Toeplitz matrices. Many of the prior algorithms rely on algebraic identities that do not

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

generalize to the block setting, and are often (experimentally) numerically unstable [GTVDV96, Gra06].

2. Develop matrix anti-concentration bounds for analyzing the word lengths of inverses of random Krylov spaces. Such bounds upper bound the probability of random matrices being in some set of small measure, which in our case is the set of nearly singular matrices. Previously, they were known assuming the matrix entries are independent [SST06, TV10], while Krylov spaces have correlated columns.

Furthermore, due to the shortcomings of the matrix anti-concentration bounds, we modify the solver algorithm so that it uses a more limited version of the block-Krylov space that fall under the cases that could be analyzed.

Before we describe the difficulties and new tools needed, we first provide some intuition on why a factor m increase in word lengths may be the right answer by upper-bounding the magnitudes of entries in a m-step Krylov space. The maximum magnitude of $A^m b$ is bounded by the max magnitude of A to the power of m, times a factor corresponding to the number of summands in the matrix product:

$$\|A^m b\|_\infty \leqslant \left(n \|\|A\|\|_\infty\right)^m \|b\|_\infty .$$

So the largest numbers in K (as well as AK) can be bounded by $(n\kappa)^{O(m)}$, or $O(m \log \kappa)$ words in front of the decimal point under the assumption of $\kappa > n$.

Should such a bound of $O(m \log \kappa)$ hold for all numbers that arise, including the matrix inversions, and the matrix B is sparse with $O(n)$ entries, the cost of computing the block-Krylov matrices becomes $O(m \log \kappa \cdot ms \cdot nnz)$, while the cost of the matrix inversion portion encounters an overhead of $O(m \log \kappa)$, for a total of $\widetilde{O}(m^2 s^\omega \log \kappa)$. In the sparse case of $nnz = O(n)$, and $n \approx ms$, this becomes:

$$\begin{aligned} &O\left(n^2 m \log \kappa + m^2 s^\omega \log \kappa\right) \\ &= O\left(n^2 m \log \kappa + \frac{n^\omega}{m^{\omega-2}} \log \kappa\right). \end{aligned} \tag{1.1}$$

Due to the gap between n^2 and n^ω, setting m appropriately gives improvement over n^ω when $\log \kappa < n^{o(1)}$.

However, the magnitude of an entry in the inverse depends on the smallest magnitude, or in the matrix case, its minimum singular value. Bounding and propagating the min singular value, which intuitively corresponds to how close a matrix is to being degenerate, represents our main challenge. In exact/finite fields settings, non-degeneracies are certified via the Schwartz-Zippel Lemma about polynomial roots. The numerical analog of this is more difficult: the Krylov space matrix K is asymmetric, even for a symmetric matrix A. It is much easier for an asymmetric matrix with correlated entries to be close to singular.

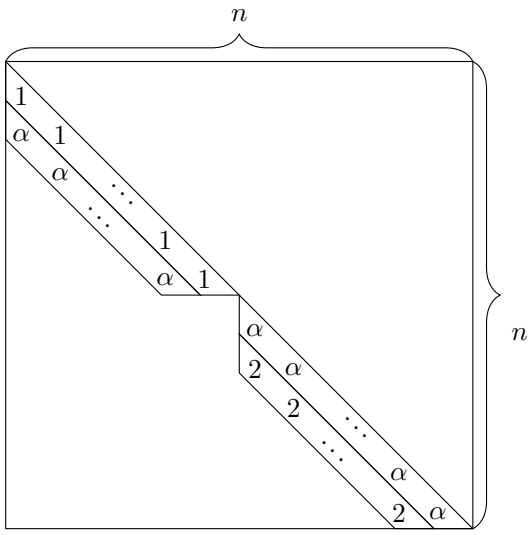

Figure 1: The difference between matrix anti-concentration over finite fields and reals: a matrix that is full rank for all $\alpha \neq 0$, but is always ill conditioned.

Consider for example a two-banded, two-block matrix with all diagonal entries set to the same random variable α (see Figure 1):

$$A_{ij} = \begin{cases} 1 & \text{if } i = j \text{ and } j \leqslant n/2, \\ \alpha & \text{if } i = j+1 \text{ and } j \leqslant n/2, \\ \alpha & \text{if } i = j+1 \text{ and } n/2 < j, \\ 2 & \text{if } i = j+1 \text{ and } n/2 < j, \\ 0 & \text{otherwise.} \end{cases}$$

In the exact case, this matrix is full rank unless $\alpha = 0$, even over finite fields. On the other hand, its minimum singular value is close to 0 for all values of α because:

Observation 1.1. *The minimum singular value of a matrix with* 1*s on the diagonal,* α *on the entries immediately below the diagonal, and* 0 *everywhere else is at most* $|\alpha|^{-(n-1)}$*, due to the test vector* $[1; -\alpha; \alpha^2; \ldots; (-\alpha)^{n-1}]$.

Specifically, in the top-left block, as long as $|\alpha| > 3/2$, the top left block has minimum singular value at most $(2/3)^{n-1}$. On the other hand, rescaling the bottom-right block by $1/\alpha$ to get 1s on the diagonal gives $2/\alpha$ on the off-diagonal. So as long as $|\alpha| < 3/2$, this value is at least $4/3$, which in turn implies a minimum singular

Downloaded 06/27/21 to 143.215.38.55. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

value of at most $(3/4)^{n-1}$ in the bottom right block. This means no matter what value α is set to, this matrix will always have a singular value that's exponentially close to 0. Furthermore, the Gram matrix of this matrix also gives such a counter example to symmetric matrices with (non-linearly) correlated entries. Previous works on analyzing condition numbers of asymmetric matrices also encounter similar difficulties: a more detailed discussion of it can be found in Section 7 of Sankar et al. [SST06].

In order to bound the bit complexity of all intermediate steps of the block Krylov algorithm by $\widetilde{O}(m)\cdot\log\kappa$, we devise a more numerically stable algorithm for solving block Hankel matrices, as well as provide a new perturbation scheme to quickly generate a well-conditioned block Krylov space. Central to both of our key components is the close connection between condition number and bit complexity bounds.

First, we give a more numerically stable solver for block Hankel/Toeplitz matrices. Fast solvers for Hankel (and closely related Toeplitz) matrices have been extensively studied in numerical analysis, with several recent developments on more stable algorithms [XXG12]. However, the notion of numerical stability studied in these algorithms is the more practical variant where the number of bits of precision is fixed. As a result, the asymptotic behavior of the stable algorithm from [XXG12] is quadratic in the number of digits in the condition number, which in our case would translate to a prohibitive cost of $\widetilde{O}(m^2)$ (i.e., the overall cost would be higher than n^ω).

Instead, we combine developments in recursive block Gausssian elimination [DDHK07, KLP+16, CKK+18] with the *low displacement rank* representation of Hankel/Toeplitz matrices [KKM79, BA80]. Such representations allow us to implicitly express both the Hankel matrix and its inverse by displaced versions of rank $2s$ matrices. This means the intermediate sizes of instances arising from recursion is $O(s)$ times the dimension, for a total size of $O(n\log n)$, giving a total of $\widetilde{O}(ns^{\omega-1})$ arithmetic operations involving words of size $\widetilde{O}(m)$. We provide a rigorous analysis of the accumulation of round-off errors similar to the analysis of recursive matrix multiplication based matrix inversion from [DDHK07].

Motivated by this close connection with the condition number of Hankel matrices, we then try to initialize with Krylov spaces of low condition number. Here we show that a sufficiently small perturbation suffices for producing a well conditioned overall matrix. In fact, the first step of our proof, that a small sparse random perturbation to A guarantees good separations between its eigenvalues is a direct combination of bounds on eigenvalue separation of random Gaussians [NTV17] as well as min eigenvalue of random sparse matrices [LV18]. This separation then ensures that the powers of A, $A^1, A^2, \ldots A^m$, are sufficiently distinguishable from each other. Such considerations also come up in the smoothed analysis of numerical algorithms [SST06].

The randomness of the Krylov matrix induced by the initial set of random vectors B is more difficult to analyze: each column of B affects m columns of the overall Krylov space matrix. In contrast, all existing analyses of lower bounds of singular values of possibly asymmetric random matrices [SST06, TV10] rely on the randomness in the columns of matrices being independent. The dependence between columns necessitates analyzing singular values of random linear combinations of matrices, which we handle by adapting ϵ-net based proofs of anti-concentration bounds. Here we encounter an additional challenge in bounding the minimum singular value of the block Krylov matrix. We resolve this issue algorithmically: instead of picking a Krylov space that spans the entire $\Re^n$, we stop things short by picking $ms = n - \widetilde{O}(m)$ This set of extra columns significantly simplify the proof of singular value lower bounds. This is similar in spirit to the analysis of minimum singular values of random matrices, which is significantly easier for non-square matrices [RV10]. In the algorithm, the remaining columns are treated as a separate block that we reduce to via a Schur complement at the very end of the block elimination algorithm. Since the block is small, so is its overhead on the running time.

1.3 History and Related Work Our algorithm has close connections with multiple lines of research on more efficient solvers for sparse linear systems. This topic has been extensively studied not only in computer science, but also in applied mathematics and engineering. For example, in the Editors of the Society of Industrial and Applied Mathematics News' 'top 10 algorithms of the 20th century', three of them (Krylov space methods, matrix decompositions, and QR factorizations) are directly related to linear systems solvers [Cip00].

At a high level, our algorithm is a hybrid linear systems solver. It combines iterative methods, namely block Krylov space methods, with direct methods that factorize the resulting Gram matrix of the Krylov space. Hybrid methods have their origins in the incomplete Cholesky method for speeding up elimination/factorization based direct solvers. A main goal of these methods is to reduce the $\Omega(n^2)$ space needed to represent matrix factorizations / inverses, which is even more problematic than time when handling large sparse

Downloaded 06/27/21 to 143.215.38.55. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 06/27/21 to 143.215.38.55. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

matrices. Such reductions can occur in two ways: either by directly dropping entries from the (intermediate) matrices, or by providing more succinct representations of these matrices using additional structures.

The main structure of our algorithm is based on the latter line of work on solvers for structured matrices. Such systems arise from physical processes where the interactions between objects have invariances (e.g. either by time or space differences). Examples of such structure include circulant matrices [Gra06], Toeplitz / Hankel matrices [KKM79, BA80, XXG12, XXCB14], and distances from n-body simulations [CRW93]. Many such algorithms require exact preservation of the structure in intermediate steps. As a result, many of these works develop algorithms over finite fields [BA80, BL94, BJMS17].

More recently, there has been work on developing more numerically stable variants of these algorithms for structured matrices, or more generally, matrices that are numerically close to being structured [XCGL10, LLY11, XXG12, XXCB14]. However, these results only explicitly discussed the entry-wise Hankel/Toeplitz case (which corresponds to $s = 1$). Furthermore, because they rely on domain-decomposition techniques similar to fast multiple methods, they produce one bit of precision per each outer iteration loop. As the Krylov space matrix has condition number $\exp(\Omega(m))$, such methods would lead to another factor of m in the solve cost when directly invoked.

Instead, our techniques for handling and bounding numerical errors are more closely related to recent developments in provably efficient sparse Cholesky factorizations [KLP$^+$16, KS16, Kyn17, CKK$^+$18]. These methods generated efficient preconditioners using only the condition of intermediate steps of Gaussian eliminatnion, known as Schur complements, having small representations. They avoided the explicit generation of the dense representations of Schur complements by treatment them as operators, and implicitly applied randomized tools to directly sample/sketch the final succinct representations, which have much smaller algorithmic costs.

On the other hand, previous works on spare Choleskfy factorizations required the input matrix to be decomposable into a sum of simple elements, often through additional combinatorial structure of the matrices. In particular, this line of work on combinatorial preconditioning was initiated through a focus on graph Laplacians, which are built from 2-by-2 matrix blocks corresponding to edges of undirected graphs [Vai89, Gre96, ST14, KMP12]. Since then, there has been substantial generalizations to the structures amenable to such approaches, notably to finite element matrices [BHV08] and directed graphs / irreversible Markov chains [CKP$^+$17]. However, recent works have also shown that many classes of structures involving more than two variables are complete for general linear systems [Zha18]. Nonetheless, the prevalence of approximation errors in such algorithms led to the development of new ways of bounding numerical/round-off errors in algorithms that are critical to our elimination routine for block-Hankel matrices.

Key to recent developments in combinatorial preconditioning is matrix concentration [RV07, Tro15]. Such bounds provide guarantees for (relative) eigenvalues of random sums of matrices. For generating preconditioners, such randomness arise from whether each element is kept, and a small condition number (which in turn implies a small number of outer iterations usign the preconditioners) corresponds to a small deviation between the original and sampled matrices. In contrast, we introduce randomness in order to obtain block Krylov spaces whose minimum eigen-value is large. As a result, the matrix tool we need is anti-concentration, which somewhat surprisingly is far less studied. Previous works on it are mostly related by similar problems from numerical precision [SST06, TV10], and mostly address situations where the entries in the resulting matrix are independent. Our bound on the min singular value of the random Krylov space can yield a crude bound for a sum of rectangluar random matrices, but we believe much better matrix anti-concentration bounds are possible.

Due to space constraints, we will only describe the overall algorithm in Section 2, and give an outline of its analysis in Section 3. Section 3.2 contains a breakdown of the main components of the analysis, whose details and proofs can be found in the arXiv version [PV20]. Some research directions raised by this work, including possible improvements and extensions are discussed in Section 4.

2 Algorithm

We describe the algorithm, as well as the running times of its main components in this section. To simplify discussion, we assume the input matrix A is symmetric, and has $poly(n)$ condition number. If it is asymmetric (but invertible), we implicitly apply the algorithm to A^TA, using the identity $A^{-1} = (A^TA)^{-1}A^T$ derived from $(A^TA)^{-1} = A^{-1}A^{-T}$. Also, recall from the discussion after Theorem 1.1 that we use $\widetilde{O}(\cdot)$ to hide lower order terms in order to simplify runtimes.

Before giving details on our algorithm, we first discuss what constitutes a linear systems solver algorithm, specifically the equivalence between many such algorithms and linear operators.

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 06/27/21 to 143.215.38.55. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

For an algorithm ALG that takes a matrix B as input, we say that ALG is linear if there is a matrix Z_{ALG} such that for any input B, we have

$$\text{ALG}(B) = Z_{\text{ALG}}.$$

In this section, in particular in the pseudocode in Algorithm 2, we use the name of the procedure, $\text{SOLVE}_A(b, \delta)$, interchangeably with the operator corresponding to a linear algorithm that solves a system in A, on vector b, to error $\delta > 0$. In the more formal analysis, we will denote such corresponding linear operators using the symbol Z, with subscripts corresponding to the routine if appropriate.

This operator/matrix based analysis of algorithms was first introduced in the analysis of recursive Chebyshev iteration by Spielman and Teng [ST14], with credits to the technique also attributed to Rohklin. It the advantage of simplifying analyses of multiple iterations of such algorithms, as we can directly measure Frobenius norm differences between such operators and the exact ones that they approximate.

Under this correspondence, the goal of producing an algorithm that solves $Ax = b$ for any b as input becomes equivalent to producing a linear operator Z_A that approximates A^{-1}, and then running it on the input b. For convenience, we also let the solver take as input a matrix instead of a vector, in which case the output is the result of solves against each of the columns of the input matrix.

The high-level description of our algorithm is in Figure 2. To keep our algorithms as linear operators, we will ensure that the only approximate steps are from inverting matrices (where condition numbers naturally lead to matrix approximation errors), and in forming operators using fast convolution. We will specify explicitly in our algorithms when such round-off errors occur.

Some of the steps of the algorithm require care for, efficiency, as well as tracking the number of words needed to represent the numbers. We assume the bounds on bit-complexity in the analysis (Section 3) below, which is $\widetilde{O}(m)$ when $\kappa = poly(n)$, and use this in the brief description of costs in the outline of the steps below.

We start by perturbing the input matrix, resulting in a symmetric positive definite matrix where all eigenvalues are separated by α_A. Then we explicitly form a Krylov matrix from sparse Random Gaussians: For any vector u, we can compute $A^i u$ from $A^{i-1}u$ via a single matrix-vector multiplication in A. So computing each column of K requires $O(nnz(A))$ operations, each involving a length n vector with words of length $\widetilde{O}(m)$.

BlockKrylov($\text{MatVec}_A(x, \delta)$: symmetric matrix given as implicit matrix vector muliplication access, α_A: eigenvalue range/separation bounds for A that also doubles as error threshold, m: Krylov step count,)

1. (FORM KRYLOV SPACE)
   (a) Set $s \leftarrow \lfloor n/m \rfloor - O(m)$, $h \leftarrow O(m^2 \log(1/\alpha_A))$. Let G^S be an $n \times s$ random matrix with G^S_{ij} set to $\mathcal{N}(0,1)$ with probability $\frac{h}{n}$, and 0 otherwise.
   (b) (Implicitly) compute the block Krylov space
   $$K = \left[\; G^S \mid AG^S \mid A^2G^S \mid \ldots \mid A^{m-1}G^S \; \right],$$
2. (SPARSE INVERSE) Use fast solvers for block Hankel matrices to obtain a solver for the matrix:
   $$M \leftarrow (AK)^T (AK),$$
   and in turn a solve to arbitrary error which we denote $\text{SOLVE}_M(\cdot, \epsilon)$.
3. (PAD and SOLVE)
   (a) Let $r = n - ms$ denote the number of remaining columns. Generate a $n \times r$ dense Gaussian matrix G, use it to complete the basis as: $Q = [K|G]$.
   (b) Compute the Schur complement of $(AQ)^T AQ$ onto its last $r = n - ms$ entries (the ones corresponding to the columns of G) via the operation
   $$(AG)^T AG - (AG)^T \cdot AK \cdot \text{SOLVE}_M\left((AK)^T AG, \alpha_A^{10m}\right)$$
   and invert this r-by-r matrix.
   (c) Use the inverse of this Schur complement, as well as $\text{SOLVE}_M(\cdot, \epsilon)$ to obtain a solver for Q^TQ, $\text{SOLVE}_{Q^TQ}(\cdot, \epsilon)$.
4. (SOLVE and UNRAVEL) Return the operator $Q \cdot \text{SOLVE}_{(AQ)^TAQ}((AQ)^T x, \alpha_A^{10m})$ as an approximate solver for A.

Figure 2: Pseudocode for block Krylov space algorithm: $\text{SOLVE}_\cdot(\cdot, \cdot)$ are operators corresponding to linear system solving algorithms whose formalization we discuss at the start of this section.

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 06/27/21 to 143.215.38.55. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

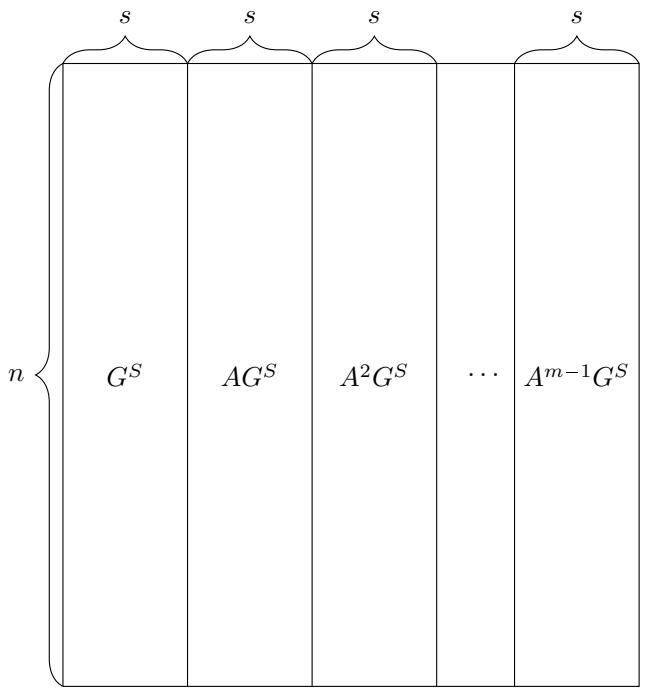

Figure 3: Randomized m-step Krylov Space Matrix with n-by-s sparse Gaussian G^S as starter

So we get the matrix K, as well as AK, in time

$$\widetilde{O}\left(nnz\left(A\right)\cdot n\cdot m\right).$$

To obtain a solver for AK, we instead solve its Gram matrix $(AK)^T(AK)$. Each block of K^TK has the form $(G^S)^TA^iG^S$ for some $2\leqslant i\leqslant 2m$, and can be computed by multiplying $(G^S)^T$ and A^iG^S. As A^iG^S is an n-by-s matrix, each non-zero in G^S leads to a cost of $O(s)$ operations involving words of length $\widetilde{O}(m)$. Then because we chose G^S to have $\widetilde{O}(m^3)$ non-zeros per column, the total number of non-zeros in G^S is about $\widetilde{O}(s\cdot m^3)=\widetilde{O}(nm^2)$. This leads to a total cost (across the m values of i) of:

$$\widetilde{O}\left(n^2m^3\right).$$

The key step is then Step 2: a block version of the Conjugate Gradient method. It will be implemented using a recursive data structure based on the notion of displacement rank [KKM79, BA80]. To get a sense of why a facter algorithm may be possible, note that there are only $O(m)$ distinct blocks in the matrix $(AK)^T(AK)$. So a natural hope is to invert these blocks by themselves: the cost of (stable) matrix inversion [DDH07], times the $\widetilde{O}(m)$ numerical word complexity, would then give a total of

$$\widetilde{O}\left(m^2s^\omega\right)=\widetilde{O}\left(m^2\left(\frac{n}{m}\right)^\omega\right)=\widetilde{O}\left(n^\omega m^{\omega-2}\right).$$

Of course, it does not suffice to solve these m s-by-s blocks independently. Instead, the full algorithm, as well as the $\textsc{Solve}_M$ operator, is built from efficiently convolving such s-by-s blocks with matrices using Fast Fourier Transforms. Such ideas can be traced back to the development of super-fast solvers for (entry-wise) Hankel/Toeplitz matrices [BA80, LS92, XXCB14].

Choosing s and m so that $n=sm$ would then give the overal running time, **assuming that we can bound the minimum singular value of K by** $\exp(-\widetilde{O}(m))$. This is a major shortcoming of our analysis: we can only prove such a bound when $n-sm\geqslant\Omega(m)$. Its underlying cause is that rectangular semi-random matrices can be analyzed using ϵ-nets, and thus are significantly easier to analyze than square matrices.

This means we can only use m and s such that $n-ms=\Theta(m)$, and we need to pad K with $n-ms$ columns to form a full rank, invertible, matrix. To this end we add $\Theta(m)$ dense Gaussian columns to K to form Q, and solve the system AQ, and its associated Gram matrix $(AQ)^T(AQ)$ instead. These matrices are shown in Figure 4.

Because these additional columns are entry-wise i.i.d, its minimum singular value can be analyzed using existing tools [SST06, TV10], namely lower bounding the dot product of a random vector against any normal vector. Thus, we can lower bound the minimum singular value of Q, and in turn AQ, by $\exp(-\widetilde{O}(m))$ as well.

This bound in turn translates to the minimum eigenvalue of the Gram matrix of AQ, $(AQ)^T(AQ)$. Partitioning its entries by those from K and G gives four blocks: one (sm)-by-(sm) block corresponding to $(AK)^T(AK)$, one $\Theta(m)$-by-$\Theta(m)$ block corresponding to $(AG)^T(AG)$, and then the cross terms. To solve this matrix, we apply block-Gaussian elimination, or equivalently, form the Schur complement onto the $\Theta(m)$-by-$\Theta(m)$ corresponding to the columns in AG.

To compute this Schur complement, it suffices to solve the top-left block (corresponding to $(AK)^T(AK)$) against every column in the cross term. As there are at most $\Theta(m)<s$ columns, this solve cost comes out to less than $\widetilde{O}(s^\omega m)$ as well. We are then left with a $\Theta(m)$-by-$\Theta(m)$ matrix, whose solve cost is a lower order term.

So the final solver operator costs

$$\widetilde{O}\left(nnz(A)\cdot nm+n^2m^3+n^\omega m^{2-\omega}\right)$$

which leads to the final running time by choosing m to balance the terms. This bound falls short of the ideal case given in Equation 1.1 mainly due to the need for a denser B to the well-conditionedness of the Krylov space matrix. Instead of $O(n)$ non-zeros total, or about $O(m)$ per column, we need $poly(m)$ non-zero variables per

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 06/27/21 to 143.215.38.55. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

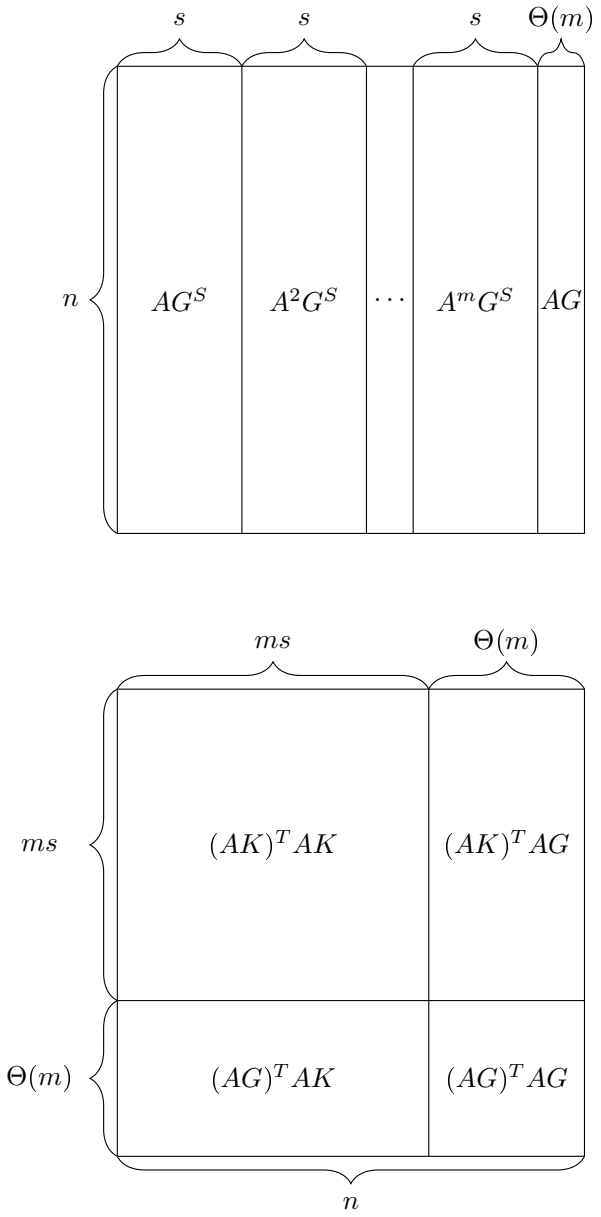

Figure 4: Full matrix AQ and its Associated Gram Matrix $(AQ)^T(AQ)$. Note that by our choice of parameters m is much smaller than $s \approx n/m$.

column to ensure the an $\exp(-O(m))$ condition number of the block Krylov space matrix K. This in turn leads to a total cost of $O(n \cdot nnz \cdot poly(m))$ for computing the blocks of the Hankel matrix, and a worse trade off when summed against the $\frac{n^\omega}{m^{\omega-2}}$ term.

3 Outline of Analysis

In this section we outline our analysis of the algorithm through formal theorem statements. We start by formalizing our tracking of convergence, and the tracking of errors and roundoff errors.

3.1 Preliminaries We will use capital letters for matrices, lower case letters for vectors and scalars. All subscripts are for indexing into entries of matrices and vectors, and superscripts are for indexing into entries of a sequence.

Norms and Singular Values. Our convergence bounds are all in terms of the Euclidean, or ℓ_2 norms. For a length n vector x, the norm of x is given by $\|x\| = \sqrt{\sum_{1 \leqslant i \leqslant n} x_i^2}$. Similarly, for a matrix M, the norm of its entries treated as a vector is known as the Frobenius norm, and we have

$$\|M\|_F = \sqrt{\sum_{ij} M_{ij}^2} = \sqrt{\textsc{Trace}\left(M^T M\right)}.$$

We will also use $\|\!|\cdot|\!\|$ to denote entry-wise norms over a matrix, specifically $\|M\|_\infty$ to denote the max magnitude of an entry in M. Note that $\|M\|_F = \|\!|M|\!\|_2$, so we have $\|\!|M|\!\|_\infty \leqslant \|M\|_F \leqslant n \|\!|M|\!\|_\infty$.

The minimum and maximum singular values of a matrix M are then defined as the min/max norms of its product against a unit vector:

$$\sigma_{\min}(M) = \min_x \frac{\|Mx\|_2}{\|x\|_2} \qquad \sigma_{\max}(M) = \max_x \frac{\|Mx\|_2}{\|x\|_2},$$

and the condition number of M is defined as $\kappa(M) = \sigma_{\max}(M)/\sigma_{\min}(M)$.

Bounds on the minimum singular value allows us to transfer perturbation errors to it to its inverse.

LEMMA 3.1. *If M is a full rank square matrix with min and max singular values in the range $[\sigma_{\min}, \sigma_{\max}]$, and $\widetilde{M}$ is some approximation of it such that $\left\|\widetilde{M} - M\right\|_F \leqslant \epsilon$ for some $\epsilon < \sigma_{\min}/2$, then*

1. *All singular values in $\widetilde{M}$ are in the range $[\sigma_{\min} - \epsilon, \sigma_{\max} + \epsilon]$, and*
2. *The inverse of $\widetilde{M}$ is close to the inverse of M:*

$$\left\|\widetilde{M}^{-1} - M^{-1}\right\|_F \leqslant 10\sigma_{\min}^{-2}\epsilon.$$

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 06/27/21 to 143.215.38.55. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Proof. The bound on singular values follows from the norm minimization/maximization definition of singular values. Specifically, we get that for a unit vector x,

$$\left|\left\|\widetilde{M}x\right\|_2 - \|Mx\|_2\right| \leqslant \left\|\left(\widetilde{M} - M\right)x\right\|_2 \leqslant \left\|\widetilde{M} - M\right\|_2 \|x\|_2 \leqslant \epsilon,$$

which means all singular values can change by at most ϵ.

Note that this implies that $\widetilde{M}$ is invertible. For the bounds on inverses, note that

$$\widetilde{M}^{-1} - M^{-1} = M^{-1}\left(M\widetilde{M}^{-1} - I\right) = M^{-1}\left(M - \widetilde{M}\right)\widetilde{M}^{-1}.$$

So applying bounds on norms, as well as $\left\|\widetilde{M}^{-1}\right\|_2 \leqslant (\sigma_{\min} - \epsilon)^{-1} \leqslant 2\sigma_{\min}^{-1}$ gives

$$\left\|\widetilde{M}^{-1} - M^{-1}\right\|_F \leqslant \left\|M^{-1}\right\|_2 \left\|M - \widetilde{M}\right\|_F \left\|\widetilde{M}^{-1}\right\|_2 \leqslant 2\sigma_{\min}^{-2}\epsilon.$$

∎

Error Accumulation. Our notion of approximate operators also compose well with errors.

LEMMA 3.2. *If $Z^{(1)}$ and $Z^{(2)}$ are linear operators with (algorithmic) approximations $\widetilde{Z}^{(1)}$ and $\widetilde{Z}^{(2)}$ such that for some $\epsilon < 0.1$, we have*

$$\left\|Z^{(1)} - \widetilde{Z}^{(1)}\right\|_F, \left\|Z^{(2)} - \widetilde{Z}^{(2)}\right\|_F \leqslant \epsilon$$

then their product satisfies

$$\left\|Z^{(1)}Z^{(2)} - \widetilde{Z}^{(1)}\widetilde{Z}^{(2)}\right\|_F \leqslant 10\epsilon \max\left\{1, \left\|Z^{(1)}\right\|_2, \left\|Z^{(2)}\right\|_2\right\}.$$

Proof. Expanding out the errors gives

$$Z^{(1)}Z^{(2)} - \widetilde{Z}^{(1)}\widetilde{Z}^{(2)} = \left(Z^{(1)} - \widetilde{Z}^{(1)}\right)Z^{(2)} + \left(Z^{(2)} - \widetilde{Z}^{(2)}\right)Z^{(1)} + \left(Z^{(1)} - \widetilde{Z}^{(1)}\right)\left(Z^{(2)} - \widetilde{Z}^{(2)}\right)$$

The terms involving the original matrix against the error gets bounded by the error times the norm of the original matrix. For the cross term, we have

$$\left\|\left(Z^{(1)} - \widetilde{Z}^{(1)}\right)\left(Z^{(2)} - \widetilde{Z}^{(2)}\right)\right\| \leqslant \left\|Z^{(1)} - \widetilde{Z}^{(1)}\right\|_F \cdot \left\|Z^{(2)} - \widetilde{Z}^{(2)}\right\|_2 \leqslant \epsilon^2,$$

which along with $\epsilon < 0.1$ gives the overall bound. ∎

Randomization and Normal Distributions Our algorithms rely on randomly perturbing the input matrices to make them non-degenerate, and much of our analysis revolving analyzing the effect of such perturbations on the eigenvalues. We make use of standard notions of probability, in particular, the union bound, which states that for any two events E_1 and E_2, $\Pr[E_1 \cup E_2] \leqslant \Pr[E_1] + \Pr[E_2]$.

Such a bound means that it suffices to show that the failure probability of any step of our algorithm is n^{-c} for some constant c. The total number of steps is $poly(n)$, so unioning over such probabilities still give a success probability of at least $1 - n^{-c+O(1)}$.

We will perturb our matrices using Gaussian random variables. These random variables $N(0, \sigma)$ have density function $g(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-x^2/2\sigma^2}$. They are particularly useful for showing anti-concentration because the sum of Gaussians is another Gaussian, with variance equalling to the sum of squares, or ℓ_2^2-norm, of the variance terms. That is, for a vector x and a (dense) Gaussian vector with entry-wise i.i.d. $N(0,1)$ normal random variables, aka. $g \sim N(0,1)^n$, we have $x^T g \sim N(0, \|x\|_2)$.

The density function of Gaussians means that their magnitude exceed n with probability at most $O(\exp(-n^2))$. This probability is much smaller than the n^{-c} failure probabilities that we want, so to simplify presentation we will remove it at the start.

CLAIM 3.1. *We can analyze our algorithm conditioning on any normal random variable with variance σ, $N(0,\sigma)$, having magnitude at most $n\sigma$.*

Tracking Word Length. The numerical rounding model that we will use is fixed point precision. The advantage of such a fixed point representation is that it significantly simplifies the tracking of errors during additions/subtractions. The need to keep exact operators means we cannot omit intermediate digits. Instead, we track both the number of digits before and after the decimal point.

The number of trailing digits, or words after the decimal point, compound as follows:

1. adding two numbers with L_1 and L_2 words after the decimal point each results in a number with $\max\{L_1, L_2\}$ words after the decimal point.
2. multiply two numbers with L_1 and L_2 words after the decimal point each results in a number with $L_1 + L_2$ words after the decimal point.

As $\max\{L_1, L_2\} \leqslant L_1 + L_2$ when L_1 and L_2 are non-negative, we will in general assume that when we multiply matrices with at most L_1 and L_2 words after the decimal point, the result has at most $L_1 + L_2$ words

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

after the decimal point. In particular, if Z is an operator with L_Z words after the decimal point, and its input B has L_B words after the decimal point, the output has at most $L_Z + L_B$ words after the decimal point.

Note that both of these bounds are for exact computations. The only round off errors come from round-off errors by dropping some of the digits, as the matrices themselves are created.

On the other hand, we need to bound the maximum magnitude of our operators. The number of digits before the decimal point is given by bounds on the magnitude of the numbers themselves. Such bounds also propagate nicely along multiplications.

LEMMA 3.3. *If the maximum magnitude of entries in two matrices Y and Z with dimension at most n are both at most α, then all entries in YZ have magnitude at most $n\alpha^2$ as well.*

Proof.

$$\left|(YZ)_{ij}\right| = \left|\sum_k Y_{ik}Z_{kj}\right| \leqslant \sum_k |Y_{ik}|\,|Z_{kj}| \leqslant n\alpha^2$$

□

Throughout our analyses, we will often rescale the matrices so that their max magnitudes are n^{-2}. This allows us to absorb any constant factor increases in magnitudes from multiplying these matrices by Lemma 3.3 because $(c \cdot n^{-2})^2 \leqslant n^{-2}$.

Finally, by doing FFT based fast multiplications for all numbers involved [CLRS09, HVDH19], we can multiple two numbers with an $O(\log n)$ factor in their lengths. This means that when handling two matrices with L_1 and L_2 words after the decimal point, and whose maximum magnitude is μ, the overhead caused by the word-lengths of the numbers involved is $\widetilde{O}(\mu + L_1 + L_2)$

Random Variables and Probability For our perturbations we use standard Gaussian $\mathcal{N}(0,1)$ random variables.

FACT 3.1. *For $x \sim \mathcal{N}(0,1)$, we have $\Pr(|x| \geqslant t) \leqslant \frac{2}{t\sqrt{2\pi}}e^{-t^2/2}$.*

Thus, with probability at least $1 - \exp(-n/2)$, a standard Gaussian variable is bounded by $\sqrt{n}$. We will use $O(n^2)$ such variables and condition on the event that all their norms are bounded by $\sqrt{n}$.

3.2 Main Technical Ingredients The main technical components of the analysis can be summarized as follows:

1. A can be perturbed so that its eigenvalues are separated (Theorem 3.1).
2. Such a separation implies a well-conditioned Krylov space, when it is initialized with sparse random Gaussian vectors. (Theorem 3.2)
3. This Krylov matrix can be solved efficiently using a combination of low displacement rank solvers and fast convolutions (Theorem 3.3).
4. The last few rows/columns can be solved efficiently via the Schur complement (Lemma 3.4).

Due to space constraints, we refer to the reader to the full version of the paper [PV20] for proofs of these components.

Anti-Concentration of semi-random matrices. A crucial part of our analysis is bounding the spectrum of semi-random matrices. Unfortunately, the highly developed literature on spectral properties of random matrices assumes independent entries or independent columns, which no longer hold in the semi-random case of K. However, getting tight estimates is not important for us (the running time is affected only by the logarithm of the gap/min value). So we adapt methods from random matrix theory to prove sufficient anti-concentration.

Specifically, after symmetrizing the potentially asymmetric input A by implicitly generating the operator A^TA, we need to bound (1) the minimum eigenvalue gap of the coefficient matrix after perturbation by a symmetric sparse matrix and (2) the minimum singular value of the block Krylov matrix constructed by multiplying with a sparse random matrix.

The first step of showing eigenvalue separation is needed because if A has a duplicate eigenvalue, the resulting Krylov space in it has rank at most $n-1$. We obtain such a separation by perturbing the matrix randomly: its analysis follows readily from recent results on separations of eigenvalues in random matrices by Luh and Vu [LV18].

THEOREM 3.1. *For any $n \times n$ symmetric positive definite matrix $\overline{A}$ with:*

1. *entries at most $1/n$,*
2. *eigenvalues at least $1/\kappa$ for some $\kappa \geqslant n^3$,*

and any probability where

$$p \geqslant \frac{300 \log \kappa \log n}{n}$$

the symmetrically random perturbed matrix A defined as

$$A_{ij} = A_{ji} \stackrel{\text{def}}{=} \begin{cases} \bar{A}_{ij} + \frac{1}{n^2\kappa}\mathcal{N}(0,1) & w.p.\ p, \\ \bar{A}_{ij} & w.p.\ 1-p, \end{cases}$$

Downloaded 06/27/21 to 143.215.38.55. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

with probability at least $1 - n^{-10}$ *has all eigenvalues separated by at least* $\kappa^{-5\log n}$.

Given this separation, we show that a random n-by-s B gives a m-step Krylov space matrix, as long as $n - ms = \Omega(m)$. Furthermore, we pick this B to be sparse in the columns, so we can quickly compute $B^T A^i B$.

THEOREM 3.2. *Let* A *be an* $n \times n$ *symmetric positive definite matrix with entries at most* $1/n$, *and* $\alpha_A < n^{-10}$ *a parameter such that:*

1. *all eigenvalues of* A *are at least* α_A, *and*
2. *all pairs of eigenvalues of* A *are separated by at least* α_A.

Let s *and* m *be parameters such that* $n^{0.01} \leqslant m \leqslant n^{\frac{1}{4}}$ *and* $s \cdot m \leqslant n - 5m$. *The* n*-by-*s *sparse Gaussian matrix* G^S *where each entry is set to* $\mathcal{N}(0,1)$ *with probability at least*

$$\frac{10000m^3 \log(1/\alpha_A)}{n}$$

leads to the Krylov space matrix

$$K = \left[\begin{array}{c|c|c|c|c} G^S & AG^S & A^2G^S & \ldots & A^{m-1}G^S \end{array} \right].$$

With probability at least $1 - n^{-2}$ *(over randomness in* G^S*),* K *has maximum singular value at most* n^2, *and minimum singular value at least* α_A^{5m}.

These bounds allow us to bound the length of numbers, and in turn running time complexity of solving $(AK)^T AK$.

Solvers for block Hankel matrices. An important ingredient in our algorithm is a numerically efficient solver for block Hankel matrices. For this we use the notion of displacement rank by Kailath, Kung and Morf [KKM79]. Its key statement is that any Schur Complement of a Toeplitz Matrix has displacement rank 2, and can be uniquely represented as the factorization of a rank 2 matrix. This combined with the fact that the displacement rank of the inverse is the same as that of the matrix is used to compute the Schur complement in the first super-fast/asymptotically fast solvers for Toeplitz matrices by Bitmead and Anderson [BA80]. Here we extend this to block Toeplitz/Hankel matrices and prove numerical stability, using the natural preservation of singular values of Schur complements. Specifically, the analysis from Demmel, Dumitriu, Holtz and Kleinberg [DDHK07] can be readily adapted to this setting.

THEOREM 3.3. *If* H *is an* $sm \times sm$ *symmetric* s*-block-Hankel matrix and* $0 < \alpha_H < (sm)^{-100}$ *is a parameter such that every contiguous square block-aligned minor of* H *containing the top-right or bottom left corner have minimum eigenvalue at least* α_H*:*

$$\sigma_{\min}\left(H_{\{1:i,(m-i+1):m\}}\right), \sigma_{\min}\left(H_{\{(m-i+1):m,1:i\}}\right) \geqslant \alpha_H$$

for all $1 \leqslant i \leqslant m$, *and all entries in* H *have magnitude at most* $(sm)^{-2}\alpha_H^{-1}$, *then for any error* ϵ, *we can pre-process* H *in time* $\widetilde{O}(ms^\omega \log(\alpha_H^{-1}\epsilon^{-1}))$ *to form* $\textsc{Solve}_H(\cdot, \epsilon)$ *that corresponds to a linear operator* Z_H *such that:*

1. *For any* $(ms) \times k$ *matrix* B *with max magnitude* $\|B\|_\infty$ *and* L_B *words after the decimal point,* $\textsc{Solve}_H(B, \epsilon)$ *returns* $Z_H B$ *in time*

$$\widetilde{O}\left(m \cdot \max\left\{s^{\omega-1}k, s^2k^{\omega-2}\right\} \cdot \left(\log\left(\frac{(1 + \|B\|_\infty)ms}{\alpha_H \epsilon}\right) + L_B\right)\right).$$

2. Z_H *is a high-accuracy approximation to* H^{-1}*:*

$$\left\|Z_H - H^{-1}\right\|_F \leqslant \epsilon.$$

3. *the entries of* Z_H *have at most* $O(\log^2 m \log(\alpha_H^{-1}\epsilon^{-1}))$ *words after the decimal point.*

The overhead of $\log^2 m$ in the word lengths of Z_H is from the $O(\log n)$ layers of recursion in Fast Fourier Transform times the $O(\log n)$ levels of recursion in the divide-and conquer block Schur complement algorithm. Note that the relation between the i^{th} block of $H = K^T K$ and K itself is:

$$\begin{aligned} H_{\{1:i,(n-i+1):m\}} &= K^T_{\{:,1:i\}} K_{\{:,(n-i+1):m\}} \\ &= K^T_{\{:,1:i\}} A^{m-i-1} K_{\{:,1:i\}}. \end{aligned}$$

So the min/max singular values of these matrices off by a factor of at most α_A^m from the singular value bounds on H itself.

It remains to pad K with random columns to make it a square matrix, Q. We bound the condition number of this matrix, and turn a solver for $M = (AK)^T(AK)$ to one for $(AQ)^T(AQ)$. Specifically, combining Theorems 3.2 and 3.3 leads to:

LEMMA 3.4. *Let* A *be an* $n \times n$ *symmetric positive definite matrix with entries at most* $1/n$, *and* $0 < \alpha_A < n^{-10}$ *a parameter such that:*

1. *all eigenvalues of* A *are at least* α_A, *and at most* α_A^{-1},

Downloaded 06/27/21 to 143.215.38.55. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

2. *all pairs of eigenvalues of A are separated by at least α_A,*

3. *all entries in A have magnitude at most α_A, and at most $O(\log(1/\alpha_A))$ words after the decimal point.*

For any parameter m such that $n^{0.01} \leqslant n \leqslant 0.01n^{0.2}$, the routine BlockKrylov *as shown in Figure 2 pre-processes A in time:*

1. *$O(n)$ matrix-vector multiplications of A against vectors with at most $O(m \log(1/\alpha_A))$ words both before and after the decimal point,*

2. *plus operations that cost a total of:*

$$\widetilde{O}\left(n^2 \cdot m^3 \cdot \log^2\left(1/\alpha_A\right) + n^{\omega} m^{2-\omega} \log\left(1/\alpha_A\right)\right).$$

and obtains a routine $\textsc{Solve}_A$ *that when given a vector b with L_b words after the decimal point, returns $Z_A b$ in time*

$$\widetilde{O}\left(n^2 m \cdot \left(\log\left(1/\alpha_A\right) + \|b\|_{\infty} + L_b\right)\right),$$

for some Z_A with at most $O(\log(1/\alpha_A))$ words after the decimal point such that

$$\left\|Z_A - A^{-1}\right\|_F \leqslant \alpha_A.$$

Note that we dropped ϵ as a parameter to simplify the statement of the guarantees. Such an omission is acceptable because if we want accuracy less than the eigenvalue bounds of A, we can simply run the algorithm with $\alpha_A \leftarrow \epsilon$ due to the pre-conditions holding upon α_A decreasing. With iterative refinement (e.g. [Saa03], it's also possible to lower separate the dependence on $\log(1/\epsilon)$ to linear instead of the cubic dependence on $\log(1/\alpha_H)$.

3.3 Proof of Main Theorem It remains to use the random perturbation specified in Theorem 3.1 and taking the outer product to reduce a general system to the symmetric, eigenvalue well separated case covered in Lemma 3.4.

The overall algorithm then takes a symmetrized version of the original matrix $\overline{A}$, perturbs it, and then converts the result of the block Krylov space method back. Its pseudocode is in Figure 5

Proof. (Of Theorem 1.1) Let $\widehat{A}$ be the copy of A scaled down by $n^2\theta_A$:

$$\widehat{A} = \frac{1}{n^2\theta_A} A = \frac{1}{n^2 \|A\|_{\infty}} A.$$

This rescaling gives us bounds on both the maximum and minimum entries of $\widehat{A}$. The rescaling ensures that the max magnitude of an entry in $\widehat{A}$ is at most $1/n^2$.

LinearEquationApproximation(A, b**: integer matrix/vector pair,** κ**: condition number bound for** A**,** ϵ**: error threshold.)**

1. Compute $\theta_A \leftarrow \|A\|_{\infty}$.

2. Generate random symmetric matrix R with

$$R_{ij} = R_{ji} = \frac{\epsilon}{n^{10}\kappa^2}\mathcal{N}(0,1)$$

with probability $\frac{O(\log(\kappa/\epsilon)\log n)}{n}$.

3. Implicitly generate

$$\widetilde{A} = \frac{1}{n^4\theta_A^2} A^T A + R$$

and its associated matrix-multiplication operator $\textsc{MatVec}_{\widetilde{A}}(\cdot, \delta)$.

4. Build solver for $\widetilde{A}$ via

$$\textsc{Solve}_{\widetilde{A}} \leftarrow \textsc{BlockKrylov}\left(\textsc{MatVec}_{\widetilde{A}}\left(\cdot, \delta\right), \left(n^8\kappa^2\epsilon^{-1}\right)^{-5\log n}, n^{\frac{\omega-2}{\omega+1}} nnz\left(A\right)^{\frac{\omega-2}{\omega+1}}\right).$$

5. Return

$$\frac{1}{n^4\theta_A^2} \cdot \textsc{Solve}_{\widetilde{A}}\left(A^T b\right).$$

Figure 5: Pseudocode for block Krylov space algorithm

Therefore its Frobenius norm, and in turn max singular value, is at most 1. On the other hand, the max singular of A is at least $\|A\|_{\infty} = \theta_A$: consider the unit vector that's 1 in the entry corresponding to the column containing the max magnitude entry of A, and 0 everywhere else. This plus the bound on condition number of κ gives that the minimum singular value of A is at least

$$\sigma_{\min}\left(A\right) \geqslant \frac{1}{\kappa}\sigma_{\max}\left(A\right) \geqslant \frac{\theta_A}{\kappa},$$

which coupled with the rescaling by $\frac{1}{n^2\theta_A}$ gives

$$\sigma_{\min}\left(\widehat{A}\right) \geqslant \frac{1}{n^2\theta_A} \cdot \frac{\theta_A}{\kappa} = \frac{1}{n^2\kappa}.$$

The matrix that we pass onto the block Krylov method, $\widetilde{A}$, is then the outer-product of $\widehat{A}$ plus a sparse random perturbation R with each entry is set (symmetrically when across the diagonal) to $\frac{\epsilon}{n^4\kappa}N(0,1)$ with probability $O(\log n \log(\kappa/\epsilon))/n$

$$\widetilde{A} \leftarrow \widehat{A}^T\widehat{A} + R.$$

Downloaded 06/27/21 to 143.215.38.55. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 06/27/21 to 143.215.38.55. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

This matrix $\widetilde{A}$ is symmetric. Furthermore, by Claim 3.1, we may assume that the max magnitude of an entry in R is at most $\epsilon n^{-9}\kappa^{-2}$, which gives

$$\|R\|_F \leqslant \frac{\epsilon}{n^8\kappa^2}.$$

So we also get that the max magnitude of an entry in $\widetilde{A}$ is still at most $2n^{-2}$. Taking this perturbation bound into Lemma 3.1 also gives that all eigenvalues of $\widetilde{A}$ are in the range

$$\left[\frac{1}{n^5\kappa^2}, 1\right].$$

By Theorem 3.1, the minimum eigenvalue separation in this perturbed matrix $\widetilde{A}$ is at least

$$\left(n^8\kappa^2\epsilon^{-1}\right)^{-5\log n}.$$

Also, by concentration bounds on the number of entries picked in R, its number of non-zeros is with high probability at most $O(n\log n\log(\kappa n/\epsilon)) = \widetilde{O}(n\log(\kappa/\epsilon))$.

As we only want an error of ϵ, we can round all entries in A to precision ϵ/κ without affecting the quality of the answer.

So we can invoke Lemma 3.4 with

$$\alpha_{\widetilde{A}} = \left(n^8\kappa^2\epsilon^{-1}\right)^{-5\log n},$$

which leads to a solve operator $Z_{\widetilde{A}}$ such that

$$\left\|Z_{\widetilde{A}} - \widetilde{A}^{-1}\right\|_F \leqslant \alpha_{\widetilde{A}} \leqslant \left(n^8\kappa^2\epsilon^{-1}\right)^{-5\log n} \leqslant \frac{\epsilon}{n^{40}\kappa^{10}}.$$

The error conversion lemma from Lemma 3.1 along with the condition that the min-singular value of $\widetilde{A}$ is at least $\frac{1}{n^5\kappa^2}$ implies that

$$\left\|Z_{\widetilde{A}}^{-1} - \widetilde{A}\right\|_F \leqslant \frac{\epsilon}{n^{30}\kappa^6}$$

or factoring into the bound on the size of R via triangle inequality:

$$\left\|Z_{\widetilde{A}}^{-1} - \widehat{A}^T\widehat{A}\right\|_F \leqslant \frac{2\epsilon}{n^8\kappa^4},$$

which when inverted again via Lemma 3.1 and the min singular value bound gives

$$\left\|Z_{\widetilde{A}} - \left(\widehat{A}^T\widehat{A}\right)^{-1}\right\|_F \leqslant \frac{\epsilon}{n^2}.$$

It remains to propagate this error across the rescaling in Step 5. Since $\widehat{A} = \frac{1}{n^2\theta_A}A$, we have

$$\left(A^TA\right)^{-1} = \frac{1}{n^4\theta_A^2}\left(\widehat{A}^T\widehat{A}\right)^{-1},$$

and in turn the error bound translates to

$$\left\|\frac{1}{n^4\theta_A^2}Z - \left(\widehat{A}^T\widehat{A}\right)^{-1}\right\|_F \leqslant \frac{\epsilon}{n^4\theta_A^2}.$$

The input on the other hand has

$$\Pi_A b = A\left(A^TA\right)^{-1}A^Tb,$$

so the error after multiplication by A is

$$A\left(\frac{1}{n^4\theta_A^2}Z - \left(\widehat{A}^T\widehat{A}\right)^{-1}\right)A^Tb,$$

which incorporating the above, as well as $\|A\|_2 \leqslant n\theta_A$ gives

$$\left\|A\left[\frac{1}{n^4\theta_A^2}ZA^Tb\right] - \pi_A b\right\|_2 \leqslant \frac{\epsilon}{n^3\theta_A}\left\|A^Tb\right\|_2.$$

On the other hand, because the max eigenvalue of A^TA is at most $\|A\|_F^2 \leqslant n^2\theta_A^2$, the minimum eigenvalue of $(A^TA)^{-1}$ is at least θ_A^{-2}. So we have

$$\|\Pi_A b\|_2 = \left\|A^Tb\right\|_{(A^TA)^{-1}} \geqslant \frac{1}{n^2\theta_A}\left\|A^Tb\right\|_2.$$

Combining the two bounds then gives that the error in the return value is at most $\epsilon\|\Pi_A b\|_2$.

For the total running time, the number of non-zeros in R implies that the total cost of mutliplying $\widetilde{A}$ against a vector with $\widetilde{O}(m\log(1/\alpha_A)) = \widetilde{O}(m\log n\log(\kappa/\epsilon)) = \widetilde{O}(m\log(\kappa/\epsilon))$ is

$$\widetilde{O}\left(\left(nnz\left(A\right)+n\right)m\log^2\left(\kappa/\epsilon\right)\right) \leqslant \widetilde{O}\left(nnz\left(A\right)m\log^2\left(\kappa/\epsilon\right)\right),$$

where the inequality of $nnz(A) \leqslant n$ follows preprocessing to remove empty rows and columns. So the total construction cost given in Lemma 3.4 simplifies to

$$O\left(n^2m^3\log^2\left(\kappa/\epsilon\right) + n^\omega m^{2-\omega}\log\left(\kappa/\epsilon\right) + n\cdot nnz\left(A\right)\cdot m\log\left(\kappa/\epsilon\right)\right)$$

The input vector, $\frac{1}{\theta_Y}y$ has max magnitude at most 1, and can thus be rounded to $O(\log(\kappa/\epsilon))$ words after the decimal point as well. This then goes into the solve cost with $\log(\|b\|_\infty) + L_b \leqslant O(\log(n\kappa/\epsilon))$, which gives a total of $\widetilde{O}(n^2m\log(\kappa/\epsilon))$, which is a lower order term compared to the construction cost. The cost of the additional multiplication in A is also a lower order term.

Optimizing m in this expression above based on only n and $nnz(A)$ gives that we should choose m so that

$$\max\left\{n\cdot nnz\left(A\right)m, n^2m^3\right\} = n^\omega m^{2-\omega},$$

or

$$\max\left\{n\cdot nnz\left(A\right)m^{\omega-1}, n^2m^{\omega+1}\right\} = n^\omega.$$

The first term implies

$$m \leqslant \left(n^{\omega-1}\cdot nnz\left(A\right)^{-1}\right)^{\frac{1}{\omega-1}} = n\cdot nnz\left(A\right)^{\frac{-1}{\omega+1}}$$

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

while the second term implies

$$m \leqslant n^{\frac{\omega-2}{\omega+1}}.$$

Substituting the minimum of these two bounds back into $n^{\omega}m^{2-\omega}$ and noting that $2-\omega \leqslant 0$ gives that the total runtime dependence on n and $nnz(A)$ is at most

$$n^{\omega} \cdot \max\left\{ n^{\frac{(\omega-2)(2-\omega)}{\omega+1}}, n^{2-\omega} \cdot nnz\left(A\right)^{\frac{-(2-\omega)}{\omega+1}} \right\}$$
$$= \max\left\{ n^{\frac{5\omega-4}{\omega+1}}, n^{2} \cdot nnz\left(A\right)^{\frac{\omega-2}{\omega+1}} \right\}.$$

Incorporating the trailing terms, then gives the the bound stated in Theorem 1.1, with c set to 2 plus the number of log factors hidden in the $\widetilde{O}$. ∎

4 Discussion

We have presented a faster solver for linear systems with moderately sparse coefficient matrices under bounded word-length arithmetic with logarithmic dependence on the condition number. This is the first separation between the complexity of matrix multiplication and solving linear systems in the bounded precision setting. While both our algorithm and analysis are likely improvable, we believe they demonstrate that there are still many sparse numerical problems and algorithms that remain to be better understood theoretically. We list a few avenues for future work.

Random Matrices. The asymptotic gap between our running time of about $n^{2.33}$ and the $O(n^{2.28})$ running time of computing inverses of sparse matrices over finite fields [EGG+06] is mainly due to the overhead our minimum singular value bound from Theorem 3.2, specifically the requirement of $\Omega(m^3)$ non-zeros per column on average. We conjecture that a similar bound holds for $\widetilde{O}(m)$ non-zeros per column, and also in the full Krylov space case.

1. Can we lower bound the min singular value of a block Krylov space matrix generated from a random matrix with $\widetilde{O}(m)$ non-zeros per column?

2. Can we lower bound the min singular value of a block Krylov space matrix where $m \cdot s = n$ for general values of s (block size) and m (number of steps)?

The second improvement of removing the additional $\Omega(m)$ columns would not give asymptotic speedups. It would however remove the need for the extra steps (padding with random Gaussian columns) in Lemma 3.4. Such a bound for the square case would likely require developing new tools for analyzing matrix anti-concentration.

3. Are there general purpose bounds on the min singular value of a sum of random matrices, akin to matrix concentration bounds (which focus on the max singular value) [RV10, Tro15].

The connections with random matrix theory can also be leveraged in the reverse direction:

4. Can linear systems over random matrices with i.i.d. entries be solved faster?

An interesting case here is sparse matrices with non-zeros set to ± 1 independently. Such matrices have condition number $\Theta(n^2)$ with constant probability [RV10], which means that the conjugate gradient algorithm has bit complexity $O(n \cdot nnz)$ on such systems. Therefore, we believe these matrices present a natural starting point for investigating the possibility of faster algorithms for denser matrices with $nnz > \Omega(n^{\omega-1})$.

Numerical Algorithms. The bounded precision solver for block Hankel matrices in Theorem 3.3 is built upon the earliest tools for speeding up solvers for such structured matrices [KKM79, BA80], as well as the first sparsified block Cholesky algorithm for solving graph Laplacians [KLP+16]. We believe the more recent developments in solvers for Hankel/Toeplitz matrices [XXCB14] as well as graph Laplacians [KS16] can be incorporated to give better and more practical routines for solving block-Hankel/Toeplitz matrices.

5. Is there a superfast solver under bounded precision for block Hankel/Toeplitz matrices that does not use recursion?

It would also be interesting to investigate whether recent developments in randomized numerical linear algebra can work for Hankel / Toeplitz matrices. Some possible questoins there are:

6. Can we turn $m \times s^{\omega}$ into $O(ms^2 + s^{\omega})$ using more recent developments sparse projections (e.g. CountSketch / sparse JL / sparse Gaussian instead of a dense Gaussian).

7. Is there an algorithm that takes a rank r factorization of $I - XY \in \Re^{n \times n}$, and computes in time $\widetilde{O}(n\text{poly}(r))$ the a rank r factorization/approximation of $I - YX$?

Another intriguing question is the extensions of this approach to the high condition number, or exact integer solution, setting. Here the current best running time bounds are via p-adic representations of fractions [Dix82], which are significantly less understood compared to decimal point based representations. In the dense case, an algorithm via shifted p-adic numbers by Storjohann [Sto05] achieves an $O(n^{\omega})$ bit complexity. Therefore, it is natural to hope for a similar

Downloaded 06/27/21 to 143.215.38.55. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

$\widetilde{O}(n \cdot nnz)$ bit complexity algorithm for producing exact integer solutions. A natural starting point could be the role of low-rank sketching in solvers that take advantage of displacement rank, i.e., extending the p-adic algorithms to handle low rank matrices:

8. Is there an $O(n \cdot r^{\omega-1})$ time algorithm for exactly solving linear regression problems involving an n-by-n integer matrix with rank r?

Finally, we note that the paper by Eberly et al. [EGG+06] that proposed block-Krylov based methods for matrix inversion also included experimental results that demonstrated good performances as an exact solver over finite fields. It might be possible to practically evaluate block-Krylov type methods for solving general systems of linear equations. Here it is worth remarking that even if one uses naive $\Theta(n^3)$ time matrix multiplication, both the Eberly et al. algorithm [EGG+07] (when combined with p-adic representations), as well as our algorithm, still take sub-cubic time.

Acknowldgements

Richard Peng was supported in part by NSF CAREER award 1846218, and Santosh Vempala by NSF awards AF-1909756 and AF-2007443. We thank Mark Giesbrecht for bringing to our attention the works on block-Krylov space algorithms; Yin Tat Lee for discussions on random linear systems; Yi Li, Anup B. Rao, and Ameya Velingker for discussions about high dimensional concentration and anti-concentration bounds; Mehrdad Ghadiri, He Jia and anonymous reviewers for comments on earlier versions of this paper.

References

[BA80] Robert R Bitmead and Brian DO Anderson. Asymptotically fast solution of Toeplitz and related systems of linear equations. *Linear Algebra and its Applications*, 34:103–116, 1980.

[BHK20] Avrim Blum, John Hopcroft, and Ravindran Kannan. *Foundations of Data Science*. Cambridge University Press, 2020.

[BHV08] Erik G. Boman, Bruce Hendrickson, and Stephen A. Vavasis. Solving elliptic finite element systems in near-linear time with support preconditioners. *SIAM J. Numer. Anal.*, 46(6):3264–3284, 2008.

[BJMS17] Alin Bostan, C-P Jeannerod, Christophe Mouilleron, and É Schost. On matrices with displacement structure: Generalized operators and faster algorithms. *SIAM Journal on Matrix Analysis and Applications*, 38(3):733–775, 2017.

[BL94] Bernhard Beckermann and George Labahn. A uniform approach for the fast computation of matrix-type padé approximants. *SIAM Journal on Matrix Analysis and Applications*, 15(3):804–823, 1994.

[Blu04] Lenore Blum. Computing over the reals: Where turing meets newton. *Notices of the AMS*, 51(9):1024–1034, 2004.

[BSS+89] Lenore Blum, Mike Shub, Steve Smale, et al. On a theory of computation and complexity over the real numbers: *np*-completeness, recursive functions and universal machines. *Bulletin (New Series) of the American Mathematical Society*, 21(1):1–46, 1989.

[Cip00] Barry A Cipra. The best of the 20th century: Editors name top 10 algorithms. *SIAM news*, 33(4):1–2, 2000. Available at: https://archive.siam.org/pdf/news/637.pdf.

[CKK+18] 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 Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 898–909. IEEE Computer Society, 2018. Available at: https://arxiv.org/abs/1811.10722.

[CKP+17] 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 Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 410–419. ACM, 2017. Available at: https://arxiv.org/abs/1611.00755.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.

[CRW93] Ronald Coifman, Vladimir Rokhlin, and Stephen Wandzura. The fast multipole method for the wave equation: A pedestrian prescription. *IEEE Antennas and Propagation magazine*, 35(3):7–12, 1993.

[CW87] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 1–6. ACM, 1987.

[DDH07] James Demmel, Ioana Dumitriu, and Olga Holtz. Fast linear algebra is stable. *Numerische Mathematik*, 108(1):59–91, 2007.

[DDHK07] James Demmel, Ioana Dumitriu, Olga Holtz, and Robert Kleinberg. Fast matrix multiplication is stable. *Numerische Mathematik*, 106(2):199–224, 2007.

[Dix82] John D Dixon. Exact solution of linear equations using P-adic expansions. *Numerische Mathematik*, 40(1):137–141, 1982.

[DMM08] Petros Drineas, Michael W Mahoney, and Shan Muthukrishnan. Relative-error cur matrix decompositions. *SIAM Journal on Matrix Analysis and Applications*, 30(2):844–881, 2008. Available at:.

Downloaded 06/27/21 to 143.215.38.55. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 06/27/21 to 143.215.38.55. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

[DRSL16] Timothy A Davis, Sivasankaran Rajamanickam, and Wissam M Sid-Lakhdar. A survey of direct methods for sparse linear systems. *Acta Numerica*, 25:383–566, 2016.

[EGG+06] Wayne Eberly, Mark Giesbrecht, Pascal Giorgi, Arne Storjohann, and Gilles Villard. Solving sparse rational linear systems. In *Symbolic and Algebraic Computation, International Symposium, ISSAC 2006, Genoa, Italy, July 9-12, 2006, Proceedings*, pages 63–70, 2006.

[EGG+07] 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.

[EH10] Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Soc., 2010.

[GO89] Gene H Golub and Dianne P O'Leary. Some history of the conjugate gradient and lanczos algorithms: 1948–1976. *SIAM review*, 31(1):50–102, 1989.

[Gra06] Robert M Gray. *Toeplitz and circulant matrices: A review*. now publishers inc, 2006.

[Gre96] Keith D. Gremban. *Combinatorial Preconditioners for Sparse, Symmetric, Diagonally Dominant Linear Systems*. PhD thesis, Carnegie Mellon University, Pittsburgh, October 1996. CMU CS Tech Report CMU-CS-96-123.

[GTVDV96] KA Gallivan, S Thirumalai, Paul Van Dooren, and V Vermaut. High performance algorithms for toeplitz and block toeplitz matrices. *Linear algebra and its applications*, 241:343–388, 1996.

[HS52] Magnus Rudolph Hestenes and Eduard Stiefel. *Methods of conjugate gradients for solving linear systems*, volume 49-1. NBS Washington, DC, 1952.

[HVDH19] David Harvey and Joris Van Der Hoeven. Polynomial multiplication over finite fields in time $o(nlogn)$. Available at `https://hal.archives-ouvertes.fr/hal-02070816/document`, 2019.

[KKM79] Thomas Kailath, Sun-Yuan Kung, and Martin Morf. Displacement ranks of matrices and linear equations. *Journal of Mathematical Analysis and Applications*, 68(2):395–407, 1979.

[KLP+16] 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*, pages 842–850. ACM, 2016. Available at http://arxiv.org/abs/1512.01892.

[KMP12] Ioannis Koutis, Gary L. Miller, and Richard Peng. A fast solver for a class of linear systems. *Communications of the ACM*, 55(10):99–107, October 2012. Available at https://cacm.acm.org/magazines/2012/10/155538-a-fast-solver-for-a-class-of-linear-systems/fulltext.

[KS16] Rasmus Kyng and Sushant Sachdeva. Approximate gaussian elimination for laplacians - fast, sparse, and simple. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 573–582, 2016. Available at: https://arxiv.org/abs/1605.02353.

[KWZ20] 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. Available at: https://dl.acm.org/doi/pdf/10.5555/3381089.3381106.

[Kyn17] Rasmus Kyng. *Approximate Gaussian Elimination*. PhD thesis, Yale University, 2017. Available at: `http://rasmuskyng.com/rjkyng-dissertation.pdf`.

[KZ17] 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. Available at: https://arxiv.org/abs/1705.02944.

[Lan50] Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 1950.

[LG14] Francois Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th international symposium on symbolic and algebraic computation*, pages 296–303. ACM, 2014. Available at http://arxiv.org/abs/1401.7714.

[LLY11] Lin Lin, Jianfeng Lu, and Lexing Ying. Fast construction of hierarchical matrix representation from matrix–vector multiplication. *Journal of Computational Physics*, 230(10):4071–4087, 2011.

[LS92] George Labahn and Tamir Shalom. Inversion of toeplitz matrices with only two standard equations. *Linear algebra and its applications*, 175:143–158, 1992.

[LV18] Kyle Luh and Van Vu. Sparse random matrices have simple spectrum. *arXiv preprint arXiv:1802.03662*, 2018.

[MMS18] Cameron Musco, Christopher Musco, and Aaron Sidford. Stability of the lanczos method for matrix function approximation. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1605–1624, 2018. Available at: https://arxiv.org/abs/1708.07788.

[NTV17] Hoi Nguyen, Terence Tao, and Van Vu. Random matrices: tail bounds for gaps between eigenvalues. *Probability Theory and Related Fields*, 167(3-4):777–816, 2017.

[Pan84] Victor Y. Pan. *How to Multiply Matrices Faster*, volume 179 of *Lecture Notes in Computer Science*. Springer, 1984.

[PS85] Franco P Preparata and Michael Ian Shamos. *Com-*

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited

Downloaded 06/27/21 to 143.215.38.55. Redistribution subject to SIAM license or copyright; see https://epubs.siam.org/page/terms

putational geometry. an introduction. Springer-Verlag New York, 1985.

[PV20] Richard Peng and Santosh S. Vempala. Solving sparse linear systems faster than matrix multiplication. *CoRR*, abs/2007.10254, 2020.

[RV07] Mark Rudelson and Roman Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *J. ACM*, 54(4):21, 2007.

[RV10] Mark Rudelson and Roman Vershynin. Non-asymptotic theory of random matrices: extreme singular values. In *Proceedings of the International Congress of Mathematicians 2010 (ICM 2010) (In 4 Volumes) Vol. I: Plenary Lectures and Ceremonies Vols. II–IV: Invited Lectures*, pages 1576–1602. World Scientific, 2010.

[Saa03] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003. Available at http://www-users.cs.umn.edu/~saad/toc.pdf.

[SST06] Arvind Sankar, Daniel A Spielman, and Shang-Hua Teng. Smoothed analysis of the condition numbers and growth factors of matrices. *SIAM Journal on Matrix Analysis and Applications*, 28(2):446–476, 2006.

[ST11] D. Spielman and S. Teng. Spectral sparsification of graphs. *SIAM Journal on Computing*, 40(4):981–1025, 2011. Available at http://arxiv.org/abs/0808.4134.

[ST14] D. Spielman and S. Teng. Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. *SIAM Journal on Matrix Analysis and Applications*, 35(3):835–885, 2014. Available at http://arxiv.org/abs/cs/0607105.

[Sto05] Arne Storjohann. The shifted number system for fast linear algebra on integer matrices. *Journal of Complexity*, 21(4):609–650, 2005. Available at: https://cs.uwaterloo.ca/~astorjoh/shifted.pdf.

[Str69] Volker Strassen. Gaussian elimination is not optimal. *Numerische mathematik*, 13(4):354–356, 1969.

[SV13] Sushant Sachdeva and Nisheeth K Vishnoi. Faster algorithms via approximation theory. *Theoretical Computer Science*, 9(2):125–210, 2013.

[Tro15] Joel A. Tropp. An introduction to matrix concentration inequalities. *Found. Trends Mach. Learn.*, 8(1-2):1–230, 2015.

[Tur48] Alan M Turing. Rounding-off errors in matrix processes. *The Quarterly Journal of Mechanics and Applied Mathematics*, 1(1):287–308, 1948.

[TV10] Terence Tao and Van H. Vu. Smooth analysis of the condition number and the least singular value. *Math. Comput.*, 79(272):2333–2352, 2010. Available at: https://arxiv.org/abs/0805.3167.

[Vai89] P. M. Vaidya. Speeding-up linear programming using fast matrix multiplication. In *30th Annual Symposium on Foundations of Computer Science*, pages 332–337, Oct 1989.

[Wil12] Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 887–898. ACM, 2012.

[Woo14] David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1-2):1–157, 2014. Available at: `https://arxiv.org/abs/1411.4357`.

[XB90] Guo-liang Xu and Adhemar Bultheel. Matrix padé approximation: definitions and properties. *Linear algebra and its applications*, 137:67–136, 1990.

[XCGL10] Jianlin Xia, Shivkumar Chandrasekaran, Ming Gu, and Xiaoye S Li. Fast algorithms for hierarchically semiseparable matrices. *Numerical Linear Algebra with Applications*, 17(6):953–976, 2010.

[XXCB14] Yuanzhe Xi, Jianlin Xia, Stephen Cauley, and Venkataramanan Balakrishnan. Superfast and stable structured solvers for toeplitz least squares via randomized sampling. *SIAM Journal on Matrix Analysis and Applications*, 35(1):44–72, 2014.

[XXG12] Jianlin Xia, Yuanzhe Xi, and Ming Gu. A superfast structured solver for toeplitz linear systems via randomized sampling. *SIAM Journal on Matrix Analysis and Applications*, 33(3):837–858, 2012.

[Ye11] Yinyu Ye. *Interior point algorithms: theory and analysis*, volume 44. John Wiley & Sons, 2011.

[Zha18] Peng Zhang. *Hardness and Tractability For Structured Numerical Problems*. PhD thesis, Georgia Institute of Technology, 2018. Available at: `https://drive.google.com/file/d/1KEZNzna-Y7y6rDERKuBFvFuU-hfHUMR3/view`.

Copyright © 2021 by SIAM
Unauthorized reproduction of this article is prohibited