On Solving Linear Systems in Sublinear Time

Alexandr Andoni[1]
Columbia University, New York, NY, USA
andoni@cs.columbia.edu

Robert Krauthgamer[2]
Weizmann Institute of Science, Rehovot, Israel
robert.krauthgamer@weizmann.ac.il

Yosef Pogrow
Weizmann Institute of Science, Rehovot, Israel
yosef.pogrow@weizmann.ac.il

Abstract

We study *sublinear* algorithms that solve linear systems locally. In the classical version of this problem the input is a matrix $S \in \mathbb{R}^{n \times n}$ and a vector $b \in \mathbb{R}^n$ in the range of S, and the goal is to output $x \in \mathbb{R}^n$ satisfying $Sx = b$. For the case when the matrix S is symmetric diagonally dominant (SDD), the breakthrough algorithm of Spielman and Teng [STOC 2004] approximately solves this problem in near-linear time (in the input size which is the number of non-zeros in S), and subsequent papers have further simplified, improved, and generalized the algorithms for this setting.

Here we focus on computing one (or a few) coordinates of x, which potentially allows for sublinear algorithms. Formally, given an index $u \in [n]$ together with S and b as above, the goal is to output an approximation $\hat{x}_u$ for x^*_u, where x^* is a fixed solution to $Sx = b$.

Our results show that there is a qualitative gap between SDD matrices and the more general class of positive semidefinite (PSD) matrices. For SDD matrices, we develop an algorithm that approximates a single coordinate x_u in time that is polylogarithmic in n, provided that S is sparse and has a small condition number (e.g., Laplacian of an expander graph). The approximation guarantee is additive $|\hat{x}_u - x^*_u| \leq \epsilon \|x^*\|_\infty$ for accuracy parameter $\epsilon > 0$. We further prove that the condition-number assumption is necessary and tight.

In contrast to the SDD matrices, we prove that for certain PSD matrices S, the running time must be at least polynomial in n (for the same additive approximation), even if S has bounded sparsity and condition number.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Linear systems, Laplacian solver, Sublinear time, Randomized linear algebra

Digital Object Identifier 10.4230/LIPIcs.ITCS.2019.3

Related Version Full version at `arXiv:1809.02995`.

Acknowledgements The authors thank anonymous reviewers for suggesting additional relevant references.

[1] Work supported in part by Simons Foundation (#491119), NSF grants CCF-1617955 and CCF-1740833.
[2] Work supported in part by ONR Award N00014-18-1-2364, the Israel Science Foundation grant #1086/18, a Minerva Foundation grant, and a Google Faculty Research Award.

1 Introduction

Solving linear systems is a fundamental problem in many areas. A basic version of the problem has as input a matrix $A \in \mathbb{R}^{n \times n}$ and a vector $b \in \mathbb{R}^n$, and the goal is to find $x \in \mathbb{R}^n$ such that $Ax = b$. The fastest known algorithm for general A is by a reduction to matrix multiplication, and takes $O(n^\omega)$ time, where $\omega < 2.373$ [19] is the matrix multiplication exponent. When A is sparse, one can do better (by applying the conjugate gradient method to the equivalent positive semidefinite (PSD) system $A^\mathsf{T}Ax = A^\mathsf{T}b$, see for example [32]), namely, $O(mn)$ time where $m = \text{nnz}(A)$ is the number of non-zeros in A. This $O(mn)$ bound for exact solvers assumes exact arithmetic, and in practice, one seeks fast approximate solvers.

One interesting subclass of PSD matrices is that of symmetric diagonally dominant (SDD) matrices.[3] Many applications require solving linear systems in SDD matrices, and most notably their subclass of graph-Laplacian matrices, see e.g. [32, 36, 14]. Solving SDD linear systems received a lot of attention in the past decade after the breakthrough result by Spielman and Teng in 2004 [31], showing that a linear system in SDD matrix S can be solved approximately in near-linear time $O(m\log^{O(1)} n \log \frac{1}{\epsilon})$, where $m = \text{nnz}(S)$ and $\epsilon > 0$ is an accuracy parameter. A series of improvements led to the state-of-the-art SDD solver of Cohen et al. [14] that runs in near-linear time $O(m\sqrt{\log n}(\log\log n)^{O(1)} \log \frac{1}{\epsilon})$. Recent improvements extend to connection Laplacians [24]. Obtaining similar results for all PSD matrices remains a major open question.

Motivated by fast linear-system solvers in alternative models, here we study which linear systems can be solved in *sublinear time*. We can hope for such sublinear times if only one (or a few) coordinates of the solution $x \in \mathbb{R}^n$ are sought. Formally, given a matrix $S \in \mathbb{R}^{n \times n}$, a vector $b \in \mathbb{R}^n$, and an index $u \in [n]$, we want to approximate the coordinate x_u of a solution $x \in \mathbb{R}^n$ to the linear system $Sx = b$ (assume for now the solution is unique), and we want the running time to be *sublinear in* n.

Our main contribution is a *qualitative separation* between the class of SDD matrices and the larger class of PSD matrices, as follows. For well-conditioned SDD matrices S, we develop a (randomized) algorithm that approximates a single coordinate x_u fast – in polylog(n) time. In contrast, for some well-conditioned PSD (but not SDD) matrices S, we show that the same task requires $n^{\Omega(1)}$ time. In addition, we justify the dependence on the condition number.

Our study is partly motivated by the advent of *quantum* algorithms that solve linear systems in sublinear time, which were introduced in [20], and subsequently improved in [2, 11], and meanwhile used for a number of (quantum) machine learning algorithms (see, e.g., the survey [15]). In particular, the model in [20] considers a system $Ax = b$ given: (1) oracle access to entries of A (including fast access to the j-th non-zero entry in the i-th row), and (2) a fast black-box procedure to prepare a quantum state $|b\rangle = \sum_i \frac{b_i|i\rangle}{\|\sum_i b_i|i\rangle\|}$. Then, if the matrix A has condition number κ, at most d non-zeros per row/column, and $\|A\| = 1$, their quantum algorithm runs in time poly$(\kappa, d, 1/\epsilon)$, and outputs a quantum state $|\hat{x}\rangle$ within ℓ_2-distance ϵ from $|x\rangle = \frac{\sum_i x_i|i\rangle}{\|\sum_i x_i|i\rangle\|}$. The runtime was later improved in [11] to depend logarithmically on $1/\epsilon$. (The original goal of [20] was different – to output a "classical" value, a linear combination of $|x\rangle$ – and for this goal the improved dependence on $1/\epsilon$ is not possible unless $BQP = PP$.) These quantum sublinear-time algorithms raise the question whether there are analogous classical algorithms for the same problems; for example, a very recent success story is a classical algorithm [35] for a certain variant of recommendation

[3] A symmetric matrix $S \in \mathbb{R}^{n \times n}$ is called SDD if $S_{ii} \geq \sum_{j \neq i} |S_{ij}|$ for all $i \in [n]$.

systems, inspired by an earlier quantum algorithm [22]. Our lower bound precludes a classical analogue to the aforementioned linear-system solver, which works for all matrices A and in particular for PSD ones.

Problem Formulation. To formalize the problem, we need to address a common issue for linear systems – they may be underdetermined and thus have many solutions x, which is a nuissance when solving for a single coordinate. We require that the algorithm approximates a single solution x^*, in the sense that invoking the algorithm with different indices $u \in [n]$ will output coordinates that are all consistent with one "global" solution. This formulation follows the concept of Local Computation Algorithms, see Section 1.3.

Our formal requirement is thus as follows. Given a matrix $S \in \mathbb{R}^{n \times n}$, a vector $b \in \mathbb{R}^n$ in the range (column space) of S, and an accuracy parameter $\epsilon > 0$, there exists $x^* \in \mathbb{R}^n$ satisfying $Sx^* = b$, such that upon query $u \in [n]$ the (randomized) algorithm outputs $\hat{x}_u$ that satisfies

$$\forall u \in [n], \qquad \Pr\left[|\hat{x}_u - x^*_u| \leq \epsilon ||x^*||_\infty\right] \geq \tfrac{3}{4}. \tag{1}$$

This guarantee corresponds (modulo amplification of the success probability) to reporting a solution $\hat{x} \in \mathbb{R}^n$ with $\|\hat{x} - x^*\|_\infty \leq \epsilon ||x^*||_\infty$. We remark that the guarantee in [31] is different, that $||\hat{x} - x^*||_S \leq \epsilon ||x^*||_S$ where $||y||_S \stackrel{\text{def}}{=} \sqrt{y^\mathsf{T} S y}$, see also Section 1.4.

Basic Notation. Given a (possibly edge-weighted) undirected graph $G = (V, E)$, we assume for convenience $V = [n]$. Its Laplacian is the matrix $L_G \stackrel{\text{def}}{=} D - A \in \mathbb{R}^{n \times n}$, where A is the (weighted) adjacency matrix of G, and D is the diagonal matrix of (weighted) degrees in G. It is well-known that all Laplacians are SDD matrices, which in turn are always PSD.

The *sparsity* of a matrix is the maximum number of non-zero entries in a single row/column. The *condition number* of a PSD matrix S, denoted $\kappa(S)$, is the ratio between its largest and smallest non-zero eigenvalues.[4] For example, for the Laplacian L_G of a connected d-regular graph G, let $\mu_1 \leq \ldots \leq \mu_n$ denote its eigenvalues, then the condition number is $\kappa(L_G) = \Theta(\frac{d}{\mu_2})$. This follows from two well-known facts, that $\mu_n \in [d, 2d]$, and that $\mu_2 > \mu_1 = 0$ if G is connected (μ_2 is called the spectral gap). Throughout, $||A||$ denotes the spectral norm of a matrix A, and A^+ denotes the Moore-Penrose pseudo-inverse of A.[5]

1.1 Our Results

Below we describe our results, which include both algorithms and lower bounds. First, we present a polylogarithmic-time algorithm for the simpler case of Laplacian matrices, and then we generalize it to all SDD matrices. We further prove two lower bounds, which show that our algorithms cannot be substantially improved to handle more general inputs or to run faster. The first lower bound shows that general PSD matrices require polynomial time, thereby showing a strong separation from the SDD case. The second one shows that our SDD algorithm's dependence on the condition number is necessary and in fact near-tight.

[4] Our definition is in line with the standard one, for a general matrix A, which uses singular values instead of eigenvalues. If A is singular, one could alternatively define $\kappa(A) = \infty$, which would only make the problem easier (say to bound performance in terms of κ), see e.g. [32].

[5] For a PSD matrix $A \in \mathbb{R}^{n \times n}$, let its eigen-decomposition be $A = \sum_{i=1}^n \lambda_i u_i u_i^\mathsf{T}$, then the Moore-Penrose pseudo-inverse of A is $A^+ = \sum_{i:\lambda_i > 0} \frac{1}{\lambda_i} u_i u_i^\mathsf{T}$.

Algorithm for Laplacian matrices. We first present our simpler algorithm for linear systems in Laplacians with a bounded condition number.

▶ **Theorem 1.1** (Laplacian Solver, see Section 2). *There exists a randomized algorithm, that given input $\langle G, b, u, \epsilon, \bar{\kappa} \rangle$, where*

- *$G = (V, E)$ is a connected d-regular graph given as an adjacency list,*
- *$b \in \mathbb{R}^n$ is in the range of L_G (equivalently, orthogonal to the all-ones vector),*
- *$u \in [n]$, $\epsilon > 0$, and*
- *$\bar{\kappa} \geq 1$ is an upper bound on the condition number $\kappa(L_G)$,*

the algorithm outputs $\hat{x}_u \in \mathbb{R}$ with the following guarantee. Letting $x^* = L_G^+ b$, we have

$$\forall u \in [n], \qquad \Pr\left[|\hat{x}_u - x^*_u| \leq \epsilon \cdot \|x^*\|_\infty\right] \geq 1 - \tfrac{1}{s},$$

and the algorithm runs in time $O(d\epsilon^{-2} s^3 \log s)$, for suitable $s = \Theta(\bar{\kappa} \log(\epsilon^{-1} \bar{\kappa}\, n))$.

A few extensions of the theorem follow easily from our proof. First, if the algorithm is given also an upper bound B_{up} on $||b||_0$, then the expression for s can be refined by replacing n with $B_{up} \leq n$. Second, we can improve the running time to $O(\epsilon^{-2} s^3 \log s)$ whenever the representation of G allows to sample a uniformly random neighbor of a vertex in constant time. Third, the algorithm has an (essentially) cubic dependence on the condition number $\kappa(L_G)$, which can be improved to quadratic if we allow a preprocessing of G (or, equivalently if we only count the number of probes into b). Later we show that this quadratic dependence is *near-optimal*.

Algorithm for SDD matrices. We further design an algorithm for SDD matrices with bounded condition number. The formal statement, which appears in Theorem 3.1, is a natural generalization of Theorem 1.1 with two differences. One difference is that a natural solution to the system $Sx = b$ is $x = S^+ b$, but our method requires S to have normalized diagonal entries, and thus we aim at another solution x^*, constructed as follows. Define

$$D \stackrel{\text{def}}{=} \text{diag}(S_{11}, ..., S_{nn}) \quad \text{and} \quad \tilde{S} \stackrel{\text{def}}{=} D^{-1/2} S D^{-1/2}, \tag{2}$$

then our linear system can be written as $\tilde{S}(D^{1/2}x) = D^{-1/2}b$, which has a solution

$$x^* \stackrel{\text{def}}{=} D^{-1/2} \tilde{S}^+ D^{-1/2} b, \tag{3}$$

which is expressed using the pseudo-inverse of $\tilde{S}$ rather than of S.

A second difference is that Theorem 3.1 makes no assumptions about the multiplicity of the eigenvalue 0 of $\tilde{S}$, e.g., if S is a graph Laplacian, then the graph need not be connected. The assumptions needed to achieve a polylogarithmic time, beyond $\tilde{S}$ having a bounded condition number,[6] are only that a random "neighbor" in the graph corresponding to S can be sampled quickly, and that $\frac{\max_{i \in [n]} D_{ii}}{\min_{i \in [n]} D_{ii}} \leq \text{poly}(n)$, which holds if S has polynomially-bounded entries.

Lower Bound for PSD matrices. Our first lower bound shows that the above guarantees cannot be obtained for a general PSD matrix, even if we are allowed to preprocess the matrix S, and only count probes into b. The proof employs a PSD matrix S that is invertible (i.e., positive definite), in which case the linear system $Sx = b$ has a unique solution $x = S^{-1}b$.

[6] We cannot phrase our requirements in terms of $\kappa(S)$, because we are not aware of a non-trivial relationship between it and $\kappa(\tilde{S})$.

▶ **Theorem 1.2** (Lower Bound for PSD Systems, see Section 4). *For every large enough n, there exists an invertible PSD matrix $S \in \mathbb{R}^{n\times n}$ with uniformly bounded sparsity $d = O(1)$ and condition number $\kappa(S) \le 3$, and a distinguished index $u \in [n]$, which satisfy the following. Every randomized algorithm that, given as input $b \in \mathbb{R}^n$, outputs $\hat{x}_u$ satisfying*

$$\Pr\left[|\hat{x}_u - x^*_u| \le \tfrac{1}{5}\|x^*\|_\infty\right] \ge \tfrac{6}{7},$$

where $x^* = S^{-1}b$, must probe $n^{\Omega(1/d^2)}$ coordinates of b (in the worst case).

Dependence on Condition Number. The second lower bound shows that our SDD algorithm has a *near-optimal* dependence on the condition number of S, even if we are allowed to preprocess the matrix S, and only count probes into b. The lower bound holds even for Laplacian matrices. Here and throughout, we use $\tilde{O}(f)$ to hide polylogarithmic factors in f or in the input size, i.e., it stands for $O(f \log^{O(1)}(f+n))$, and similarly for $\tilde{\Omega}(f)$.

▶ **Theorem 1.3** (Lower Bound for Laplacian Systems). *For every large enough n and $k \le O(n^{1/2}/\log n)$, there exist an unweighted graph $G = ([n], E)$ with maximum degree 4 and whose Laplacian L_G has condition number $\kappa(L_G) = O(k)$, and a distinguished edge (u, v) in G, which satisfy the following. Every randomized algorithm that, given input b in the range of L_G, succeeds with probability $2/3$ to approximate $x_u - x_v$ within additive error $\epsilon\|x^*\|$ for $\epsilon = \Theta(1/\log n)$ and any solution $x^* \in \mathbb{R}^n$ for $L_G x = b$, must probe $\tilde{\Omega}(k^2)$ coordinates of b (in the worst case).*

The proof of this result is omitted here but appears in the full version.

Applications. An example application of our algorithmic results is computing the effective resistance between a pair of vertices u, v in a graph G (given u,v and G as input). It is well known that the effective resistance, denoted $\mathrm{R}_{\mathrm{eff}}(u, v)$, can be expressed as $x_u - x_v$, where x solves $L_G x = e_u - e_v$. The spectral-sparsification algorithm of Spielman and Srivastava [33] relies on a near-linear time algorithm (that they devise) for approximating the effective resistances of all edges in G. For unweighted graphs, there is also a faster algorithm [25] that runs in time $\tilde{O}(n)$, which is sublinear in the number of edges, and approximates effective resistances within a larger factor polylog(n). In a d-regular expander G, it is the effective resistance of every vertex pair is $\Theta(1/d)$, and in this case our algorithm from Theorem 1.1 can quickly compute, for any single pair, an arbitrarily good approximation (factor $1+\epsilon$). Indeed, observe that we can use $B_{up} = 2$, hence the running time is $O(\frac{1}{\epsilon^2}\,\mathrm{polylog}\,\frac{1}{\epsilon})$, independently of n. The additive accuracy is $\epsilon\|x\|_\infty$, where $x \in \mathbb{R}^n$ represents the vertex potentials when imposing a unit of current from u to v, or equivalently, imposing a potential difference $\mathrm{R}_{\mathrm{eff}}(u, v)$ between u and v, which implies that every $x_i \in [x_u, x_v]$. By considering a solution with $x_u = 0$ we get $\|x\|_\infty = x_u - x_v = \mathrm{R}_{\mathrm{eff}}(u, v)$, and thus with high probability, the output actually achieves a multiplicative guarantee $\hat{\mathrm{R}}_{\mathrm{eff}}(u, v) \in (1 \pm \epsilon)\,\mathrm{R}_{\mathrm{eff}}(u, v)$.

1.2 Technical Outline

Algorithms. Our basic technique relies on a classic idea of von Neumann and Ulam [18, 38] for estimating a matrix inverse by a power series; see Section 1.3 for a discussion of related work. Our starting point is the identity

$$\forall X \in \mathbb{R}^{n\times n}, \|X\| < 1, \qquad (I - X)^{-1} = \sum_{t=0}^{\infty} X^t.$$

(Recall that $||X||$ denotes the spectral norm of a matrix X.) Now given a Laplacian $L = L_G$ of a d-regular graph G, observe that $\frac{1}{d}L = I - \frac{1}{d}A$, where A is the adjacency matrix of G. Assume for a moment that $||\frac{1}{d}A|| < 1$; then by the above identity, $(\frac{1}{d}L)^{-1} = (I - \frac{1}{d}A)^{-1} = \sum_{t=0}^{\infty}(\frac{1}{d}A)^t$, and the solution of the linear system $Lx = b$ would be $x^* = L^{-1}b = \frac{1}{d}\sum_{t=0}^{\infty}(\frac{1}{d}A)^t b$. The point is that the summands decay exponentially because $||(\frac{1}{d}A)^t b||_2 \leq ||(\frac{1}{d}A)^t|| \cdot ||b||_2 \leq ||(\frac{1}{d}A)||^t \cdot ||b||_2$. Therefore, we can estimate x^*_u using the first t_0 terms, i.e., $\hat{x}_u = e_u^{\mathsf{T}}\frac{1}{d}\sum_{t=0}^{t_0}(\frac{1}{d}A)^t b$, where t_0 is logarithmic (with base $\|\frac{1}{d}A\|^{-1} > 1$). In order to compute each term $e_u^{\mathsf{T}}\frac{1}{d}(\frac{1}{d}A)^t b$, observe that $e_u^{\mathsf{T}}(\frac{1}{d}A)^t e_w$ is exactly the probability that a random walk of length t starting at u will end at vertex w. Thus, if we perform a random walk of length t starting at u, and let z be its (random) end vertex, then

$$\mathop{\mathbb{E}}_{z}[b_z] = \sum_{w \in V} e_u^{\mathsf{T}}(\tfrac{1}{d}A)^t e_w b_w = e_u^{\mathsf{T}}(\tfrac{1}{d}A)^t b.$$

If we perform several random walks (specifically, $\mathrm{poly}(t_0, \frac{1}{\epsilon})$ walk suffice), average the resulting b_z's, and then multiply by $\frac{1}{d}$, then with high probability, we will obtain a good approximation to $e_u^{\mathsf{T}}\frac{1}{d}(\frac{1}{d}A)^t b$.

As a matter of fact, we have a non-strict inequality $||\frac{1}{d}A|| \leq 1$, because of the all-ones vector $\vec{1} \in \mathbb{R}^n$. Nevertheless, we can still get a meaningful result if all eigenvalues of A except for the largest one are smaller than d (equivalently, the graph G is connected). First, we get rid of any negative eigenvalues by the standard trick of considering $(dI + A)/2$ instead of A, which is equivalent to adding d self-loops at every vertex. Second, we may assume b is orthogonal to $\vec{1}$ (otherwise the linear system has no solution), and while the linear system $Lx = b$ has infinitely many solutions, we estimate the specific solution $x^* \stackrel{\text{def}}{=} L^+ b$ (recall L is PSD) by $\frac{1}{d}\sum_{t=0}^{t_0}(\frac{1}{d}A)^t b$. Indeed, the idealized analysis above still applies by restricting all our calculations to the subspace orthogonal to $\vec{1}$. This is carried out in Theorem 1.1.

To generalize the above approach to SDD matrices, we face three issues. First, due to the irregularity of general SDD matrices, it is harder to properly define the equivalent random walk matrix. We resolve this by normalizing the SDD matrix S into $\tilde{S}$ defined in (2), and solving the equivalent (normalized) system $\tilde{S}(D^{1/2}x) = D^{-1/2}b$. Second, general SDD matrices can have *positive* off-diagonal elements, in constrast to Laplacians. To address this, we interpret such entries as negative-weight edges, and employ random walks that "remember" the signs of the traversed edges. Third, diagonal elements may strictly dominate their row, which we address by terminating the random walk early with some positive probability.

Lower Bound: Polynomial Time for PSD Matrices. We first discuss our lower bound for PSD matrices, which is one of the main contributions of our work. It exhibits a family of matrices S for which estimating a coordinate x^*_u of the solution $x^* = S^{-1}b$ requires $n^{\Omega(1)}$ probes into the input b.

Without the sparsity constraint on S, one can deduce such a lower bound via a reduction from the communication complexity of the *Vector in Subspace Problem* (VSP), in which Alice has an $n/2$-dimensional subspace $H \subset \mathbb{R}^n$, Bob has a vector $b \in \mathbb{R}^n$, and their goal is to determine whether $b \in H$ or $b \in H^{\perp}$. The randomized communication complexity of this promise problem is between $\Omega(n^{1/3})$ [23] and $O(\sqrt{n})$ [28] (while for quantum communication it is $O(\log n)$). To reduce this problem to linear-system solvers, let $P_H \in R^{n \times n}$ be the projection operator onto the subspace H, and set $S = I + P_H$. Consider the system $Sx = b$, and notice that Alice knows S and Bob knows b. It is easy to see that the unique solution x^* is either b or $\frac{1}{2}b$, depending on whether $b \in H^{\perp}$ or $b \in H$. Alice and Bob could use a solver that makes few probes to b, as follows. Bob would pick an index $u \in [n]$ that maximizes

$|b_u|$ (and thus also $|x_u|$), and send it to Alice. She would then apply the solver, asking Bob for only a few entries of b, to estimate x_u within additive error $\frac{1}{2}\|x\|_\infty$, which suffices to distinguish the two cases. This matrix S is PSD with condition number $\kappa(S) \leq 2$. However, it is dense.

We thus revert to a different approach of proving it from basic principles. Our high-level idea is to take a $2d$-regular expander and assign to its edges random signs (± 1) that are balanced everywhere, namely, at every vertex the incident edges are split evenly between positive and negative. The signed adjacency matrix $A \in \{-1, 0, +1\}^{n \times n}$ should have spectral norm $\mu \stackrel{\text{def}}{=} \|A\| = O(\sqrt{d})$, and then instead of the (signed) Laplacian $L = (2d)I - A$, we consider $S = 2\mu I - A$, which is PSD with condition number $\kappa(S) \leq 3$, as well as invertible and sparse. Now following arguments similar to our algorithm, we can write S^{-1} as a power series of the matrix A, and express coordinate x^*_u of the solution $x^* = S^{-1}b$ via $\mathbb{E}_z[b_z]$ where z is the (random) end vertex of a random walk that starts at u and its length is bounded by some t_0 (performed in the "signed" graph corresponding to A). Now if the graph around u looks like a tree (e.g., it has high girth), then not-too-long walks are highly symmetric and easy to count. We now let b_v be non-zero only at vertices v at distance exactly t_0 from u, and for these vertices set $b_v \in \{+1, -1\}$ at random but with a small bias δ towards one of the values. Some calculations show that $\operatorname{sgn}(\mathbb{E}_z[b_z])$, and consequently $\operatorname{sgn}(x^*_u)$, will be according to our bias (with high probability), however discovering this $\operatorname{sgn}(x^*_u)$ via probes to b is essentially the problem of learning a biased coin, which requires $\Omega(\delta^{-2})$ coin observations. An additional technical obstacle is to bound $\|x^*\|_\infty$, so that we can argue that an $\frac{1}{5}\|x^*\|_\infty$-additive error to x^*_u will not change its sign. Overall, we show we can set $t_0 = \Omega(\log_d n)$ and $\delta \approx ((2d-1)^{t_0})^{-1/2}$, thus concluding that the algorithm must observe $\Omega(\delta^{-2}) = n^{\Omega(1)}$ entries of b.

It is instructive to ask where in the above argument is it crucial to have $\mu = O(\sqrt{d})$, because if it were valid also for $\mu = d$, then it would hold also for the SDD matrix $S = 2\mu I - A$, and contradict our own algorithm for SDD matrices. The answer is that $\mu \ll 2d$ is required to bound $\|x^*\|_\infty$ in Lemma 4.8.

Lower Bound: Quadratic Dependence on Condition Number. We now outline the ideas to prove the $\tilde{\Omega}(\kappa^2)$ lower bound even for Laplacian systems with condition number κ. First, it is relatively straightforward to prove that a *linear* dependence on the condition number is necessary. Indeed, consider a dumbbell graph, namely, two 3-regular expanders connected by a bridge edge (u, v), and suppose one need to estimate $x^*_u - x^*_v$. For input $b = e_i - e_j$, the value of $x^*_u - x^*_v$ is non-zero iff vertices i, j are on opposite sides of the bridge, and determining the latter requires $\Omega(n)$ probes into b. Since this graph has condition number $O(n)$, we obtain an $\Omega(\kappa)$ lower bound.

The quadratic lower bound requires both a different graph and a different vector b. We use the following graph G with condition number $O(k)$: take two 3-regular expanders and connect them with n/k "bridge edges". The vector $b \in \{-1, +1\}^n$ is *dense* and in particular it is either: 1) balanced, i.e., $\sum b_i$ on each expander is zero; or 2) unbalanced, i.e., each $b_i \in \{+1, -1\}$ at random with a bias $p \approx 1/k$ towards $+1$ on the first expander, and towards -1 on the second one. Now, as above, it is simple to prove that: 1) in the balanced case, the average of $x^*_u - x^*_v$ *over all bridge edges* (u, v) must be zero; and 2) in the unbalanced case, the same average must be $\Omega(1)$. The main challenge is that the actual values might differ from the average – e.g., even in the balanced case, each bridge edge (u, v) will likely have non-zero value of $x^*_u - x^*_v$. Nonetheless, we manage to prove an upper bound on the maximum value of $|x^*_u - x^*_v|$ over all edges (u, v) (as in the previous lower bound, we need to bound $\|x^*\|_\infty$ as well). For the latter, we need to again analyze $\mathbb{E}_z[b_z]$ where z is the end vertex of a random walk of some fixed length $i \geq 1$ starting from u in the graph G. Since

the vector b is not symmetric over the graph G, a direct analysis seems hard – instead we estimate $\mathbb{E}_z[b_z]$ via a coupling of such walks in G with random walks in an expander, which is amenable to a direct analysis.

1.3 Related Work

The idea of approximating the inverse $(I-X)^{-1} = \sum_{t=0}^{\infty} X^t$ (for $||X|| < 1$) by random walks dates back to von Neumann and Ulam [18, 38]. While we approximate each power X^t by separate random walks of length t and truncate the tail (powers above some t_0), their method employs random walks whose length is random and whose expectation gives exactly the infinite sum, achieved by assigning some probability to terminate the walk at each step, and weighting the contributions of the walks accordingly (to correct the expectation).

The idea of approximating a generalized inverse L^* of $L = dI - A$ by the truncated series $\frac{1}{d}\sum_{t=0}^{t_0}(\frac{1}{d}A)^t$ on directions that are orthogonal to the all-ones vector was recently used by Doron, Le Gall, and Ta-Shma [16] to show that L^* can be approximated in probabilistic log-space. However, since they wanted to output L^* explicitly, they could not ignore the all-ones direction and they needed to relate L^* to $\frac{1}{d}\sum_{t=0}^{\infty}(\frac{1}{d}A)^t$ by "peeling off" the all-ones direction, inverting using the infinite sum formula, and then adding back the all-ones direction.

The idea of estimating powers of a normalized adjacency matrix $\frac{1}{d}A$ (or more generally, a stochastic matrix) by performing random walks is well known, and was used also in [16] mentioned above, and in [17]. Chung and Simpson [12] used it in a context that is related to ours, of solving a Laplacian system $L_G x = b$, but with a boundary condition, namely, a constraint that $x_i = b_i$ for all i in the support of b. Their algorithm solves for a subset of the coordinates $W \subseteq V$, i.e., it approximates $x|_W$ (the restriction of x to coordinates in W) where x solves $Lx = b$ under the boundary condition. They relate the solution x to the Dirichlet heat-kernel PageRank vector, which in turn is related to an infinite power series of a transition matrix (specifically, to $f^{\mathsf{T}} e^{-t(I-P_W)} = e^{-t} f^{\mathsf{T}} \sum_{k=0}^{\infty} \frac{t^k}{k!} P_W^k$ where P_W is the transition matrix of the graph induced by W, $t \in \mathbb{R}$, and $f \in \mathbb{R}^{|W|}$), and their algorithm uses random walks to approximate the not-too-large powers of the transition matrix, proving that the remainder of the infinite sum is small enough.

Recently, Shyamkumar, Banerjee and Lofgren [30] considered a related matrix-power problem, where the input is a matrix $A \in \mathbb{R}^{n\times n}$, a power $\ell \in \mathbb{N}$, a vector $z \in \mathbb{R}^n$, and an index $u \in [n]$, and the goal is to compute coordinate u of $A^\ell z$. They devised for this problem a sublinear (in $\mathrm{nnz}(A)$) algorithm, under some bounded-norm conditions and assuming $u \in [n]$ is uniformly random. Their algorithm relies, in part, on von Neumann and Ulam's technique of computing matrix powers using random walks, but of prescribed length. It can be shown that approximately solving positive definite systems for a particular coordinate is reducible to the matrix-power problem.[7] However, in contrast to our results, their expected running time is polynomial in the input size, namely $\mathrm{nnz}(A)^{2/3}$, and holds only for a random $u \in [n]$.

Comparison with PageRank. An example application of our results is computing quickly the PageRank (defined in [8]) of a single node in an undirected d-regular graph. Recall that the PageRank vector of an n-vertex graph with associated transition matrix P is the solution

[7] Let $Ax = b$ be a linear system where A is positive definite. Let λ be the largest eigenvalue of A. Let $A' \stackrel{\text{def}}{=} \frac{1}{2\lambda}A$ and $b' \stackrel{\text{def}}{=} \frac{1}{2\lambda}b$. Consider the equivalent system $(I-(I-A'))x = b'$. As the eigenvalues of A' are in $(0, 1/2]$, the eigenvalues of $I - A'$ are in $[1/2, 1)$. Thus, the solution to the linear system is given by $x = (I-(I-A'))^{-1}b' = \sum_{t=0}^{\infty}(I-A')^t b$. Therefore, we can approximate x_u by truncating the infinite sum at some t_0 and approximating each power $t < t_0$ by the algorithm for the matrix-power problem.

to the linear system $x = \frac{1-\alpha}{n}\vec{1} + \alpha P x$, where $0 < \alpha < 1$ is a given parameter. In personalized PageRank, one replaces $\frac{1}{n}\vec{1}$ (the uniform distribution) with some $b \in \mathbb{R}^n$, e.g., a standard basis vector. Equivalently, x solves the system $Sx = \frac{1-\alpha}{n}\vec{1}$ where $S = I - \alpha P$ is an SDD matrix with 1's on the diagonal. As all eigenvalues of P are of magnitude at most 1 (recall P is a transition matrix), all eigenvalues of $I - \tilde{S} = I - S = \alpha P$ are of magnitude at most α, and the running time guaranteed by Theorem 3.1 is logarithmic (with base $\frac{2}{\alpha+1}$).

Algorithms for the PageRank model were studied extensively, and usually consider arbitrary (and even directed) graphs. In particular, the sublinear algorithms of [7] approximate the PageRank of a vertex using $\tilde{O}(n^{2/3})$ queries, or using $\tilde{O}((n\Delta)^{1/2})$ queries when the maximum degree is Δ. Another example is the heavy-hitters algorithm of [6], which reports all vertices whose approximate PageRank exceeds a threshold T in sublinear time $\tilde{O}(1/\Delta)$, when PageRanks are viewed as probabilities and sum to 1. Other work explores connections to other graph problems, including for instance using PageRank algorithms to approximate effective resistances [13], the PageRank vector itself, and computing sparse cuts [4].

Local Algorithms. Our algorithms in Theorems 1.1 and 3.1 are *local* in the sense that they query a small portion of their input, usually around the input vertex, when viewed as graph algorithms. Local algorithms for graph problems were studied in several contexts, like graph partitioning [31, 5], Web analysis [10, 3], and distributed computing [34]. Rubinfeld, Tamir, Vardi, and Xie [29] introduced a formal concept of *Local Computation Algorithms* that requires consistency between the local outputs of multiple executions (namely, these local outputs must all agree with a single global solution). As explained earlier, our problem formulation (1) follows this consistency requirement.

1.4 Future Work

One may study alternative ways of defining the problem of solving a linear system in sublinear time, in particular if the algorithm can access b in a different way. For example, similarly to assumptions and guarantees in [35], the goal may be to produce an ℓ_2-sample from the solution x (i.e., report a random index in $[n]$ such that the probability of each coordinate $i \in [n]$ is proportional to x_i^2) assuming oracle access to an ℓ_2-sampler from $b \in \mathbb{R}^n$, i.e., use an ℓ_2-sampler for b to construct an ℓ_2-sampler for x. Another version of the problem may ask to produce heavy hitters in x, assuming, say,[8] heavy hitters in b (which may be useful for the PageRank application). We leave these extensions as interesting open questions, focusing here on the classical access mode to b, via queries to its coordinates.

Another variation one may consider is to bound the error using a norm other than ℓ_∞, like $\|y\|_S \stackrel{\text{def}}{=} \sqrt{y^\mathsf{T} S y}$ used in [31]. For example, if S is the Laplacian of a d-regular expander and y is orthogonal to the all-ones vector, then $\|y\|_S = \Theta(\sqrt{d}\|y\|_2)$, which might exceed $\|y\|_\infty$ significantly even for constant d. Nevertheless, our requirement $\|\hat{x} - x^*\|_\infty \le \epsilon ||x^*||_\infty$ is generally incomparable to $\|\hat{x} - x^*\|_2 \le \epsilon ||x^*||_2$.

2 Laplacian Solver (for Regular Graphs)

In this section we shall prove Theorem 1.1. The ensuing description deals mostly with a slightly simplified scenario, where the algorithm is given not one but two vertices $u, v \in [n]$, and returns an approximation $\hat{\delta}_{u,v}$ to $x_u - x_v$ with a slightly different error bound, see

[8] This kind of oracle seems necessary even when $S = I$.

Algorithm 1 `Solve-Linear-Laplacian`.

input : d-regular graph G; vector b; $||b||_0$; vertices u, v; accuracy parameter ϵ; and μ_2
output : estimate $\hat{\delta}_{u,v}$ for $x_u - x_v$

set $s = \frac{\log(2\sqrt{2}\epsilon^{-1}\frac{d}{\mu_2}\sqrt{||b||_0})}{\log(\frac{d}{d-\mu_2})}$ and $\ell = O((\frac{\epsilon}{4s})^{-2}\log s)$
for $t = 0, 1, \ldots, s-1$ **do**
  Perform ℓ independent random walks of length t starting at u, and let $u_1^{(t)}, \ldots, u_\ell^{(t)}$ be their end vertices. Independently, perform ℓ independent random walks of length t starting at v, and let $v_1^{(t)}, \ldots, v_\ell^{(t)}$ be their end vertices.
  set $\hat{\delta}_{u,v}^{(t)} = \frac{1}{\ell}\sum_{i\in[\ell]}(b_{u_i^{(t)}} - b_{v_i^{(t)}})$
return $\hat{\delta}_{u,v} = \frac{1}{d}\sum_{t=0}^{s-1}\hat{\delta}_{u,v}^{(t)}$

Theorem 2.5 for the precise statement. The advantage is that if G is connected, all solutions x give rise to a unique value for $x_u - x_v$. We will then explain the modifications required to prove Theorem 1.1 (which actually follows also from our more general Theorem 3.1).

Let $G = (V = [n], E)$ be a connected d-regular graph with adjacency matrix $A \in \mathbb{R}^{n\times n}$. Let the eigenvalues of A be $d = \lambda_1 > \lambda_2 \geq \cdots \geq \lambda_n$, and let their associated orthonormal eigenvectors be $u_1, \ldots, u_n$. Then $u_1 = \frac{1}{\sqrt{n}}\cdot\vec{1} \in \mathbb{R}^n$, and we can write $A = U\Lambda U^\mathsf{T}$ where $U = [u_1\, u_2 \ldots u_n]$ is unitary and $\Lambda = \mathrm{diag}(\lambda_1, ..., \lambda_n)$. For $u, v \in [n]$, let $\chi_{u,v} \stackrel{\text{def}}{=} e_u - e_v$ where e_i is the i-th standard basis vector. Then the Laplacian of G is given by

$$L \stackrel{\text{def}}{=} \sum_{uv\in E}\chi_{u,v}\chi_{u,v}^\mathsf{T} = dI - A = U(dI - \Lambda)U^\mathsf{T}.$$

Observe that L does not depend on the orientation of each edge uv, and that $\mu_2 \stackrel{\text{def}}{=} d - \lambda_2$ is the smallest non-zero eigenvalue of L. The Moore-Penrose pseudo-inverse of L is

$$L^+ \stackrel{\text{def}}{=} U\cdot\mathrm{diag}(0, (d-\lambda_2)^{-1}, \ldots, (d-\lambda_n)^{-1})\cdot U^\mathsf{T}.$$

We assume henceforth that all eigenvalues of A are non-negative. At the end of the proof, we will remove this assumption (by adding self-loops).

The idea behind the next fact is that $L = d(I - \frac{1}{d}A)$, and $\frac{1}{d}A$ has norm strictly smaller than one when operating on the subspace that is orthogonal to the all-ones vector, and hence, the formula $(I - X)^{-1} = \sum_{t=0}^\infty X^t$ for $||X|| < 1$ is applicable for the span of $\{u_2, ..., u_n\}$.

▶ **Fact 2.1.** *For every $x \in \mathbb{R}^n$ that is orthogonal to the all-ones vector, $L^+x = \frac{1}{d}\sum_{t=0}^\infty(\frac{1}{d}A)^t x$.*

Proof. It suffices to prove the claim for each of $u_2, \ldots, u_n$ as the fact will then follow by linearity. Fix $i \in \{2, \ldots, n\}$. Then since $|\frac{\lambda_i}{d}| < 1$,

$$\sum_{t=0}^\infty\left(\frac{1}{d}A\right)^t u_i = \sum_{t=0}^\infty\left(\frac{\lambda_i}{d}\right)^t u_i = \frac{1}{1-\frac{\lambda_i}{d}}u_i = \frac{d}{d-\lambda_i}u_i = dL^+u_i.$$ ◀

We now describe an algorithm that on input $b \in \mathbb{R}^n$ that is orthogonal to the all-ones vector, and two vertices $u \neq v \in [n]$, returns an approximation $\hat{\delta}_{u,v}$ to $x_u - x_v$, where x solves $Lx = b$. As G is connected, the null space of L is equal to $\mathrm{span}\{\vec{1}\}$ and hence $x_u - x_v$ is uniquely defined, and can be written as $x_u - x_v = \chi_{u,v}^\mathsf{T}L^+b$.

▶ **Claim 2.2.** *For b that is orthogonal to the all-ones vector and $s = \frac{\log(2\sqrt{2}\epsilon^{-1}\frac{d}{\mu_2}\sqrt{||b||_0})}{\log(\frac{d}{d-\mu_2})}$, $|\chi_{u,v}^{\mathsf{T}} L^{+} b - \chi_{u,v}^{\mathsf{T}} \frac{1}{d}\sum_{t=0}^{s-1}(\frac{1}{d}A)^t b| \leq \frac{\epsilon}{2d}||b||_\infty$.*

Proof. Using Fact 2.1,

$$\chi_{u,v}^{\mathsf{T}} L^{+} b - \chi_{u,v}^{\mathsf{T}} \frac{1}{d}\sum_{t=0}^{s-1}(\frac{1}{d}A)^t b = \chi_{u,v}^{\mathsf{T}} \frac{1}{d}\sum_{t=s}^{\infty}(\frac{1}{d}A)^t b,$$

and thus

$$|\chi_{u,v}^{\mathsf{T}} L^{+} b - \chi_{u,v}^{\mathsf{T}} \frac{1}{d}\sum_{t=0}^{s-1}(\frac{1}{d}A)^t b| \leq ||\chi_{u,v}^{\mathsf{T}}||_2 \cdot ||\frac{1}{d}\sum_{t=s}^{\infty}(\frac{1}{d}A)^t b||_2.$$

We know that $||\chi_{u,v}^{\mathsf{T}}||_2 = \sqrt{2}$, so it remains to bound $||\frac{1}{d}\sum_{t=s}^{\infty}(\frac{1}{d}A)^t b||_2$. Decomposing $b = \sum_{i=2}^{n} c_i u_i$ we get that $\sum_{i=2}^{n} c_i^2 = ||b||_2^2$ and

$$\sum_{t=s}^{\infty}(\frac{1}{d}A)^t b = \sum_{i=2}^{n} c_i u_i \sum_{t=s}^{\infty}(\frac{\lambda_i}{d})^t = \sum_{i=2}^{n} \frac{(\frac{\lambda_i}{d})^s}{1-\frac{\lambda_i}{d}} c_i u_i = d\sum_{i=2}^{n} \frac{(\frac{\lambda_i}{d})^s}{d-\lambda_i} c_i u_i.$$

Hence,

$$||\frac{1}{d}\sum_{t=s}^{\infty}(\frac{1}{d}A)^t b||_2^2 = \sum_{i=2}^{n}\left(\frac{(\frac{\lambda_i}{d})^s}{d-\lambda_i}\right)^2 c_i^2||u_i||_2^2 \leq \left(\frac{(\frac{\lambda_2}{d})^s}{d-\lambda_2}\right)^2 \sum_{i=2}^{n} c_i^2 = \left(\frac{(1-\frac{\mu_2}{d})^s}{\mu_2}\right)^2 ||b||_2^2,$$

where the first equality is because the u_i's are orthogonal. Altogether,

$$|\chi_{u,v}^{\mathsf{T}} L^{+} b - \chi_{u,v}^{\mathsf{T}} \frac{1}{d}\sum_{t=0}^{s-1}(\frac{1}{d}A)^t b| \leq \sqrt{2}\frac{(1-\frac{\mu_2}{d})^s}{\mu_2}||b||_2 \leq \sqrt{2}\frac{(1-\frac{\mu_2}{d})^s}{\mu_2}\sqrt{||b||_0}\cdot||b||_\infty = \frac{\epsilon}{2d}||b||_\infty,$$

as claimed. ◀

▶ **Claim 2.3.** $\Pr\left[|\hat{\delta}_{u,v} - \chi_{u,v}^{\mathsf{T}} \frac{1}{d}\sum_{t=0}^{s-1}(\frac{1}{d}A)^t b| > \frac{\epsilon}{2d}||b||_\infty\right] \leq \frac{1}{s}$.

Proof. Observe that $e_u^{\mathsf{T}}(\frac{1}{d}A)^t$ is a probability vector over V, and $e_u^{\mathsf{T}}(\frac{1}{d}A)^t e_w$ is exactly the probability that a random walk of length t starting at u will end at w. Thus, for every $t \in \{0,1,\ldots,s-1\}$ and $i \in [\ell]$, we have

$$\mathbb{E}[b_{u_i^{(t)}}] = \sum_{w\in[n]} e_u^{\mathsf{T}}(\frac{1}{d}A)^t e_w b_w = e_u^{\mathsf{T}}(\frac{1}{d}A)^t b,$$

and similarly $\mathbb{E}[b_{v_i^{(t)}}] = e_v^{\mathsf{T}}(\frac{1}{d}A)^t b$. By a union bound over Hoeffding bounds, with probability at least $1-\frac{1}{s}$, for every $t \in \{0,1,\ldots,s-1\}$, we have $|\frac{1}{\ell}\sum_{i\in[\ell]} b_{u_i^{(t)}} - e_u^{\mathsf{T}}(\frac{1}{d}A)^t b| \leq \frac{\epsilon}{4s}||b||_\infty$ and $|\frac{1}{\ell}\sum_{i\in[\ell]} b_{v_i^{(t)}} - e_v^{\mathsf{T}}(\frac{1}{d}A)^t b| \leq \frac{\epsilon}{4s}||b||_\infty$. Recalling that $\hat{\delta}_{u,v} = \frac{1}{d}\sum_{t=0}^{s-1}\frac{1}{\ell}\sum_{i\in[\ell]}(b_{u_i^{(t)}} - b_{v_i^{(t)}})$, with probability at least $1-\frac{1}{s}$ we have $|\hat{\delta}_{u,v} - \chi_{u,v}^{\mathsf{T}}\frac{1}{d}\sum_{t=0}^{s-1}(\frac{1}{d}A)^t b| \leq \frac{\epsilon}{2d}||b||_\infty$, as claimed. ◀

Combining Claim 2.2 and Claim 2.3 we get that (with probability $1-\frac{1}{s}$) $|\hat{\delta}_{u,v} - \chi_{u,v}^{\mathsf{T}} L^{+} b| \leq \frac{\epsilon}{d}||b||_\infty$. Now, as x solves $Lx = b$, for every $i \in [n]$ we have $\sum_{j\in N(i)}(x_i - x_j) = b_i$ where $N(i)$ is the set of neighbors $\{j : ij \in E\}$, which implies that for some neighbor j of i, it holds that $|x_i - x_j| \geq \frac{|b_i|}{d}$. Therefore, $\max_{ij\in E}|x_i - x_j| \geq \frac{1}{d}||b||_\infty$. We conclude that $|\hat{\delta}_{u,v} - \chi_{u,v} L^{+} b| \leq \epsilon \cdot \max_{ij\in E}|x_i - x_j|$. We now turn to the running time of Algorithm 1,

which is dominated by the time it takes to perform the random walks. There are $2s \cdot \ell$ random walks in total. The random walks do not need to be independent for different values of t (as we applied a union bound over the different t), we can extend, at each iteration t, the 2ℓ respective random walks constructed at iteration $t-1$ by an extra step in time $O(d)$ (recall we assume G is given as an adjacency list), obtaining a total runtime $O(s \cdot \ell \cdot d) = O(d\epsilon^{-2}s^3 \log s)$. To simplify the expression for s, we need the following bound.

▶ **Fact 2.4.** *For all $\delta \in (0,1)$, $\frac{1}{\ln(1-\delta)^{-1}} \leq \frac{1}{\delta}$.*

Proof. We need to show that $\delta \leq \ln(1-\delta)^{-1}$, or equivalently, $e^{-\delta} \geq 1-\delta$, which is well known. ◀

Applying Fact 2.4 to $\delta = \frac{\mu_2}{d}$, we have $s \leq \frac{d}{\mu_2}\log(2\sqrt{2}\epsilon^{-1}\frac{d}{\mu_2}\sqrt{||b||_0})$, and conclude the following.

▶ **Theorem 2.5.** *Given an adjacency list of a connected d-regular n-vertex graph G, a vector $b \in \mathbb{R}^n$ that is orthogonal to the all-ones vector, vertices $u, v \in [n]$, and scalars $||b||_0$, $\epsilon > 0$, and $\mu_2 = d - \lambda_2 > 0$, Algorithm 1 outputs $\hat{\delta}_{u,v} \in \mathbb{R}$ satisfying*

$$\Pr\left[|\hat{\delta}_{u,v} - \chi_{u,v}^{\mathsf{T}} L^{+} b| \leq \epsilon \cdot \max_{ij \in E} |x_i - x_j|\right] \geq 1 - \tfrac{1}{s},$$

in time $O(d\epsilon^{-2}s^3 \log s)$ for $s = O(\frac{d}{\mu_2}\log(\epsilon^{-1}\frac{d}{\mu_2}||b||_0))$.

▶ Remark. If we allow preprocessing of G, the runtime of Algorithm 1 can be reduced to $O(\epsilon^{-2}s^2)$, as follows. At the preprocessing phase, compute $(\frac{1}{d}A)^t$ for all powers $t \leq s$. Then, instead of approximating $e_u^{\mathsf{T}}(\frac{1}{d}A)^t b$ for all powers $t \leq s$, sample a uniform $t \in \{0, 1, ..., s\}$, and then, in $O(1)$ time (because the probability vector is precomputed, see [37]), sample $z \in [n]$ based on the probability vector $e_u^{\mathsf{T}}(\frac{1}{d}A)^t$, and finally, output $\frac{s+1}{d}b_z$. The expectation of the output is $\frac{1}{d}\sum_{t=0}^{s}(\frac{1}{d}A)^t b$. As for concentration, since the output is in $[-\frac{s+1}{d}\cdot||b||_\infty, \frac{s+1}{d}\cdot||b||_\infty]$, by the Hoeffding bound, $O(\epsilon^{-2}s^2)$ many repetitions suffice to obtain (with constant probability) an approximation with additive error $\frac{\epsilon}{2d}||b||_\infty$ (as in Claim 2.3).

We still need to show how to remove the assumption that A has no negative eigenvalues. Given an adjacency matrix A which might have negative eigenvalues, consider the PSD matrix $A' = A + dI$, which is the adjacency matrix of the $2d$-regular graph G' obtained from G by adding d self-loops to each vertex. Observe that $A' = U(\Lambda + dI)U^{\mathsf{T}}$ and we can write $L = dI - A = (2dI - A')$, and thus, similarly to Fact 2.1, $L^{+}x = \frac{1}{2d}\sum_{t=0}^{\infty}(\frac{1}{2d}A')^t$, for $x \in \mathbb{R}^n$ that is orthogonal to the all-ones vector. Therefore, if we use A' (which is PSD) to guide Algorithm 1's random walks (i.e., at each step of a walk, with probability 1/2 the walk stays put and with probability 1/2 it moves to a uniform neighbor in G) and apply Claims 2.2 and 2.3 (which apply even when A has self-loops), an estimate $\hat{\delta}_{u,v}$ satisfying with high probability $|\frac{1}{2}\hat{\delta}_{u,v} - \chi_{u,v}^{\mathsf{T}} L^{+} b| \leq \epsilon \max_{ij \in E}|x_i - x_j|$ is obtained. When running Algorithm 1 on G', the term s evaluates to $O(\frac{2d}{2d-(\lambda_2+d)}\log(\epsilon^{-1}\frac{2d}{2d-(\lambda_2+d)}||b||_0)) = O(\frac{d}{\mu_2}\log(\epsilon^{-1}\frac{d}{\mu_2}||b||_0))$, thus, leaving the guarantee of Theorem 2.5 intact (up to constant factors).

Proof of Theorem 1.1. The theorem follows by a simple modifications to the analysis above. Observe that the analysis in Claims 2.2 and 2.3 holds also when replacing μ_2 by a lower bound on μ_2, which in turn is easy to derive from the upper bound $\bar{\kappa}$ given in the input and d given as part of input G. Similarly, $||b||_0$ can be replaced by an upper bound $B_{up} \geq ||b||_0$.

To handle one vertex $u \in [n]$ instead of two vertices $u, v \in [n]$, ignore the part dealing with v in Algorithm 1, and modify the analysis in the two aforementioned claims to use e_u instead of $\chi_{u,v}$. The error bound obtained from combining these lemmas is $\frac{\epsilon}{d}||b||_\infty$, but since each $|b_i| = |\sum_j L_{ij}x_j| \leq \sum_j |L_{ij}| \cdot \|x\|_\infty = 2d\|x\|_\infty$, we can bound the error by $\frac{\epsilon}{d}||b||_\infty \leq 2\epsilon\|x\|_\infty$. ◀

3 An SDD Solver

In this section we prove the following theorem for solving linear systems in SDD matrices. To generalize from Laplacianss of regular graphs to SDD matrices, we face several issues as described in Section 1.2. We use the notation defined in (2)-(3).

▶ **Theorem 3.1** (SDD Solver). *There exists a randomized algorithm, that given input* $\langle S, b, u, \epsilon, \tilde{\lambda}_{up} \rangle$*, where*

- $S \in \mathbb{R}^{n \times n}$ *is an SDD matrix,*
- $b \in \mathbb{R}^n$ *is in the range of* S *(equivalently, orthogonal to the kernel of* S*),*
- $u \in [n]$*,* $\epsilon > 0$*, and*
- $\bar{\kappa} \geq 1$ *is an upper bound on the condition number* $\kappa(\tilde{S})$*,*

this algorithm outputs $\hat{x}_u \in \mathbb{R}$ *with the following guarantee. Suppose* x^* *is the solution for* $Sx = b$ *given in* (3)*, then*

$$\forall u \in [n], \qquad \Pr\left[|\hat{x}_u - x^*_u| \leq \epsilon ||x^*||_\infty\right] \geq 1 - \frac{1}{s}$$

for suitable $s = O(\bar{\kappa} \log(\epsilon^{-1} \bar{\kappa} \, ||b||_0 \cdot \frac{\max_{i \in [n]} D_{ii}}{\min_{i \in [n]} D_{ii}}))$*. The algorithm runs in time* $O(f \epsilon^{-2} s^3 \log s)$*, where* f *is the time to make a step in a random walk in the weighted graph formed by the non-zeros of* S*.*

Due to space constraints, the proof is omitted from this version.

4 Lower Bound for PSD Matrices

In this section we prove Theorem 1.2. The entire proof relies on a d-regular n-vertex graph G_1, such that (i) its girth is $\Omega(\log_d n)$; and (ii) its adjacency matrix A_1 has eigenvalues $\lambda_1 \geq \ldots \geq \lambda_n$ that satisfy $\max\{|\lambda_2|, |\lambda_n|\} \leq \frac{1}{4} d^{2/3}$ (this bound is somewhat arbitrary, chosen to simplify the exposition). We actually need such a graph to exist for infinitely many n, with d bounded uniformly (as n grows). Such graphs are indeed known, for example the Ramanujan graphs constructed by Lubotzky, Philips and Sarnak [26] and by Margulis [27] for the case where $d-1$ is a prime, have eigenvalue upper bound $2\sqrt{d-1}$ and girth lower bound $(4/3 - o(1)) \log_{d-1} n$ (see e.g. [21]).

In what follows, let G_2 be a certain isomorphic copy of G_1 (i.e., obtained from G_1 by permuting the vertices, as explained below). It will be convenient to assume that G_1 and G_2 have the *same* vertex set, which we denote by V, as then we can consider the multi-graph obtained by their edge union, denoted $G_1 \cup G_2$. Denoting the adjacency matrix of each G_i by A_i, the adjacency matrix of their edge union $G_1 \cup G_2$ is simply $A_1 + A_2$. We can similarly view $A_1 - A_2$ as the adjacency matrix of the same graph, except that now the edges are signed – those from G_1 are positive, and those from G_2 are negative.

The proof of the theorem will follow easily from the three propositions below. Proposition 4.1 provides combinatorial, girth-like, information about $G_1 \cup G_2$. Proposition 4.2 provides spectral information, like the condition number, about $A_1 - A_2$. These two propositions are proved by straightforward arguments, and the heart of the argument is in Proposition 4.3. This proposition constructs a PSD linear system based on $A_1 - A_2$, for which we can sow show that recovering a specific coordinate of the solution x, even approximately, requires many probes to b. Due to space constraints, the proof of the next proposition is omitted from this version.

▶ **Proposition 4.1.** *Let G_1 be as above and fix a vertex $\hat{w} \in V$. Then there exists an isomorphic copy G_2 of G_1 (on the same vertex set), such that in their edge-union $G_1 \cup G_2$, the neighborhood of $\hat{w}$ of radius $r_{\text{tree}} \stackrel{\text{def}}{=} 0.2 \log_{4d} n$ is a $2d$-regular tree.*

▶ **Proposition 4.2.** *Let A_1, A_2 be the adjacency matrices described above, and let $\mu \stackrel{\text{def}}{=} 2\|A_1 - A_2\|$. Then $\mu \leq \frac{1}{2}d^{2/3}$, and the matrix $M \stackrel{\text{def}}{=} \mu I + A_1 - A_2 \in \mathbb{R}^{n \times n}$ is PSD with all its eigenvalues in the range $[\frac{1}{2}\mu, \frac{3}{2}\mu]$. Thus, M is invertible and has condition number $\kappa(M) \leq 3$.*

Proof. By the triangle inequality, $\mu/2 = \|A_1 - A_2\| \leq \|A_1 - dI\| + \|-(A_2 - dI)\| \leq 2\max\{|\lambda_2|, |\lambda_n|\} \leq \frac{1}{2}d^{2/3}$. The eigenvalues of $A_1 - A_2$ are in the range $[-\frac{1}{2}\mu, \frac{1}{2}\mu]$, and thus those of M are in the range $[\frac{1}{2}\mu, \frac{3}{2}\mu]$. ◀

▶ **Proposition 4.3** (Proved in Section 4.1)**.** *Let the graphs G_1, G_2 be according to Proposition 4.1, let $M \stackrel{\text{def}}{=} \mu I + A_1 - A_2 \in \mathbb{R}^{n \times n}$ as above, and fix $r \leq r_{\text{tree}}/d^2$. Then every randomized algorithm that, given input $b \in \{-1, 0, +1\}^n$, succeeds with probability at least $6/7$ to approximate coordinate $x_{\hat{w}}$ of $x = M^{-1}b$ within additive error at most $\frac{1}{5}\|x\|_\infty$, must probe $d^{\Omega(r)}$ entries from $b \in \mathbb{R}^n$, even when b is supported only on vertices at distance r from $\hat{w}$ (in $G_1 \cup G_2$).*

We can now prove Theorem 1.2 using the above 3 propositions. Let G_1,G_2,A_1,A_2 and M be as required for these propositions, and fix $r = r_{\text{tree}}/d^2$. Let $S \stackrel{\text{def}}{=} M$ and observe that it has the sparsity and condition number required for Theorem 1.2, and let the distinguished index be $u \stackrel{\text{def}}{=} \hat{w}$. Now consider a randomized algorithm that, given an input $b \in \mathbb{R}^n$, estimates coordinate x^*_u of $x^* = S^{-1}b$, or in other words, coordinate $x_{\hat{w}}$ of $x = M^{-1}b$. We can then apply Proposition 4.3 and deduce that this algorithm must probe $b \in \mathbb{R}^n$ in

$$d^{\Omega(r)} \geq d^{\Omega((\log_{4d} n)/d^2)} \geq n^{\Omega(1/d^2)}$$

entries, which proves Theorem 1.2.

4.1 Proof of Proposition 4.3

Let $V_k \subset V$ be the set of vertices at distance exactly k from $\hat{w}$ in the edge-union graph $G_1 \cup G_2$. By Proposition 4.1, we can view the radius-r_{tree} neighborhood of $\hat{w}$ as a tree rooted at $\hat{w}$. In particular, for all $k \leq r_{\text{tree}}$ we have $|V_k| = 2d(2d-1)^{k-1}$. For each vertex $v \in V_k$, let $s_v \in \{\pm 1\}$ be the value of entry $(\hat{w}, v)$ in $(A_2 - A_1)^k$, i.e., the product of the signs along the unique length-k walk from $\hat{w}$ to v in $A_2 - A_1$ (i.e., the shortest path in $G_1 \cup G_2$).

Now generate a random $b \in \{-1, 0, +1\}^n$ as follows. First pick an arbitrary signal $\sigma \in \{\pm 1\}$; then use it to choose for each $v \in V_r$, a random $b_v \in \{\pm 1\}$ with a small bias $\delta > 0$ (determined below) towards $\sigma s_v \in \{\pm 1\}$, i.e.,

$$\mathbb{E}[b_v | \sigma] = (\tfrac{1}{2} + \tfrac{\delta}{2})\sigma s_v + (\tfrac{1}{2} - \tfrac{\delta}{2})(-\sigma s_v) = \delta \sigma s_v.$$

Observe that $\mathbb{E}[s_v b_v | \sigma] = s_v(\delta \sigma s_v) = \delta\sigma$, which means that $s_v b_v$ has a small bias towards the signal σ. Finally, let all other entries be 0, i.e., $b_v = 0$ for $v \notin V_r$. Observe that $\|b\|_2^2 = |\text{supp}(b)| = |V_r|$ and $\mathbb{E}[\sigma \sum_{v \in V_r} s_v b_v \mid \sigma] = \delta |V_r|$. We set the bias to be

$$\delta \stackrel{\text{def}}{=} C(r^2 \log d)\ |V_r|^{-1/3} \tag{4}$$

for a sufficiently large constant $C > 0$. Notice that $\{s_v\}_{v \in V_r}$ have fixed values known to the algorithm, hence observing b_v (by probing this entry of b) is information-theoretically equivalent to observing $s_v b_v$.

The next lemma is standard and follows easily from Yao's minimax principle, together with a bound on the total-variation distance between two Binomial distributions, with biases $\frac{1}{2}+\delta$ and $\frac{1}{2}-\delta$), see e.g. [9, Fact D.1.3] or [1, Eqn. (2.15)].

▶ **Lemma 4.4.** *Every randomized algorithm that, with probability at least $1/2+\gamma$ for $\gamma \in (0,1/2)$, recovers an unknown signal $\sigma \in \{\pm 1\}$ from $b_1, b_2, \ldots \in \{\pm 1\}$, each set independently to σ or $-\sigma$ with bias $\delta > 0$, must probe at least $\Omega(\delta^{-2}\gamma^2)$ entries of b.*

Thus, all probabilities from this point onward are computed over the randomness of b. We proceed to analyze $x_{\hat{\mathrm{w}}}$, aiming to show that it can be used to recover σ, namely, that with high probability $\mathrm{sgn}(x_{\hat{\mathrm{w}}}) = \sigma$. Later we will bound $\|x\|_\infty$ aiming to show a similar conclusion for $x_{\hat{\mathrm{w}}} \pm \frac{1}{5}\|x\|_\infty$. For convenience, denote $B \stackrel{\text{def}}{=} \frac{A_2 - A_1}{\mu}$, hence $\|B\| = \frac{\mu/2}{\mu} = \frac{1}{2}$ and

$$M^{-1} = (\mu(I - \tfrac{A_2-A_1}{\mu}))^{-1} = \mu^{-1}\sum_{i\geq 0} B^i,$$

and since B is symmetric, for every vertex $u \in V$ (including $\hat{\mathrm{w}}$),

$$x_u = \langle e_u, M^{-1}b\rangle = \mu^{-1}\sum_{i\geq 0}\langle e_u, B^i b\rangle = \mu^{-1}\sum_{i\geq 0} b^{\mathsf{T}}B^i e_u. \tag{5}$$

Each summand $b^{\mathsf{T}}B^i e_u$ can be viewed as the summation, over all length-i walks from vertex u, of the coordinate b_v corresponding to the walk's end-vertex v, multiplied by μ^{-i} and by the product of the signs of $A_2 - A_1$ along the walk. We can restrict the summation to walks ending at vertices $v \in V_r$, as otherwise $b_v = 0$.

▶ **Lemma 4.5.** *For every vertex $u \in V$ (including $\hat{\mathrm{w}}$),*

$$\sum_{i\geq 2r\log\mu}\left|b^{\mathsf{T}}B^i e_u\right| \leq \tfrac{1}{4}\mu^{-2r}\cdot\delta|V_r|.$$

Proof of Lemma 4.5. For each i, we have by Cauchy-Schwartz $|b^{\mathsf{T}}B^i e_u| \leq \|b\|_2\cdot\|B\|_2^i \leq |V_r|^{1/2}\cdot 2^{-i}$, and then by our choice of the bias δ in (4),

$$\sum_{i\geq 2r\log\mu}|b^{\mathsf{T}}B^i e_u| \leq |V_r|^{1/2}\sum_{i\geq 2r\log\mu} 2^{-i} \leq (|V_r|\cdot\delta/8)\cdot 2\mu^{-2r}. \qquad \blacktriangleleft$$

Recall that by Proposition 4.1, the neighborhood of $\hat{\mathrm{w}}$ of radius r_{tree} is a tree, and view it as a tree rooted at $\hat{\mathrm{w}}$. For a vertex u in this tree, let S_u be the set of all vertices $v \in V_r$ that are descendants of u; for example, $S_{\hat{\mathrm{w}}} = V_r$, and if the distance of u from $\hat{\mathrm{w}}$ is greater than r then $S_u = \emptyset$. Define a random variable $Z_u \stackrel{\text{def}}{=} \sum_{v\in S_u} s_v b_v$, whose expectation is

$$\mathbb{E}[Z_u] = \sum_{v\in S_u}\mathbb{E}[s_v b_v \mid \sigma] = |S_u|\cdot\delta\sigma.$$

▶ **Lemma 4.6.** *With probability at least $6/7$,*

$$\forall 0\leq k\leq r,\ \forall u\in V_k, \qquad \left|Z_u - \mathbb{E}[Z_u]\right| \leq O\left(\sqrt{|S_u|}\cdot\ln(3|V_k|)\right). \tag{6}$$

We remark that the constant 3 is somewhat arbitrary but needed to make sure the righthand-side is positive even for $k = 0$ (as $|V_0| = 1$). In addition, applying (6) to $\hat{\mathrm{w}} \in V_0$ yields, by our choice of the bias δ in (4),

$$\left|Z_{\hat{\mathrm{w}}} - \mathbb{E}[Z_{\hat{\mathrm{w}}}]\right| \leq O\left(\sqrt{|V_r|}\cdot\ln(3|V_r|)\right) \leq \tfrac{1}{4}\delta|V_r|. \tag{7}$$

Proof of Lemma 4.6. Fix $0 \le k \le r$ and $u \in V_k$. By Hoeffding's inequality, for every $c > 0$,

$$\begin{aligned}\Pr\left[|Z_u - \mathbb{E}[Z_u]| \ge c\sqrt{|S_u|\ln(3|V_k|)}\right] &\le e^{-2c^2|S_u|\ln(3|V_k|)/(4|S_u|)} \\ &\le e^{-(c^2/2)\ln(3|V_k|)} = (3|V_k|)^{-c^2/2}.\end{aligned}$$

By a union bound over all $u \in V_0 \cup \cdots \cup V_r$,

$$\Pr\left[\exists u,\ |Z_u - \mathbb{E}[Z_u]| \ge c\sqrt{|S_u|\ln(3|V_k|)}\right] \le \sum_{k=0}^{r} |V_k| \cdot (3|V_k|)^{-c^2/2} = \tfrac{1}{3}\sum_{k=0}^{r}(3|V_k|)^{1-c^2/2}.$$

For all $c \ge 2$ this series is decreasing geometrically, because $|V_k|$ grows at least by a factor of $2d - 1 \ge 5$, and thus the sum is dominated by its first term. By choosing c to be an appropriate constant, the first term (and the entire sum) can be made arbitrarily small. ◀

We assume henceforth that the event described in Lemma 4.6 occurs. Let W_i be the set of all walks of length i that start at $\hat{\mathrm{w}}$ and end (at some vertex) in V_r, i.e., at distance exactly r from $\hat{\mathrm{w}}$. To simplify notation (later), define

$$Q \stackrel{\text{def}}{=} \sum_{i=r}^{5r\log\mu} \mu^{-i}|W_i|.$$

We make two remarks. First, we can equivalently start the summation from $i = 0$, because $W_i = \emptyset$ for all $i < r$. Second, the range of i here complements the one in Lemma 4.5, except that the constant 5 here is intentionally bigger than the 2 there, creating a slack needed at the very end of the proof.

▶ **Lemma 4.7.** *If the event in Lemma 4.6 occurs, then*

$$x_{\hat{\mathrm{w}}} \in (\sigma \pm \tfrac{1}{2})\delta \cdot \mu^{-1}Q,$$

and thus $\operatorname{sgn}(x_{\hat{\mathrm{w}}}) = \sigma$ *(i.e., recovers the signal).*

Proof of Lemma 4.7. We would like to employ (5) and the interpretation of $b^{\mathsf{T}}B^i e_{\hat{\mathrm{w}}}$ via walks of length i. To this end, fix $0 \le i \le 5r\log\mu$. Observe that $i \le r_{\text{tree}}$, hence a walk of length i from $\hat{\mathrm{w}}$ is entirely contained in the $2d$-regular tree formed by the neighborhood of $\hat{\mathrm{w}}$ of radius r_{tree}. Each such walk contributes to $b^{\mathsf{T}}B^i e_{\hat{\mathrm{w}}}$ the value b_v at the walk's end vertex v, multiplied by all the signs seen along the walk. We make two observations. First, we can restrict attention to end vertices $v \in V_r$ (and in particular $i \ge r$), because otherwise $b_v = 0$ and the contribution is 0. Second, because it is a tree, every edge along the shortest path between $\hat{\mathrm{w}}$ and v (the start and end vertices) is traversed by the walk an odd number of times, and every other edge is traversed an even number of times. Hence, the product of the signs along the walk equals the product along that shortest path, which is exactly s_v. By symmetry, the number of walks ending at each $v \in V_r$ is the same, namely, $\frac{|W_i|}{|V_r|}$, and thus

$$b^{\mathsf{T}}B^i e_{\hat{\mathrm{w}}} = \sum_{v\in V_r} \tfrac{|W_i|}{|V_r|}\mu^{-i}s_v b_v = \tfrac{Z_{\hat{\mathrm{w}}}}{|V_r|} \cdot \mu^{-i}|W_i|. \tag{8}$$

Assuming the event in Lemma 4.6 occurs, we have $Z_{\hat{\mathrm{w}}} \in (\delta\sigma|S_{\hat{\mathrm{w}}}| \pm \frac{1}{4}\delta|V_r|) = (1 \pm \frac{1}{4})\sigma\delta|V_r|$, and therefore (recall terms for $i < r$ have zero contribution)

$$\sum_{i=0}^{5r\log\mu} b^{\mathsf{T}}B^i e_{\hat{\mathrm{w}}} \in \sum_{i=r}^{5r\log\mu} (1 \pm \tfrac{1}{4})\sigma\delta \cdot \mu^{-i}|W_i| = (1 \pm \tfrac{1}{4})\sigma\delta \cdot Q.$$

For the range of $i > 5r \log \mu$, we can use Lemma 4.5 and the obvious $|W_r| = |V_r|$ to derive

$$\Big| \sum_{i > 5r \log \mu} b^{\mathsf{T}} B^i e_{\hat{\mathrm{w}}} \Big| \leq \sum_{i > 5r \log \mu} \Big| b^{\mathsf{T}} B^i e_{\hat{\mathrm{w}}} \Big| \leq \tfrac{1}{4} \mu^{-2r} \cdot \delta |V_r| \leq \tfrac{1}{4} \delta Q.$$

Altogether, plugging into (5) we obtain

$$\mu \cdot x_{\hat{\mathrm{w}}} = \sum_{i \geq 0} b^{\mathsf{T}} B^i e_{\hat{\mathrm{w}}} \in \sum_{i=0}^{5r \log \mu} (1 \pm \tfrac{1}{4}) \sigma \delta \cdot Q \pm \tfrac{1}{4} \delta \cdot Q = (1 \pm \tfrac{1}{2}) \sigma \delta \cdot Q,$$

which proves the lemma because $\sigma \in \{\pm 1\}$. ◀

▶ **Lemma 4.8.** *If the event in Lemma 4.6 occurs, then*

$$\|x\|_\infty \leq 2\delta \cdot \mu^{-1} Q.$$

The proof of this lemma is omitted here but appears in the full version.

We can now complete the proof of Proposition 4.3. By Lemma 4.6, with probability at least 6/7 the event described therein occurs. Assume this is the case and consider an estimate $\hat{x}_{\hat{\mathrm{w}}}$ for $x_{\hat{\mathrm{w}}}$ that has additive error at most $\epsilon \|x\|_\infty$ for $\epsilon \leq \frac{1}{5}$. By Lemma 4.7 we have $x_{\hat{\mathrm{w}}} \in (\sigma \pm \frac{1}{2}) \delta \cdot \mu^{-1} Q$, and by Lemma 4.8 we have $\|x\|_\infty \leq 2\delta \cdot \mu^{-1} Q$. Altogether

$$\hat{x}_{\hat{\mathrm{w}}} \in x_{\hat{\mathrm{w}}} \pm \tfrac{1}{5} \|x\|_\infty \subseteq (\sigma \pm \tfrac{1}{2} \pm \tfrac{2}{5}) \delta \cdot \mu^{-1} Q,$$

which implies that $\operatorname{sgn}(x_{\hat{\mathrm{w}}}) = \sigma$.

Now consider a randomized algorithm for estimating $x_{\hat{\mathrm{w}}}$, and whose output $\hat{x}_{\hat{\mathrm{w}}}$ satisfies the above additive bound with probability at least 6/7. We can use this estimation algorithm to recover the signal σ, by simply reporting the sign of its estimate, namely $\operatorname{sgn}(x_{\hat{\mathrm{w}}})$. This recovery does not require additional probes to b, and by a union bound, it succeeds (in recovering σ) with probability at least 5/7. But by Lemma 4.4, such a recovery algorithm, and in particular the algorithm for estimating $x_{\hat{\mathrm{w}}}$, must probe b in at least

$$\Omega(\delta^{-2}) \geq \Omega\left(|V_r|^{2/3} / (r^4 \log^2 d) \right) \geq \Omega\left((2d-1)^{2r/3} / (r^4 \log^2 d) \right) \geq d^{\Omega(r)}$$

entries, which proves Proposition 4.3.

References

1. José A. Adell and P. Jodrá. Exact Kolmogorov and total variation distances between some familiar discrete distributions. *Journal of Inequalities and Applications*, 2006(1):64307, 2006. doi:10.1155/JIA/2006/64307.
2. Andris Ambainis. Variable time amplitude amplification and quantum algorithms for linear algebra problems. In *29th Symposium on Theoretical Aspects of Computer Science (STACS'12)*, volume 14, pages 636–647. LIPIcs, 2012. doi:10.4230/LIPIcs.STACS.2012.636.
3. Reid Andersen, Christian Borgs, Jennifer T. Chayes, John E. Hopcroft, Vahab S. Mirrokni, and Shang-Hua Teng. Local computation of PageRank contributions. *Internet Mathematics*, 5(1):23–45, 2008. doi:10.1080/15427951.2008.10129302.
4. Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. Using PageRank to locally partition a graph. *Internet Mathematics*, 4(1):35–64, 2007. doi:10.1080/15427951.2007.10129139.
5. Reid Andersen and Yuval Peres. Finding sparse cuts locally using evolving sets. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 235–244. ACM, 2009. doi:10.1145/1536414.1536449.

6 Christian Borgs, Michael Brautbar, Jennifer T. Chayes, and Shang-Hua Teng. Multiscale matrix sampling and sublinear-time PageRank computation. *Internet Mathematics*, 10(1-2):20–48, 2014. doi:10.1080/15427951.2013.802752.

7 Marco Bressan, Enoch Peserico, and Luca Pretto. Brief announcement: On approximating PageRank locally with sublinear query complexity. In *30th on Symposium on Parallelism in Algorithms and Architectures*, SPAA '18, pages 87–89. ACM, 2018. arXiv:1404.1864, doi:10.1145/3210377.3210664.

8 Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998. doi:10.1016/S0169-7552(98)00110-X.

9 Clément L. Canonne. A survey on distribution testing: Your data is big. but is it blue? *Electronic Colloquium on Computational Complexity (ECCC)*, 22:63, 2015. URL: http://eccc.hpi-web.de/report/2015/063.

10 Yen-Yu Chen, Qingqing Gan, and Torsten Suel. Local methods for estimating PageRank values. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*, CIKM '04, pages 381–389. ACM, 2004. doi:10.1145/1031171.1031248.

11 A. Childs, R. Kothari, and R. Somma. Quantum algorithm for systems of linear equations with exponentially improved dependence on precision. *SIAM Journal on Computing*, 46(6):1920–1950, 2017. doi:10.1137/16M1087072.

12 Fan Chung and Olivia Simpson. Solving local linear systems with boundary conditions using heat kernel PageRank. *Internet Mathematics*, 11(4-5):449–471, 2015. doi:10.1080/15427951.2015.1009522.

13 Fan Chung and Wenbo Zhao. *PageRank and Random Walks on Graphs*, pages 43–62. Springer, 2010. doi:10.1007/978-3-642-13580-4_3.

14 Michael B. Cohen, Rasmus Kyng, Gary L. Miller, Jakub W. Pachocki, Richard Peng, Anup B. Rao, and Shen Chen Xu. Solving SDD linear systems in nearly $m \log^{1/2} n$ time. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 343–352, 2014. doi:10.1145/2591796.2591833.

15 Danial Dervovic, Mark Herbster, Peter Mountney, Simone Severini, Naïri Usher, and Leonard Wossnig. Quantum linear systems algorithms: a primer. *CoRR*, abs/1802.08227, 2018. arXiv:1802.08227.

16 Dean Doron, François Le Gall, and Amnon Ta-Shma. Probabilistic logarithmic-space algorithms for Laplacian solvers. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017*, pages 41:1–41:20, 2017. doi:10.4230/LIPIcs.APPROX-RANDOM.2017.41.

17 Dean Doron, Amir Sarid, and Amnon Ta-Shma. On approximating the eigenvalues of stochastic matrices in probabilistic logspace. *Comput. Complex.*, 26(2):393–420, June 2017. doi:10.1007/s00037-016-0150-y.

18 George E Forsythe and Richard A Leibler. Matrix inversion by a Monte Carlo method. *Mathematics of Computation*, 4(31):127–129, 1950. doi:10.1090/S0025-5718-1950-0038138-X.

19 François Le Gall. Powers of tensors and fast matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation, ISSAC '14*, pages 296–303, 2014. doi:10.1145/2608628.2608664.

20 Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009. doi:10.1103/PhysRevLett.103.150502.

21 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc.*, 43(4):439–561, 2006. doi:10.1090/S0273-0979-06-01126-8.

22 Iordanis Kerenidis and Anupam Prakash. Quantum recommendation systems. In *8th Innovations in Theoretical Computer Science Conference (ITCS' 17)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 49:1–49:21. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.ITCS.2017.49.

23 Bo'az Klartag and Oded Regev. Quantum one-way communication can be exponentially stronger than classical communication. In *43rd Annual ACM Symposium on Theory of Computing*, STOC '11, pages 31–40, 2011. doi:10.1145/1993636.1993642.

24 Rasmus Kyng, Yin Tat Lee, Richard Peng, Sushant Sachdeva, and Daniel A Spielman. Sparsified Cholesky and multigrid solvers for connection Laplacians. In *48th Annual ACM Symposium on Theory of Computing*, pages 842–850. ACM, 2016.

25 Yin Tat Lee. Probabilistic spectral sparsification in sublinear time. *CoRR*, abs/1401.0085, 2014. arXiv:1401.0085.

26 Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988. doi:10.1007/BF02126799.

27 G. A. Margulis. Explicit group-theoretic constructions of combinatorial schemes and their applications in the construction of expanders and concentrators. *Problemy Peredachi Informatsii*, 24(1):51–60, 1988.

28 Ran Raz. Exponential separation of quantum and classical communication complexity. In *31st Annual ACM Symposium on Theory of Computing*, STOC '99, pages 358–367, 1999. doi:10.1145/301250.301343.

29 Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. In *Innovations in Computer Science - ICS 2010*, pages 223–238, 2011. URL: http://conference.iiis.tsinghua.edu.cn/ICS2011/content/papers/36.html.

30 Nitin Shyamkumar, Siddhartha Banerjee, and Peter Lofgren. Sublinear estimation of a single element in sparse linear systems. In *54th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2016*, pages 856–860, 2016. doi:10.1109/ALLERTON.2016.7852323.

31 D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *36th Annual ACM Symposium on Theory of Computing*, pages 81–90. ACM, 2004. doi:10.1145/1007352.1007372.

32 Daniel A. Spielman. Algorithms, graph theory, and linear equations in Laplacian matrices. In *Proceedings of the International Congress of Mathematicians*, volume 4, pages 2698–2722, 2010. doi:10.1142/9789814324359_0164.

33 Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, December 2011. doi:10.1137/080734029.

34 Jukka Suomela. Survey of local algorithms. *ACM Comput. Surv.*, 45(2):24:1–24:40, March 2013. doi:10.1145/2431211.2431223.

35 Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. *CoRR*, abs/1807.04271, 2018. arXiv:1807.04271.

36 Nisheeth K. Vishnoi. Lx = b. *Foundations and Trends in Theoretical Computer Science*, 8(1–2):1–141, 2013. doi:10.1561/0400000054.

37 Alastair J. Walker. An efficient method for generating discrete random variables with general distributions. *ACM Trans. Math. Softw.*, 3(3):253–256, 1977. doi:10.1145/355744.355749.

38 Wolfgang R Wasow. A note on the inversion of matrices by random walks. *Mathematical Tables and Other Aids to Computation*, 6(38):78–81, 1952. doi:10.2307/2002546.