On Solving Linear Systems in Sublinear Time

Alexandr Andoni∗ Robert Krauthgamer† Yosef Pogrow†

September 11, 2018

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 $n$. This holds even when one wants to obtain the same additive approximation, and $S$ has bounded sparsity and condition number.
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 $\omega < 2.372$ [Gal14] 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 $A^T Ax = A^T b$, see for example [Spi10]), namely, $O(mn)$ time where $m = \text{nnz}(A)$ is the number of non-zeros in $A$. Note that 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. Many applications require solving linear systems in SDD matrices, and most notably their subclass of graph-Laplacian matrices, see e.g. [Spi10, Vis13, CKM+] [72x292]. Solving SDD linear systems received a lot of attention in the past decade after the breakthrough result by Spielman and Teng in 2004 [ST04], 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. [CKM+] that runs in near-linear time $O(m\sqrt{n}(\log \log n)^O(1)\log \frac{1}{\epsilon})$. Recent improvements extend to connection Laplacians [KLP+] [72x332]. 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 PSD 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 [HHL09], and subsequently improved in [Amb12, CKS17], and meanwhile used for a number of (quantum) machine learning algorithms (see, e.g., the survey [DHM+18]). In particular, [HHL09] consider the 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 $|b\rangle = \sum_i \frac{b_i}{\|\sum_i b_i|i\|}$. Then, if the matrix $A$ has condition number $\kappa$, at most $d$ non-zeros per row/column, and $\|A\| = 1$, their quantum algorithm runs in time $\text{poly}(\kappa, d, 1/\epsilon)$, and outputs a quantum state $|\hat{x}\rangle$ within $\ell_2$-distance $\epsilon$ from $|x\rangle = \sum_i x_i|i\rangle$.

The runtime was later improved in [CKS17] to depend logarithmically on $1/\epsilon$. (The original goal of [HHL09] 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 analogues classical algorithms for the same problems; for example, a very recent success story is a classical algorithm [Tan18] for a certain variant of recommendation systems, inspired by an earlier quantum algorithm [KLP17]. Our

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

Formally, 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

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

This guarantee corresponds (modulo amplification of the success probability) to finding a solution $\hat{x} \in \mathbb{R}^n$ with $||\hat{x} - x^*||_\infty \leq \epsilon ||x^*||_\infty$. We remark that the guarantee in [ST04] is different, that $||\hat{x} - x^*||_S \leq \epsilon ||x^*||_S$ where $||y||_S = \sqrt{y^T S y}$.

**Basic Notation.** Given a (possibly edge-weighted) undirected graph $G = (V, E)$, we assume for convenience $V = [n]$. Its Laplacian is the matrix $L_G \overset{\text{def}}{=} D - A \in \mathbb{R}^{n \times n}$, 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.

For example, for the Laplacian $L_G$ of a $d$-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 ($\mu_2$ is called the spectral gap). Throughout, $||A||$ denotes the spectral norm of a matrix $A$, and $A^+$ denotes the Moore-Penrose pseudo-inverse of $A$.

### 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.

---

\textsuperscript{2}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 simpler (it becomes easier to be linear in $\kappa$), see e.g. [Spi11].

\textsuperscript{3}For a PSD matrix $A \in \mathbb{R}^{n \times n}$, let its eigen-decomposition be $A = \sum_{i=1}^n \lambda_i u_i u_i^T$, then the Moore-Penrose pseudo-inverse of $A$ is $A^+ = \sum_{i: \lambda_i > 0} \frac{1}{\lambda_i} u_i u_i^T$. 

2
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 \((G, b, u, \epsilon, \bar{\kappa})\), 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),
- \(u \in \mathbb{R}^n\), \(\epsilon > 0\), 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. If \(x^*\) satisfies \(L_Gx^* = b\) then

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

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

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 runtime 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 appears in Theorem 5.1 and is a natural generalization of Theorem 1.1 with two differences. One difference is that a natural solution to the system \(Sx = b\) is \(x = S^{-1}b\), but our method requires \(S\) to have normalized diagonal entries, and thus we aim at another solution \(x^*\), construed as follows. Define

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

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

\[
x^* \overset{\text{def}}{=} D^{-1/2}\tilde{S}D^{-1/2}b, \tag{3}
\]

that is expressed using the pseudo-inverse of \(\tilde{S}\), rather than of \(S\).

A second difference is that Theorem 5.1 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, are only that a random “neighbor” in the graph corresponding to \(S\) can be sampled quickly, and that \(\max_{j \in [n]} D_{ii} \leq \min_{j \in [n]} D_{ii} \leq \text{poly}(n)\), 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 $x = S^{-1}b$.

Theorem 1.2 (Lower Bound for PSD Systems, see Section 3). 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 $\kappa(S) \leq 3$, and a distinguished index $u \in [n]$, with the following property. Every randomized algorithm that, given as input $b \in \mathbb{R}^n$, outputs $\hat{x}_u$ satisfying

\[ \Pr \left[ |\hat{x}_u - x_u^*| \leq \frac{1}{5} \|x^*\|_\infty \right] \geq \frac{6}{7}, \]

where $x^* = S^{-1}b$, must probe $n^{O(1/d^2)}$ 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.

Theorem 1.3 (Lower Bound for Laplacian Systems, see Section 4). For every large enough $n$ and $k \leq O(n^{1/2}/\log n)$, 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 an 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 $\epsilon = \Theta(1/\log n)$, where $x^* \in \mathbb{R}^n$ is any solution to $L_Gx = b$, must probe $\Omega(k^2)$ coordinates of $b$ (in the worst case).

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 $R_{\text{eff}}(u, v)$, can be expressed as $x_u - x_v$, where $x$ solves $L_Gx = e_u - e_v$.

The spectral-sparseification algorithm of Spielman and Srivastava [SS11] 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 [Lee12] that runs in time $\tilde{O}(n)$, which is sublinear in the number of edges, and approximates effective resistances within a larger factor polyylog$(n)$.

In a $d$-regular expander $G$, the effective resistance between every two vertices is $\Theta(1/d)$, and our algorithm in Theorem 1.1 can quickly compute an arbitrarily good approximation (factor $1 + \epsilon$). Indeed, observe that we can use $B_{up} = 2$, hence the runtime is $O(\frac{1}{\epsilon^2} \log \log \frac{1}{\epsilon})$, independently of $n$. The additive accuracy is $\epsilon \cdot \max_{ij \in E(G)} |x_i - x_j|$; in fact, each $x_i - x_j$ is the potential difference between $i$ and $j$ when a potential difference of $R_{\text{eff}}(u, v)$ is imposed between $u$ and $v$, thus $\max_{ij \in E(G)} |x_i - x_j| \leq x_u - x_v = R_{\text{eff}}(u, v)$, and hence with high probability the output actually achieves a multiplicative guarantee $R_{\text{eff}}(u, v) \in (1 \pm \epsilon)R_{\text{eff}}(u, v)$.

1.2 Technical Outline

Algorithms. Our basic technique relies on a classic idea of von Neumann and Ulam [FL51, Was52] 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

\[ \forall X \in \mathbb{R}^{n \times n}, \|X\| < 1, \quad (I - X)^{-1} = \sum_{t=0}^{\infty} X^t. \]
(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., \(\hat{x}_u = e_u^T \frac{1}{d} \sum_{t=0}^{t_0} (\frac{1}{d} A)^t b\), where \(t_0\) is logarithmic (with base \(|\frac{1}{d} A|^{-1} > 1\)). In order to compute each term \(e_u^T (\frac{1}{d} A)^t b\), observe that \(e_u^T (\frac{1}{d} A)^t e_w\) 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

\[
\mathbb{E}[b_z] = \sum_{w \in V} e_w^T (\frac{1}{d} A)^t e_u b_w = e_u^T (\frac{1}{d} A)^t b.
\]

If we perform several random walks (specifically, \(\text{poly}(t_0, \frac{1}{\epsilon})\) 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 \(e_u^T (\frac{1}{d} A)^t b\).

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. We may assume \(b\) is orthogonal to \(\vec{1}\) (otherwise the linear system has no solution), hence the linear system \(Lx = b\) has infinitely many solutions, and since \(L\) is PSD, we can aim to estimate the specific solution \(x^* \equiv L^{-1/2} b\) 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 \(\hat{S}\) defined in (3), and solving the equivalent (normalized) system \(\hat{S}(D^{1/2} x) = D^{-1/2} b\). Second, general SDD matrices can have positive off-diagonal elements, in contrast to the 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. The goal is to exhibit a matrix \(S\) 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 to the communication complexity of the Vector in Subspace Problem (VSP), in which Alice has an \(n/2\)-dimensional subspace \(H \subseteq \mathbb{R}^n\), 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 \(\Omega(n^{1/3})\) [KRT1] and \(O(\sqrt{n})\) [Raz09] (while for quantum communication it is \(O(\log n)\)). To reduce this problem to linear-system solvers, let \(P_H \in \mathbb{R}^{n \times n}\) be the projection operator onto the subspace \(H\), and set \(S = I + P_H\). Consider the system \(Sx = b\), and note that Alice knows \(S\) and Bob has \(b\). 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
Our own algorithm for SDD matrices. The answer is that specifically in the analysis that follows immediately after Eqn. (1) and negative). The signed adjacency matrix $A$ should have spectral norm $\mu = \|A\| = O(\sqrt{d})$, 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 similar arguments as in 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 $b_v$ is set to $+1$ or $-1$ at random but with a small bias $\delta$ towards one of the values. Some calculations show that $\text{sgn}(\mathbb{E}_z [b_z])$, and consequently $\text{sgn}(x^*_u)$, will be according to our bias (with high probability), however discovering this $\text{sgn}(x^*_u)$ 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 prove that the solution $x^*$ has a small $\ell_\infty$-norm, so that we can argue that an $\frac{1}{2}\|x^*\|_\infty$-additive error to $x^*_u$ will not change its sign. Overall, we show we can set $t_0 = \Omega(\log n)$ and $\delta \approx (2d - 1)^n)^{-1/2}$, 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 also valid for $\mu = d$, in which case the matrix $S = 2\mu I - A$ is SDD, then it would contradict our own algorithm for SDD matrices. The answer is that $\mu \ll 2d$ is required to bound $\|x^*\|_\infty$, specifically in the analysis that follows immediately after Eqn. (1).

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

The quadratic lower bound requires both a different graph and a different distribution over $b$. We use the following graph $G$ with condition number $O(k)$: take two 3-regular expanders and connect them with $n/k$ edges ("bridge edges"). The vector $b \in \{-1, +1\}^n$ will be 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 $b_i$ is chosen $\pm 1$ 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 on average over the bridge edges $(u, v)$: 1) in the balanced case, the average of $x^*_u - x^*_v$ must be zero, and 2) in the unbalanced case, the average must be $\Omega(1)$. However, the main challenge is that the actual values may 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 endpoint 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 \((I - X)^{-1} = \sum_{i=0}^{\infty} X^t\) (for \(\|X\| < 1\)) by random walks dates back to von Neumann and Ulam [L50, Was52]. 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 [DGT17] 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 [DGT17] mentioned above, and in [DSTS17]. Chung and Simpson [CS15] 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 \(f^T e^{-t(I - P_W)} = e^{-t f^T \sum_{k=0}^{\infty} \frac{t^k}{k!} P_W^k}\) 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 [SBL18] 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 \(\text{nnz}(A)\)) 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.\(^4\)

\(^4\)Let \(Ax = b\) be a linear system where \(A\) is positive definite. Let \(\lambda\) be the largest eigenvalue of \(A\). Let \(A' \overset{\text{def}}{=} \frac{1}{\lambda}A\) and \(b' \overset{\text{def}}{=} \frac{1}{\lambda}b\). 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.
However, in contrast to our results, their expected runtime is polynomial in the input size, namely $\text{nnz}(A)^{2/3}$, 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 [BP98]) 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 to the linear system $x = \frac{1-\alpha}{n} \mathbf{1} + \alpha P x$, where $0 < \alpha < 1$ is a given parameter. In personalized PageRank, one replaces $\frac{1}{n} \mathbf{1}$ (the uniform distribution) with some $b \in \mathbb{R}^n$, e.g., a standard basis vector. Equivalently, $x$ solves the system $Sx = \frac{1-\alpha}{n} \mathbf{1}$ 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 $\alpha$, and the running time guaranteed by Theorem 5.1 is logarithmic (with base $\frac{2}{\alpha+1}$).

Algorithms for the PageRank model were studied extensively. In particular, the sublinear algorithms of [BPP18] approximate the PageRank of a vertex using $\tilde{O}(n^{2/3})$ queries to an arbitrary graph, or using $\tilde{O}((n\Delta)^{1/2})$ queries when the maximum degree is $\Delta$. Another example is the heavy-hitters algorithm of [BCTT14], 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 instances using PageRank algorithms to approximate effective resistances [CZ10], the PageRank vector itself, and computing sparse cuts [ACL07].

**Local Algorithms.** Our algorithms in Theorems 1.1 and 5.1 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 [ST04, AP03], Web analysis [CGS04, ABC+08], and distributed computing [Suo13]. Rubinfeld, Tamir, Vardi, and Xie [RTVX11] 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 [Tan18], the goal may be to produce an $\ell_2$-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 $\ell_2$-sampler from $b \in \mathbb{R}^n$, i.e., use an $\ell_2$-sampler for $b$ to construct an $\ell_2$-sampler for $x$. Another version of the problem may ask to produce heavy hitters in $x$, assuming, say 5 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.

\footnote{This kind of oracle seems necessary even when $S = I.$}
2 Laplacian Solver (for Regular Graphs)

In this section we prove Theorem \[1\]. 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 Theorem \[2.3\] for the precise statement. We will then explain the modifications required to prove Theorem \[1.1\] (which actually follows also from our more general Theorem \[5.1\]).

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 \(u_1 = \frac{1}{\sqrt{n}} \cdot \mathbf{1} \in \mathbb{R}^n\), and we can write \(A = U \Lambda U^T\) where \(U = [u_1 u_2 \ldots u_n]\) is unitary and \(\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)\). For \(u, v \in [n]\), let \(\chi_{u,v} \overset{\text{def}}{=} e_u - e_v\), where \(e_i\) is the \(i\)-th standard basis vector. Then the Laplacian of \(G\) is given by

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

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

\[
L^+ \overset{\text{def}}{=} U \cdot \text{diag}(0, (d - \lambda_2)^{-1}, \ldots, (d - \lambda_n)^{-1}) \cdot U^T.
\]

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 \(\{u_2, \ldots, u_n\}\).

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 \(|\frac{\lambda_i}{d}| < 1\),

\[
\sum_{t=0}^{\infty} \left(\frac{1}{d} A\right)^t u_i = \sum_{t=0}^{\infty} \left(\frac{\lambda_i}{d}\right)^t u_i = \frac{1}{1 - \frac{\lambda_i}{d}} u_i = \frac{d}{d - \lambda_i} u_i = d L^+ u_i.
\]

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 span\(\{\mathbf{1}\}\) and hence \(x_u - x_v\) is uniquely defined, and can be written as \(x_u - x_v = \chi_{u,v}^T L^+ b\).

Claim 2.2. For \(b\) that is orthogonal to the all-ones vector, \(|\chi_{u,v}^T L^+ b - \chi_{u,v}^T \frac{1}{d} \sum_{t=0}^{s-1} (\frac{1}{d} A)^t b| \leq \frac{d}{d - \lambda_2} ||b||_\infty\).

Proof. Using Fact 2.1,

\[
\chi_{u,v}^T L^+ b - \chi_{u,v}^T \frac{1}{d} \sum_{t=0}^{s-1} (\frac{1}{d} A)^t b = \chi_{u,v}^T \frac{1}{d} \sum_{t=s}^{\infty} (\frac{1}{d} A)^t b,
\]
Algorithm 1 $\hat{\delta}_{u,v} = \text{SolveLinearLaplacian}(G, b, ||b||_0, u, v, \epsilon, d, \mu_2)$

1. Set

$$s = \frac{\log(2\sqrt{2}e^{-1}d\mu_2\sqrt{||b||_0})}{\log(\frac{d}{\delta})},$$

and $\ell = O((\frac{d}{\delta})^{-2}\log s)$.

2. For $t = 0, 1, \ldots, s - 1$ do

(a) Perform $\ell$ independent random walks of length $t$ starting at $u$, and let $u_1^{(t)}, \ldots, u_\ell^{(t)}$ be the vertices at which the random walks ended. Independently, perform $\ell$ independent random walks of length $t$ starting at $v$, and let $v_1^{(t)}, \ldots, v_\ell^{(t)}$ be the vertices at which the random walks ended.

(b) Set $\hat{\delta}^{(t)}_{u,v} = \frac{1}{\ell} \sum_{i=\ell}||b_{u_i^{(t)}} - b_{v_i^{(t)}}||$.

3. Return $\hat{\delta}_{u,v} = \frac{1}{d} \sum_{t=0}^{s-1} \hat{\delta}^{(t)}_{u,v}$.

and thus

$$||\chi_{u,v}^L b - \chi_{u,v}^T \frac{1}{d} \sum_{t=0}^{s-1} (\frac{1}{d} A)^t b|| \leq ||\chi_{u,v}^T||_2 \cdot ||\frac{1}{d} \sum_{t=0}^{\infty} (\frac{1}{d} A)^t b||_2.$$ We know that $||\chi_{u,v}^T||_2 = \sqrt{2}$, so it remains to bound $||\frac{1}{d} \sum_{t=0}^{\infty} (\frac{1}{d} A)^t b||_2$. Decomposing $b = \sum_{i=2}^n c_i u_i$ we get that $\sum_{i=2}^n c_i^2 = ||b||^2$ and

$$\sum_{t=0}^{\infty} (\frac{1}{d} A)^t b = \sum_{i=2}^n c_i u_i \sum_{t=0}^{\infty} (\frac{\lambda_i}{d})^t = \sum_{i=2}^n \frac{(\lambda_i)^t}{1 - \lambda_i} c_i u_i = d \sum_{i=2}^n \frac{(\lambda_i)^t}{d - \lambda_i} c_i u_i.$$ Hence,

$$||\frac{1}{d} \sum_{t=0}^{\infty} (\frac{1}{d} A)^t b||_2^2 = \sum_{i=2}^n \left(\frac{\lambda_i}{d - \lambda_i}\right)^2 c_i^2 ||u_i||_2^2 \leq \left(\frac{\lambda_2}{d - \lambda_2}\right)^2 \sum_{i=2}^n c_i^2 = \left(\frac{1 - \mu_2}{\mu_2}\right)^2 ||b||^2,$$

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

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

as claimed. \hfill $\square$

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

Proof. Observe that $e_u^T (\frac{1}{d} A)^t$ is a probability vector over $V$, and $e_u^T (\frac{1}{d} A)^t e_w$ 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

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

10
and similarly $\mathbb{E}[b_{(i)}] = e_T^t(\frac{1}{d}A)^t b$. 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 $\frac{1}{t} \sum_{i \in [t]} b_{u, (i)} - e_T^t(\frac{1}{d}A)^t b \leq \frac{s}{4t} ||b||_\infty$ and $\frac{1}{t} \sum_{i \in [t]} b_{u, (i)} - e_T^t(\frac{1}{d}A)^t b \leq \frac{s}{4t} ||b||_\infty$. Recalling that $\delta_{u,v} = \frac{1}{d} \sum_{t=0}^{s-1} \frac{1}{t} \sum_{i \in [t]} (b_{u, (i)} - b_{v, (i)})$, with probability at least $1 - \frac{1}{s}$ we have $|\delta_{u,v} - \chi_T^T \sum_{t=0}^{s-1} (\frac{1}{d}A)^t b| \leq \frac{s}{2t} ||b||_\infty$, as claimed. \hfill \Box

Combining Claim 2.2 and Claim 2.3 we get that (with probability $1 - \frac{1}{s}$) $|\delta_{u,v} - \chi_T^T L^+ b| \leq \frac{s}{d} ||b||_\infty$. 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, $\max_{i \in [n]} |x_i - x_j| \geq \frac{1}{d} \max_{i \in [n]} |x_i|$. We conclude that $|\delta_{u,v} - \chi_{u,v} L^+ b| \leq \epsilon \cdot \max_{i \in [n]} |x_i - x_j|$. We now turn to the running time of Algorithm 2, which is dominated by the time it takes to perform the random walks. There are $2s \cdot \ell \cdot \delta$ 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 $2\ell$ 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 $O(s \cdot \ell \cdot d) = O(d e^{-2s^2} \log s)$. 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. \hfill \Box

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

Theorem 2.5. Given an adjacency list of a $d$-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 2 outputs $\delta_{u,v} \in \mathbb{R}$ satisfying

$$\Pr \left[ |\delta_{u,v} - \chi_{u,v} L^+ b| \leq \epsilon \cdot \max_{i,j \in E} |x_i - x_j| \right] \geq 1 - \frac{1}{s},$$

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

Remark. If we allow preprocessing of $G$, the runtime of Algorithm 2 can be reduced to $O(e^{-2s^2})$, as follows. At the preprocessing phase, compute $(\frac{1}{d}A)^t$ for all powers $t \leq s$. Then, instead of approximating $e_T^t(\frac{1}{d}A)^t b$ for all powers $t \leq s$, sample a uniform $t \in \{0, 1, \ldots, s\}$, and then, in $O(1)$ time (because the probability vector is precomputed, see [Val77]), sample $z \in [n]$ based on the probability vector $e_T^t(\frac{1}{d}A)^t$, and finally, output $\frac{1}{d} \sum_{i=0}^{s} (\frac{1}{d}A)^b$. The expectation of the output is $\frac{1}{d} \sum_{i=0}^{s} (\frac{1}{d}A)^t b$. As for concentration, since the output is in $[-\frac{s+1}{d}, \frac{s+1}{d}, ||b||_\infty - \frac{s+1}{d}]$, by the Hoeffding bound, $O(e^{-2s^2})$ many repetitions suffice to obtain (with constant probability) an approximation with additive error $\frac{s+1}{d}$ $||b||_\infty$ (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 $A' = U(A + dI)U^T$ and we can write $L = dI - A = (2dI - A')$, and thus, similarly to Fact 2.1 $L^+ x = \frac{1}{d} \sum_{i=0}^{\infty} (\frac{1}{d}A')^t x$, 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 2’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 \( \delta_{u,v} \) satisfying with high probability \( \frac{1}{2} \delta_{u,v} - \chi_{u,v} L^+ b \leq \epsilon \max_{ij \in E} |x_i - x_j| \) is obtained. When running Algorithm \( \ref{alg:partition} \) on \( G' \), the term \( s \) evaluates to \( O(\frac{2d}{\mu_2} \log(\epsilon^{-1} \frac{d}{\mu_2} ||b||_0)) \), thus, leaving the guarantee of Theorem 2.3 intact (up to constant factors).

**Proof of Theorem 2.4.** 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 \( \mu_2 \) by a lower bound on \( \mu_2 \), which in turn is easy to derive from the upper bound \( \kappa \) 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 \( \ref{alg:partition} \), and modify the analysis in the aforementioned claims to use \( e_u \) instead of \( \chi_{u,v} \). The error bound obtained from combining these lemmas is \( \frac{4}{d} ||b||_\infty \), but since each \( ||b||_0 = \sum_j L_{ij} x_j | \leq \sum_j |L_{ij}| \cdot ||x||_\infty = 2d ||x||_\infty \), we can bound the error by \( \frac{4}{d} ||b||_\infty \leq 2\epsilon ||x||_\infty \). \( \square \)

### 3 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 [PSS88] and by Margulis [Mar88] 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. [HLAW06]).

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 3.1 provides combinatorial, girth-like, information about \( G_1 \cup G_2 \). Proposition 3.2 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 3.3, that constructs a PSD linear system based on \( A_1 - A_2 \), in which the coordinates of the solution \( x \) can be analyzed, showing that recovering a specific coordinate, even approximately, requires many probes to \( b \).

**Proposition 3.1 (Proved in Section 3.2).** 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 \( r_{tree} \) is a \( 2d \)-regular tree.

**Proposition 3.2.** Let \( A_1, A_2 \) be the adjacency matrices described above, and let \( \mu \) be \( 2||A_1 - A_2|| \). Then \( \mu \leq \frac{1}{4}d^{2/3} \), and the matrix \( M = \mu I + A_1 - A_2 \in \mathbb{R}^{n \times n} \) is PSD with all its eigenvalues in the range \( \left[ \frac{1}{2} \mu, \frac{3}{2} \mu \right] \). Thus, \( M \) is invertible and has condition number \( \kappa(M) \leq 3 \).
Proof. By the triangle inequality, \( \mu/2 = \|A_1 - A_2\| \leq \|A_1 - dI\| + \|(A_2 - dI)\| \leq 2 \max\{|\lambda_2|, |\lambda_n|\} \leq \frac{1}{d}d^{2/3} \). The eigenvalues of \( A_1 - A_2 \) are in the range \([-\frac{1}{2}\mu, \frac{3}{2}\mu]\), and thus those of \( M \) are in the range \([\frac{1}{2}\mu, \frac{3}{2}\mu]\). \( \square \)

Proposition 3.3 (Proved in Section 3.1). Let the graphs \( G_1, G_2 \) be according to Proposition 3.1, let \( M \eqdef \mu I + A_1 - A_2 \in \mathbb{R}^{n \times n} \) as above, and fix \( r \leq r_{\text{tree}}/d^2 \). Then every randomized algorithm that, with probability at least 6/7, approximates coordinate \( x_\hat{w} \) of \( x = M^{-1}b \) within additive error at most \( \frac{1}{3}\|x\|_\infty \), must probe \( d^{\Omega(r)} \) entries from \( b \in \mathbb{R}^n \), even when restricted to \( b \in \{-1,0,+1\}^n \) that are 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 \( S \eqdef M \) and observe that it has the sparsity and condition number required for Theorem 1.2, and let the distinguished index be \( u \eqdef \hat{w} \). 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 3.3 and deduce that this algorithm must probe \( b \in \mathbb{R}^n \) in

\[
d^{\Omega(r)} \geq d^{\Omega(\log d n)/d^2)} \geq n^{\Omega(1/d^2)}
\]

entries, which proves Theorem 1.2.

3.1 Proof of Proposition 3.3

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 the Proposition 3.1, we can view the radius-\( r_{\text{tree}} \) neighborhood of \( \hat{w} \) as a tree rooted at \( \hat{w} \). In particular, for all \( k \leq r_{\text{tree}} \) we have \( |V_k| = 2d(2d-1)^{k-1} \simeq (2d-1)^k \). 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 unknown (or random) “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 \( \sigma s_v \), i.e.,

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

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 \( \sigma \). 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 \( \mathbb{E}[\sigma \sum_{v \in V_r} s_v b_v | \sigma] = \delta |V_r| \). We set the bias to be

\[
\delta \eqdef C(r^2 \log d) |V_r|^{-1/3}
\]

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. [Can17, Fact D.1.3] or [AJ06, Eqn. (2.15)].
Lemma 3.4. Every randomized algorithm that, with probability at least \(1/2 + \gamma\) for \(\gamma \in (0, 1/2)\), recovers an unknown signal \(\sigma \in \{-1, 1\}\) from \(b_1, b_2, \ldots \in \{-1, 1\}\), each set independently to \(\sigma\) or \(-\sigma\) with bias \(\delta > 0\), must probe at least \(\Omega(\delta^{-2}\gamma^2)\) entries of \(b\).

We proceed to analyze \(x_{\hat{w}}\), aiming to show that it can be used to recover \(\sigma\), namely, that with high probability \(\text{sgn}(x_{\hat{w}}) = \sigma\). Later we will bound \(\|x\|_\infty\) aiming to show a similar conclusion for \(x_{\hat{w}} \pm \frac{1}{\eta}\|x\|_\infty\). For convenience, denote \(B \overset{\text{def}}{=} \frac{A_2-A_1}{\mu}\), hence \(\|B\| = \frac{\mu/2}{\mu} = \frac{1}{2}\) and

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

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

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

Each summand \(b^T B^i e_u\) 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 \(\mu^{-i}\) 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 3.5. For every vertex \(u \in V\) (including \(\hat{w}\)),

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

Proof of Lemma 3.5. For each \(i\), we have by Cauchy-Schwartz \(|b^T B^i e_u| \leq \|b\|_2 \cdot \|B\|_2^i \leq |V_r|^{1/2} \cdot 2^{-i}\), and then by our choice of the bias \(\delta\) in (4),

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

Recall that by Proposition 3.3, the neighborhood of \(\hat{w}\) of radius \(r\) is a tree, and view it as a tree rooted at \(\hat{w}\). For a vertex \(u\) in this tree, let \(S_u\) be the set of all vertices \(v \in V\) that are descendants of \(u\); for example, \(S_{\hat{w}} = V_r\), and if the distance of \(u\) from \(\hat{w}\) is greater than \(r\) then \(S_u = \emptyset\). Define a random variable \(Z_u \overset{\text{def}}{=} \sum_{v \in S_u} s_v b_v\), 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 3.6. With probability at least \(6/7\),

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

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 (3) to \(\hat{w} \in V_0\) yields, by our choice of the bias \(\delta\) in (3),

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

14
Proof of Lemma 3.4. Fix $0 \leq k \leq r$ and $u \in V_k$. By Hoeffding’s inequality, for every $c > 0$,  
\[ \Pr \left[ \left| Z_u - \mathbb{E}[Z_u] \right| \geq c \sqrt{|S_u| \ln(3|V_k|)} \right] \leq e^{-2c^2|S_u| \ln(3|V_k|)/(4|S_u|)} \leq e^{-(c^2/2 \ln(3|V_k|))} = (3|V_k|)^{-c^2/2}. \]
By a union bound over all $u \in V_0 \cup \cdots \cup V_r$,  
\[ \Pr \left[ \exists u, \left| Z_u - \mathbb{E}[Z_u] \right| \geq c \sqrt{|S_u| \ln(3|V_k|)} \right] \leq \sum_{k=0}^{r} \left| V_k \right| \cdot (3|V_k|)^{-c^2/2} = \frac{1}{2} \sum_{k=0}^{r} (3|V_k|)^{1-c^2/2}. \]
For all $c \geq 2$ this series is decreasing geometrically, because $|V_k|$ grows at least by a factor of $2d - 1 \geq 5$, 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. \(\square\)

We assume henceforth that the event described in Lemma 3.4 occurs. Let $W_i$ be the set of all walks of length $i$ that start at $\hat{w}$ and end (at some vertex) in $V_r$, i.e., at distance exactly $r$ from $\hat{w}$. Define  
\[ Q \overset{\text{def}}{=} \sum_{i=r}^{5r \log \mu} \mu^{-i}|W_i|. \]
We make two remarks. First, we can equivalently start the summation from $i = 0$, because $W_i = \emptyset$ for all $i < r$. Second, the leading constant 5 here is bigger than the 2 used in Lemma 3.3, this is intentional and the slack be needed at the very end of the proof.

**Lemma 3.7.** If the event in Lemma 3.4 occurs, then  
\[ x_\hat{w} \in (\sigma + \frac{1}{3})\delta \cdot \mu^{-1}Q, \]
and thus $\text{sgn}(x_\hat{w}) = \sigma$ (i.e., recovers the signal).

Proof of Lemma 3.7. We would like to employ (8) and the interpretation of $b^T B^i e_{\hat{w}}$ via walks of length $i$. To this end, fix $0 \leq i \leq 5r \log \mu$. Observe that $i \leq r_{\text{tree}}$, hence a walk of length $i$ from $\hat{w}$ is entirely contained in the $2d$-regular tree formed by the neighborhood of $\hat{w}$ of radius $r_{\text{tree}}$. Each such walk contributes 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 $i \geq r$), because otherwise $b_v = 0$. Moreover, the same number of walks end at each $v \in V_r$, by symmetry. Second, the signs along a walk in a tree cancel, except for the signs on the shortest path between $\hat{w}$ and $v$ (the start and end vertices), hence the product of these signs 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  
\[ b^T B^i e_{\hat{w}} = \sum_{v \in V_r} \frac{|W_i|}{|V_r|} \mu^{-i}s_v b_v = \frac{Z_{\hat{w}}}{|V_r|} \cdot \mu^{-i}|W_i|. \]  
(8)
Assuming the event in Lemma 3.4 occurs, we have $Z_{\hat{w}} \in (\delta \sigma |S_{\hat{w}}| \pm \frac{1}{3} \delta |V_r|) = (1 \pm \frac{1}{3})\sigma \delta |V_r|$, and therefore (recall terms for $i < r$ have zero contribution)
\[ \sum_{i=0}^{5r \log \mu} b^T B^i e_{\hat{w}} \in \sum_{i=r}^{5r \log \mu} (1 \pm \frac{1}{3})\sigma \delta \cdot \mu^{-i}|W_i| = (1 \pm \frac{1}{3})\sigma \delta \cdot Q. \]
For the range of \( i > 5r \log \mu \), we can use Lemma \ref{lem:radiusr} and the obvious \( |W_r| = |V_r| \) to derive

\[
\left| \sum_{i>5r \log \mu} b^T B^i e_{\hat{w}} \right| \leq \sum_{i>5r \log \mu} \left| b^T B^i e_{\hat{w}} \right| \leq \frac{1}{4} \mu^{2r} \cdot \delta |V_r| \leq \frac{1}{4} \delta Q.
\]

Altogether, plugging into \( (\ref{eq:mucdotx}) \) we obtain

\[
\mu \cdot x_{\hat{w}} = \sum_{i \geq 0} b^T B^i e_{\hat{w}} \in \sum_{i=0}^{5r \log \mu} \left( 1 \pm \frac{1}{2} \right) \sigma \delta \cdot Q \pm \frac{1}{2} \delta \cdot Q = (1 \pm \frac{1}{2}) \sigma \delta \cdot Q,
\]

which proves the lemma because \( \sigma \in \{ \pm 1 \} \).

**Lemma 3.8.** If the event in Lemma \ref{lem:no-walks} occurs, then

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

**Proof.** Fix \( u \in V \), and let us bound \( |x_u| \). Similarly to the proof of Lemma \ref{lem:radiusr}, we employ \( (\ref{eq:mucdotx}) \) and interpret \( b^T B^i e_u \) via walks of length \( i \), which now start at vertex \( u \) rather than at \( \hat{w} \).

Let \( k \) be the distance of \( u \) from \( \hat{w} \), i.e., \( u \in V_k \). The case \( k > r + 2r \log \mu \) is easy, as follows. A walk of length \( i \leq 2r \log \mu \) from \( u \in V_k \) cannot end in \( V_r \) (because the distance of the end vertex from \( \hat{w} \) is at least \( k - i > r \)), hence \( b^T B^i e_u = 0 \). Plugging this information and Lemma \ref{lem:radiusr} into \( (\ref{eq:mucdotx}) \), we have

\[
|\mu \cdot x_u| \leq \sum_{i \geq 0} \left| b^T B^i e_u \right| = \sum_{i>2r \log \mu} \left| b^T B^i e_u \right| \leq \frac{1}{4} \mu^{-2r} \cdot \delta |V_r| \leq \frac{1}{4} \delta Q,
\]

which proves the lemma in this case.

We thus assume henceforth \( k \leq r + 2r \log \mu \). For each \( i \leq 2r \log \mu \), let \( U_i \) be the set of all length-\( i \) walks that start at \( u \) and end in \( V_r \). (The difference from \( W_i \) is that the walks start at \( u \) instead of at \( \hat{w} \).) Observe that such a walk (in \( U_i \)) is entirely contained in the \( 2d \)-regular tree formed by the radius-\( r_{\text{tree}} \) neighborhood of \( \hat{w} \), because the maximum distance from \( \hat{w} \) it can reach is \( k + i \leq (r + 2r \log \mu) + 2r \log \mu \leq 5r \log d \leq r_{\text{tree}} \). In addition, we claim that

\[
|W_{i+k}| \geq |V_k| \cdot |U_i| \geq (2d - 1)^k |U_i|.
\]

Indeed, we can generate walks in \( W_{i+k} \) by first walking from \( \hat{w} \) to any vertex in \( u' \in V_k \) directly, i.e., along the unique shortest path, and then "imitating" a walk from \( U_i \), in the sense of executing it from \( u' \in V_k \) instead of from \( u \in V_k \). (Formally, view a walk in \( U_i \) as a sequence in \( [2d]^k \) that determines which outgoing edge to traverse next, according to a fixed numbering of the \( 2d \) incident edge, that reserves \( 2d \) to the edge that gets us closer to \( \hat{w} \), if one exists.) This yields \( |V_k| \cdot |U_i| \) walks that are all distinct and end in \( V_r \), hence these are distinct walks in \( W_{i+k} \).

Denote the shortest path from \( \hat{w} \) to \( u \) by \( u_0 = \hat{w}, u_1, \ldots, u_k = u \); notice that these vertices are exactly the ancestors of \( u \) when we view the neighborhood of \( \hat{w} \) as a tree rooted at \( \hat{w} \). Now partition \( U_i = U_{i,0} \cup \cdots \cup U_{i,k} \) by letting each \( U_{i,j} \) contain the walks in \( U_i \) that visit \( u_j \) but not \( u_0, \ldots, u_{j-1} \), which means that \( u_j \) is the farthest (from \( u \)) ancestor visited by the walk. For example, \( U_{i,0} \) contains all walks in \( U_i \) that visit \( u_0 = \hat{w} \), and \( W_{i,k} \) contains all walks in \( U_i \) that never visit \( u_{k-1} \) and thus never visit \( V_0 \cup \cdots \cup V_{k-1} \) (of course, for \( k > r \) this cannot happen and thus \( U_{i,k} = \emptyset \)). The walks in \( U_{i,j} \) all end in \( S_{u_j} \) (recall this is the set of vertices in \( V_r \) that are descendants of \( u_j \)),

16
and by symmetry, the number of walks ending at each \( v \in S_{u_j} \) is the same, namely, \( \frac{|U_{i,j}|}{|S_{u_j}|} \), and thus similarly to (8),

\[
b^T B^i e_u = \sum_{j=0}^{k} \sum_{v \in S_{u_j}} \frac{|U_{i,j}|}{|S_{u_j}|} \mu^{-i} s_{i,v} = \sum_{j=0}^{k} \frac{Z_{u_j}}{|S_{u_j}|} \mu^{-i} |U_{i,j}|.
\]

Assuming the event in Lemma 3.3 occurs,

\[
\forall j = 0, \ldots, k, \quad \frac{Z_{u_j}}{|S_{u_j}|} \in \delta \sigma \pm O\left(\frac{\ln(3|V_k|)}{|S_{u_j}|^{1/2}}\right),
\]

and thus

\[
|b^T B^i e_u| \leq \sum_{j=0}^{k} \left| \frac{Z_{u_j}}{|S_{u_j}|} \mu^{-i} |U_{i,j}| \right| \leq \sum_{j=0}^{k} \left( \delta + \frac{\ln(3|V_k|)}{|S_{u_j}|^{1/2}} \right) \mu^{-i} |U_{i,j}|.
\]

Recall that \( u_j \) is an ancestor of \( u \), hence \( |S_{u_j}| \geq |S_u| \), and use (9) to obtain \( \sum_{j=0}^{k} |U_{i,j}| = |U_i| \leq \frac{1}{(2d-1)\varepsilon} |W_{i+k}| \), and altogether we have

\[
|b^T B^i e_u| \leq \left( \delta + \frac{\ln(3|V_k|)}{|S_u|^{1/2}} \right) \mu^{-i} \sum_{j=0}^{k} |U_{i,j}| \leq \left( \delta + \frac{\ln(3|V_k|)}{|S_u|^{1/2}} \right) \left( \frac{\mu}{2d-1} \right)^k \mu^{-(i+k)} |W_{i+k}|.
\]

To simplify notation, define the quantity (notice it does not depend on \( i \))

\[
\alpha_k \overset{\text{def}}{=} \left( \delta + \frac{\ln(3|V_k|)}{|S_u|^{1/2}} \right) \left( \frac{\mu}{2d-1} \right)^k. \tag{10}
\]

We claim that \( \alpha_k \leq \frac{3}{2} \delta \). To prove this claim, we first easily bound one part \( \delta \left( \frac{\mu}{2d-1} \right)^k \leq \delta \). For the other part, observe that \( |V_r| = |S_{u_k}| \leq 4(2d-1)^k |S_u| \), and thus

\[
\frac{\ln(3|V_k|)}{|S_u|^{1/2}} \left( \frac{\mu}{2d-1} \right)^k \leq \frac{2 \ln(3|V_r|)}{|V_r|^{1/2}} \left( \frac{\mu^2}{2d-1} \right)^{k/2} \leq \frac{2 \ln(3|V_r|)}{|V_r|^{1/2}} \left( \frac{\mu^2}{2d-1} \right)^{r/2} \leq \frac{1}{2} \delta,
\]

where the last inequality is by our assumption \( \mu \leq \frac{1}{2} d^{2/3} \), which implies \( \left( \frac{\mu^2}{2d-1} \right)^{r/2} \leq (2d-1)^{1/3} \leq |V_r|^{1/6} \), and by our choice of the bias \( \delta \) in (9). Putting these bounds together proves the claim.

With this bound \( \alpha_k \leq \frac{3}{2} \delta \) in hand, we are finally ready to conclude the lemma. Using this claim and that \( k \leq r + 2r \log \mu \leq 3r \log \mu \),

\[
\sum_{i \leq 2r \log \mu} |b^T B^i e_u| \leq \sum_{i \leq 2r \log \mu} \alpha_k \mu^{-(i+k)} |W_{i+k}| \leq \alpha_k Q \leq \frac{3}{2} \delta Q.
\]

For the range of \( i > 2r \log \mu \), we can use Lemma 3.3 and the obvious \( |W_r| = |V_r| \) to derive

\[
\sum_{i > 2r \log \mu} |b^T B^i e_u| \leq \frac{1}{4} \mu^{-2r} \cdot \delta |V_r| \leq \frac{1}{4} \delta Q.
\]

Plugging the above information into (9), we have

\[
|\mu \cdot x_u| \leq \sum_{i \geq 0} |b^T B^i e_u| \leq 2 \delta Q,
\]

which concludes the case \( k \leq r + 2r \log \mu \), and completes the proof of Lemma 3.8. \( \square \)
We can now complete the proof of Proposition 3.3. By Lemma 3.6, with probability at least 6/7 the event described therein occurs. Assume this is the case and consider an estimate $\hat{x}_\psi$ for $x_\psi$ that has additive error at most $\epsilon \|x\|_\infty$ for $\epsilon \leq \frac{1}{3}$. By Lemma 3.7 we have $x_\psi \in (\sigma \pm \frac{1}{2})\delta \cdot \mu^{-1}Q$, and by Lemma 3.8 we have $\|x\|_\infty \leq 2\delta \cdot \mu^{-1}Q$. Altogether
\[
\hat{x}_\psi \in x_\psi \pm \frac{1}{\delta} \|x\|_\infty \subseteq (\sigma \pm \frac{1}{2} \pm \frac{\delta}{\sigma}) \delta \cdot \mu^{-1}Q,
\]
which implies that $\text{sgn}(x_\psi) = \sigma$.

Now consider a randomized algorithm for estimating $x_\psi$, and whose output $\hat{x}_\psi$ satisfies the above additive bound with probability at least 6/7. We can use this estimation algorithm to recover the signal $\sigma$, by simply reporting the sign of its estimate, namely $\text{sgn}(x_\psi)$. This recovery does not require additional probes to $b$, and by a union bound, it succeeds (in recovering $\sigma$) with probability at least 5/7. But by Lemma 3.4, such a recovery algorithm, and in particular the algorithm for estimating $x_\psi$, 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 3.3.

### 3.2 Proof of Proposition 3.1

We prove Proposition 3.1 by the probabilistic method, namely, we let $G_2$ be a random isomorphic copy of $G_1$, and argue that the desired property of $G_1 \cup G_2$ holds with positive, in fact high, probability. The edge-union graph $G_1 \cup G_2$ can be equivalently constructed as follows. Start with the fixed graph $G_1$ as above and a copy of it $G'$ on a disjoint set of vertices, i.e., $V(G_1) \cap V(G') = \emptyset$; now draw a random perfect matching $M$ between $V(G_1)$ and $V(G')$, and then contract every edge of $M$. The rest of the proof considers the graph prior to the contraction. The idea is to expose the edges of $M$ gradually, and then by the principle of deferred decision, the remaining edges of $M$ form a random perfect matching between the yet unmatched vertices in $G$ and $G'$.

A walk in the edge-union graph $G_1 \cup G_2$ can be viewed as a walk in the prior-to-contraction graph, except that moves along the matching $M$ are not counted as steps. Fixing a vertex $\hat{w} \in V$ and an integer $l \geq 1$, every length-$l$ walk starting at $\hat{w}$ can be associated with a distinct sequence $a \in [1..2d]^l$ as follows. For each step $i = 1 \ldots , l$, move from the current vertex (starting at $\hat{w} \in V$) along an edge represented by $a_i$, where $a_i \in [d..d]$ corresponds to an edge in $G$, and $a_i \in [d + 1..2d]$ corresponds to an edge in $G'$. To make it more precise, recall that $G$ is fixed, hence the $d$ edges of $G$ incident to each vertex in $v \in V(G)$ have a fixed ordering (say lexicographic), and can be associated with a distinct index from $[1..d]$. (An edge $(u, v)$ may have different indices at $u$ and at $v$.) The same applies also to $G'$, except that now the indices are from $[d + 1..2d]$. Thus, each index $a_i \in [1..2d]$ represents a transition along an edge in $E(G) \cup E(G')$, but if the current vertex is not in the “correct” graph, then the walk first moves along the matching $M$ (to cross between $E(G)$ and $V(G')$), and then moves along the edge of $E(G) \cup E(G')$. Observe that step $i$ in the walk does not expose any edge of $M$ if both $a_{i-1}, a_i \in [1..d]$ or both $a_{i-1}, a_i \in [d + 1..2d]$, where by convention $a_0 \in [1..d]$.

For the neighborhood of $\hat{w}$ of radius $r_{\text{tree}}$ in $G_1 \cup G_2$ to not be a $2d$-regular tree, obviously there must exist $a \in [1..2d]^{2r_{\text{tree}}}$, whose corresponding walk is non-backtracking (i.e., no step $i \geq 2$ moves along the same edge as step $i - 1$) yet it is self-intersecting (i.e., at least one vertex is visited more than once). Our analysis employs another necessary condition, whose existence follows by
identifying a cycle near \( \hat{w} \) in \( G_1 \cup G_2 \), and two successive edges in it that originate from different graphs (they must exist because each of \( G_1 \) and \( G_2 \) has high girth). Specifically, there must exist two sequences (walks) \( a \in [1..2d]^{l'} \) and \( b \in [1..2d]^{l''} \) of lengths \( 1 \leq l', l'' \leq 2r_{\text{tree}} \), such that (i) \( a_\nu \in [1..d] \), i.e., the last step of \( a \) is in \( G \); (ii) \( b_\nu \in [d + 1..2d] \), i.e., the last step of \( b \) is in \( G' \); (iii) the last vertex in the walk \( a \) and that in \( b \) are matched to each other by \( M \); and (iv) these two last vertices were not visited by or matched to any earlier vertex in the two walks. Let \( E_{a,b} \) denote the event that requirements (i)-(iv) are satisfied. Observe that \( a \) and \( b \) may have a common prefix, from which they will obviously visit the same vertices. For instance, a common prefix of length \( l' - 1 = l'' - 1 \) corresponds to having two parallel edges (originating from \( G_1 \) and \( G_2 \)).

We now claim that \( \Pr[E_{a,b}] \leq \frac{2}{n} \) for every fixed \( a, b \) of length \( l', l'' \leq 2r_{\text{tree}} \). To see this, follow the walks corresponding to \( a \) and to \( b \), and expose the edges of \( M \) incident to all visited vertices except for the last vertex in each walk. If requirement (iv) is already violated, then the probability of \( E_{a,b} \) is 0. We may thus assume henceforth it is satisfied, which implies that after exposing at most \( l' + l'' - 1 \leq 4r_{\text{tree}} \) edges of \( M \), the last vertex in each walk is still not matched by \( M \). If we now expose the edges of \( M \) incident to the last vertex in each walk, the probability that these two vertices are matched to each other is at most \( \frac{n - 4r_{\text{tree}}}{n} \leq \frac{2}{n} \), and the claimed bound follows.

Finally, by a union bound over all possible sequence pairs \((a, b)\), the probability that the neighborhood of \( \hat{w} \) is not a 2d-regular tree, is at most

\[
\Pr[V_{a,b}E_{a,b}] \leq (2r_{\text{tree}})^2(2d)^{2r_{\text{tree}}} \cdot \frac{2}{n} \leq (4d)^{2r_{\text{tree}}} \cdot \frac{2}{n} \leq \frac{1}{\sqrt{n}}.
\]

This completes the proof of Proposition 3.

4 Square of Condition Number is Necessary

In this section, we prove Theorem 1. In particular, we show that there exist graphs for which one needs to query \( b \) at least \( t \) times where \( t \) is nearly-quadratic in the condition number of the Laplacian \( L = L_G \).

We first describe the construction of the graph \( G \). Let \( X \) be a 3-regular expander on \( n/2 \) nodes, indexed by a set \( V_X \), with girth \( \Omega(\log n) \). We build a graph \( G \) as follows. Take two copies of \( X \), termed \( X_1 \) on vertices \( 1, \ldots, n/2 \) and \( X_2 \) on vertices \( n/2 + 1, \ldots, n \). Then we pick \( n/k \) nodes in \( X_1 \), termed \( C_1 \), and the equivalent \( n/k \) nodes in \( C_2 \) (i.e., originating from the same nodes in \( V_X \)), and connect \( C_1 \) to \( C_2 \) via a matching \( M \). Let \( L \) be the Laplacian of the resulting graph \( G \).

**Lemma 4.1.** The condition number of \( L \) is \( O(k) \).

**Proof.** We need to prove that, for any unit-norm \( x \) orthogonal to all 1s, we have that \( x^T L x \geq \Omega(1/k) \) as the largest eigenvalue is \( \Theta(1) \). We can decompose the Laplacian \( L \) into 3 components, corresponding to \( X_1, X_2 \) and \( M \): \( L = L_1 + L_2 + L_M \). Similarly, decompose \( x = x' + x'' \) corresponding to vertices \( V_1 \) and \( V_2 \). Let \( m = \frac{n}{2} \sum_{i \in [n/2]} x_i' = -\frac{n}{2} \sum_{i > n/2} x_i'' \), and \( \bar{x}' \) be \( x \) minus \( m \) on the \( V_1 \) coordinates. Similarly \( \bar{x}'' \) is \( x_2 \) plus \( m \) on the \( V_2 \) coordinates. Then, we have that:

\[
x^T L x = x'^T L_1 x' + (x'')^T L_2 x'' + x'^T L_M x.
\]

For the sake of contradiction, suppose that all three terms are \( < c/k \) for small \( c > 0 \). Then, using the fact \( X_1 \) is expander, \( \frac{1}{2} \| \bar{x}' \|^2 \leq (x')^T L_1 x' \leq c/k \), i.e., \( \| \bar{x}' \| \leq \sqrt{2c/k} \). Similarly \( \| \bar{x}'' \| \leq \sqrt{2c/k} \).
Also we have that:
\[ x^T M x = \sum_{(i,j) \in M} (x_i - x_j)^2 = \sum_{(i,j) \in M} (2m + \bar{x}_i' - \bar{x}_j' )^2 \geq (2m \cdot \sqrt{n/k} - \|\bar{x}'\| - \|\bar{x}''\|)^2, \]
where the last inequality uses triangle inequality.

Since \( |x| = 1 \), we also have that 1 = \( |x| \leq m \cdot \sqrt{n} + \|\bar{x}'\| + \|\bar{x}''\| \), and thus \( m \geq 0.5/\sqrt{n} \). Plugging \( m \) into the above, we obtain that \( x^T M x \geq (1/\sqrt{k} - 2 \cdot 2c/k)^2 \geq 0.25/k > c/k \) — a contradiction.

To prove the lower bound on the number of probes into \( b \), we consider two distributions on \( b \) that can be distinguished using an estimate to \( |x_u - x_v| \). Distinguishing these two distributions will require a large number of probes into \( b \), giving us a query lower bound for estimating \( |x_u - x_v| \).

We now describe these two distributions. Partition the graph \( G \) into 4 parts, termed \( P_1, \ldots, P_4 \) as follows: the vertices \( V_1 \) are split arbitrarily into 2 equal-size sets \( V_1 = P_1 \cup P_2 \), and similarly with \( V_2 = P_3 \cup P_4 \). Consider distinguishing the following two cases, where \( p = \Theta(\sqrt{n}/k) \).

**Balanced:** For each coordinate \( u \in P_1, P_3 \) pick \( b_u \in \{ \pm 1 \} \) randomly; for \( u \in P_2 \) and \( u \in P_1 \) pick \( b_u \in \{ \pm 1 \} \) randomly conditioned on the fact that \( |\{ u \in P_2 \mid b_u = +1 \}| = |\{ u \in P_1 \mid b_u = -1 \}| \) (i.e., \( b \) is fully balanced on \( V_1 \), and similarly for \( V_2 \)).

**Unbalanced:** For each \( u \in P_1 \), set \( b_u \) to +1 with probability \( 1/2 + p/2 \) and to -1 with probability \( 1/2 - p/2 \); for \( u \in P_2 \), we similarly set \( b_u \in \{ \pm 1 \} \) randomly conditioned on \( |\{ u \in P_2 \mid b_u = +1 \}| = |\{ u \in P_1 \mid b_u = -1 \}| \) (i.e., the bias is exactly the same in \( P_1 \) and \( P_2 \)). For \( V_2 \), we do the same but with bias \( \Pr[b_u = +1] = 1/2 - p/2 \).

Note that distinguishing the two cases with probability \( \geq 1/2 + \delta \) requires probing \( \Omega(\delta^2 \cdot 1/p^2) = \Omega(k^2/\log^3 n) \) coordinates of \( b \) (see Lemma 4.4). Now let us show how to distinguish the two cases by computing \( x_u - x_v \) for some fixed edge \((u, v) \in G \), where \( x \) is the solution to \( Lx = b \).

**Lemma 4.2.** In the Balanced case, for any two vertices \( u, v \) of \( G \), we have that \( x_u - x_v = O(\sqrt{\log n}) \) with probability at least \( 1 - 1/n \).

In the Unbalanced case, for any two vertices \( u, v \) of \( G \), we have that \( x_u - x_v = O(pk \log n) + O(\sqrt{\log n}) \) with probability at least \( 1 - 1/n \).

The lemma is the core of the argument and its proof is deferred to Section 4.4.

**Lemma 4.3.** In the Unbalanced case, if we pick an edge \((u, v) \in M \) at random, then \( |x_u - x_v| = \Omega(\sqrt{\log n}) \) with probability at least \( \Omega(1/\log n) \).

**Proof.** Since \( \sum_{u \in V_1} b_u = -\sum_{u \in V_2} b_u = \Omega(np) \) with high probability, we have that \( \frac{1}{|M|} \sum_{(u,v) \in M} |x_u - x_v| \geq \frac{1}{|M|} \Omega(np) = \Omega(kp) \). Since \( \max_{u,v} |x_u - x_v| \leq O(pk \log n + \sqrt{\log n}) = O(pk \log n) \). Hence, if we pick a random \((u, v) \in M \), we have a probability of at least \( \Omega(1/\log n) \) that \( |x_u - x_v| \geq \Omega(kp) \).

**Proof of Theorem 4.3, assuming Lemma 4.2.** The proof of the theorem follows immediately from the above two lemmas. In particular, for a random edge \((u, v) \in M \), in the balanced case, we have \( |x_u - x_v| \leq O(\sqrt{\log n}) \) with \( 1 - 1/n \) probability. On the other hand, in the unbalanced case, we have \( |x_u - x_v| \geq \Omega(\sqrt{\log n}) = \Omega(1/\log n) \cdot \max_{u',v'} |x_{u'} - x_{v'}| \geq \frac{\delta}{2} \max_{u',v'} |x_{u'} - x_{v'}| \). Hence we can distinguish the two distributions with probability at least \( \Omega(1/\log n) \) as follows. Suppose the
implicit constants from Lemma 4.3 are respectively $q, w > 0$. Then we estimate $|x_u - x_v|$ and if it’s $< q\sqrt{\log n}$, then output “balanced” with probability $1/2 + \frac{w}{2}/\log n$. If $|x_u - x_v| \geq q\sqrt{\log n}$, then output “unbalanced”. This algorithm has probability of correctly distinguishing balanced vs unbalanced with probability at least $1/2 + \frac{w}{2}/\log n$ (as long as $|x_u - x_v|$ is estimated correctly, which happens with probability at least $1 - O(1/\log n)$ after a standard amplification by taking the median of $O(\log \log n)$ independent runs of the algorithm). On the other hand, any such algorithm must make at least $\Omega((1/\log n)^2 \cdot 1/p^2) = \Omega(k^2/\log^3 n)$ coordinates of $b$ (see Lemma 3.4). □

4.1 Proof of Lemma 4.2

Proof of Lemma 4.2. Consider the matrix $A$ to be the adjacency matrix of the graph $G$, where each node $i$ not in the matching $M$ has a self-loop. Thus all nodes have degree $d = 4$. Then we have that $L = dI - A$, and hence $b = Lx = dIx - Ax$, or $x = \frac{1}{d}b + \frac{1}{d}Ax$. Furthermore, using this identity iteratively, we have:

$$x = \frac{1}{d}b + \frac{1}{d}Ax = \frac{1}{d}b + \frac{1}{d}A(\frac{1}{d}b + \frac{1}{d}Ax) = \ldots = \sum_{i=0}^{t} (\frac{1}{d}A)^i \cdot (\frac{1}{d}b) + (\frac{1}{d}A)^{t+1}x.$$

Let $x' = \sum_{i=0}^{t} (\frac{1}{d}A)^i \cdot (\frac{1}{d}b)$. We take the solution $x$ such that $x \perp \bar{1}$ and hence $\|x - x'\| \leq (1 - O(1/k))^t \cdot O(k) \cdot \|b\|$. Choosing $t = \Theta(k \log n)$ ensures that $\|x - x'\| \leq 1/n$. It is thus enough to compute $x'_u - x'_v$.

Consider $x'_i = \sum_{i=0}^{t} e_u^T (\frac{1}{d}A)^i \cdot (\frac{1}{d}b)$. Note that each term corresponds to a random walk of length $t$ (using matrix $A$).

Claim 4.4. There is some $c > 1$ and $i_0$, such that for $i \geq i_0$, and any node $u$ in $G$, the following holds. In the balanced case:

$$\Pr_b \left[ \|e_u^T (\frac{1}{d}A)^i \cdot b\| \geq O(\sqrt{\log n}) \cdot (c^{-i/2} + n^{-1/2}) \right] \leq 1/n^2.$$

In the unbalanced case:

$$\Pr_b \left[ \|e_u^T (\frac{1}{d}A)^i \cdot b\| \geq p + O(\sqrt{\log n}) \cdot (c^{-i/2} + n^{-1/2}) \right] \leq 1/n^2.$$

Proof. Note that $(\frac{1}{d}A)^i$ corresponds to the following random walk on $G$. We index the vertices of $G$ as $(v, q)$ where $v \in V_X$ and $q \in \{1, 2\}$ depending whether it is in $X_1$ or $X_2$. Then the random walk is equivalent to: with probability 1/4 we take a self-loop or jump into the $X_{2-q}$ component, and with probability 3/4 we take a random step in the $X_q$ component. Thus, we can consider a graph $X'$ to be the graph $X$ where each node has a self-loop, and thus degree $d = 4$. The variable $v$ does a random walk in $X'$, independently of $q$, whereas $q$ does a more complex walk (depending on $v$). For the graph $X$, for any starting vertex, the probability that the random walk hits a particular node $l$ in $X'$ is at most $\alpha = c^{-i} + 1/n$. This is easy to note by observing that if $\frac{1}{d}AX = \sum_j \lambda_j u_j u_j^T$ is the spectral decomposition of the random walk matrix corresponding to $X$ with $\lambda_1 = 1$ and $c^{-1} \triangleq \max_{j \geq 2} |\lambda_j| < 1 - \Omega(1)$, then $\|((\frac{1}{d}AX)^i e_v - \frac{1}{n}\bar{1})\|_\infty \leq \|((\frac{1}{d}AX)^i e_v - \frac{1}{n}\bar{1})\|_2 = \|\sum_{j \geq 2} u_j \cdot \lambda_j^i \cdot \langle u_j, e_v \rangle\|_2 \leq \max_{j \geq 2} |\lambda_j^i| = c^{-i}$.

Let $\pi(l, q)$ be the probability that the random walk of length $i$, starting at $u$, stops at vertex $(l, q)$. By the above analogy between random walk in $A$ vs random walk in $X'$, we have that $\pi(l, q) \leq 2\alpha$. 21
We now use Bernstein inequality to argue about the concentration of \( \sum_{(l,q)} \pi_{(l,q)} b_{(l,q)} \). This is where the balanced and unbalanced case will differ. In the balanced case \( b_{(l,q)} \) are essentially random ±1, although there’s a minor dependence: each side is precisely balanced to 0. Hence we apply the inequality for each of the 4 parts \( P_1 \ldots P_4 \) of the graph. In each of the parts, the values \( b_{(l,q)} \) are independent. Hence, over the 4 parts, we have that, for any \( z > 0 \):

\[
\Pr_b \left[ \left| \sum_{(l,q)} \pi_{(l,q)} b_{(l,q)} \right| \geq 4z \right] \leq \exp \left[ \frac{-z^2/2}{\max_{(l,q)} \pi_{(l,q)} \cdot (1+z/3)} \right].
\]

We take \( z^2 = 2 \ln n \cdot 3 \cdot 2\alpha \), and then the probability becomes \( \leq 1/n^2 \). In particular, we have that \( z = O(\sqrt{\log n \cdot \epsilon}) = O(\sqrt{\log n \cdot (c^{-i/2} + n^{-1/2})}) \), and the claim follows for the balanced case.

In the unbalanced case, we can also consider the 4 parts \( P_1, \ldots P_4 \). For each part \( j \), we define the random variable, depending on the bias the values of \( b \). Wlog, suppose the part has a bias +p (i.e., \( b_{(l,q)} = +1 \) with probability \( 1/2 + p/2 \)). Then define \( Y_{(l,q)} = \pi_{(l,q)} b_{(l,q)} - \pi_{(l,q)} \cdot p \). Note that \( \mathbb{E}[Y_{(l,q)}] = 0 \) and \( \mathbb{E}[Y_{(l,q)}^2] = \pi_{(l,q)}^2 - (p - \pi_{(l,q)})^2 = \pi_{(l,q)}^2 \cdot (1 - p^2) \leq \pi_{(l,q)}^2 \). Applying Bernstein’s inequality similarly to before, we have that

\[
\Pr_b \left[ \left| \sum_{(l,q)} \pi_{(l,q)} b_{(l,q)} \right| \geq 4p + 4z \right] \leq \exp \left[ \frac{-z^2/2}{\max_{(l,q)} \pi_{(l,q)} \cdot (1+z/3)} \right].
\]

There’s \( z = O(\sqrt{\log n \cdot \epsilon}) \) making the above probability \( \leq 1/n^2 \). This finishes the unbalanced case.

We now complete the proof of the lemma, for the balanced and unbalanced cases. In the balanced case, we use the claim to conclude that, by union bound, with probability at least \( 1 - 1/n \),

\[
|x'_u| \leq \sum_{i=0}^{t} O(\sqrt{\log n}) \cdot (c^{-i/2} + n^{-1/2}) \leq O(\sqrt{\log n} \cdot (1 + t/n^{1/2})�.
\]

Since the same bound hold for \( x'_v \), and since \( t \leq O(k \log n) \), we have that \( |x'_u - x'_v| \leq O(\sqrt{\log n} \cdot (1 + k \log n / n^{1/2}) \) \( \leq O(\sqrt{\log n} \) for \( k < O(n^{1/4} / \log n) \).

For the unbalanced case, we have, by union bound, with probability at least \( 1 - 1/n \):

\[
|x'_u| \leq \sum_{i=0}^{t} O(p + \sqrt{\log n}) \cdot (c^{-i/2} + n^{-1/2}) \leq O(tp + \sqrt{\log n})�.
\]

Replacing \( t = O(k \log n) \) completes the unbalanced case.

5 An SDD Solver

In this section we prove the following theorem for solving linear systems in SDD matrices. To generalize from Laplacians 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 5.1 (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 (B), then

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

for suitable \( s = O(\bar{\kappa} \log(\epsilon^{-1} \kappa \|b\|_0 \cdot \max_{i \in [n]} \|D_{ii}\|_1)) \). 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 \( S \).

Given an SDD matrix \( S \in \mathbb{R}^{n \times n} \), we may assume that \( S_{ii} > 0 \) for every \( i \) (as otherwise the entire \( i \)-th row and column are zero and can be safely ignored). Recall \( D = \text{diag}(S_{11}, \ldots, S_{nn}) \), and define

\[ A \stackrel{\text{def}}{=} D - S. \]

Let \( \tilde{A} \stackrel{\text{def}}{=} D^{-1/2}AD^{-1/2} \) (for intuition, this is the normalized adjacency matrix when \( S \) is a Laplacian) and recall \( \tilde{S} = D^{-1/2}SD^{-1/2} = I - \tilde{A} \) (the normalized Laplacian, respectively). For an eigenvalue \( \mu \) of \( \tilde{A} \), let \( E_\mu(\tilde{A}) \stackrel{\text{def}}{=} \{ x : \tilde{A}x = \mu x \} \). Observe that \( \tilde{A} \preceq I \iff A \succeq D \iff S \succeq 0 \); recalling that \( S \) is SDD, we conclude that \( \tilde{A} \preceq I \). Moreover,

\[ \tilde{A}x = x \iff D^{-1/2}(D - S)D^{-1/2}x = x \iff D^{-1/2}SD^{-1/2}x = 0 \iff SD^{-1/2}x = 0, \]

so \( E_1(\tilde{A}) = D^{1/2}\ker(S) \). Observe that \( \tilde{A} \succeq -I \iff A \succeq -D \iff D+A \succeq 0 \); Since \( D+A \) is SDD, we conclude that \( \tilde{A} \succeq -I \). Let \( 1 \geq \tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_n \geq -1 \) be \( \tilde{A} \)’s eigenvalues with associated orthonormal eigenvectors \( \tilde{u}_1, \ldots, \tilde{u}_n \) (note that \( \tilde{A} \) is symmetric). We can write \( \tilde{A} = \tilde{U}\tilde{\Lambda}\tilde{U}^T \) where \( \tilde{U} = [\tilde{u}_1 \ \tilde{u}_2 \ldots \tilde{u}_n] \) is unitary and \( \tilde{\Lambda} = \text{diag}(\tilde{\lambda}_1, \ldots, \tilde{\lambda}_n) \). Note that \( \tilde{S} = \tilde{U}(I - \tilde{\Lambda})\tilde{U}^T \), and that \( \tilde{S}^+ = \tilde{U}(1/1-\tilde{\lambda}_1, \ldots, 1/1-\tilde{\lambda}_n)\tilde{U}^T \) where by convention \( \frac{1}{0} \) stands for 0. Let \( d_1 \stackrel{\text{def}}{=} D_{ii}, d_{\text{max}} \stackrel{\text{def}}{=} \max_{i \in [n]} d_i \), and \( d_{\text{min}} \stackrel{\text{def}}{=} \min_{i \in [n]} d_i \). Let \( \tilde{B} \stackrel{\text{def}}{=} \tilde{A} + \tilde{I} \). Note that \( \tilde{B} = \tilde{U}\tilde{\Lambda}+\tilde{I}\tilde{U}^T \), and that the eigenvalues of \( \tilde{B} \) are in \([0,1] \). Let \( \tilde{\lambda} \stackrel{\text{def}}{=} \max\{\frac{\tilde{\lambda}_i+1}{2} : i \in [n], \tilde{\lambda}_i < 1 \} \) (the largest non-one eigenvalue of \( \tilde{B} \)). We now describe an algorithm that on input \( b \in \mathbb{R}^n \) that is in the range of \( S \) (equivalently, is orthogonal to the kernel of \( S \)), and \( u \in [n] \), returns an approximation \( \hat{x}_u \) to \( x_u^* \), where \( x^* = D^{-1/2}\tilde{S}^+D^{-1/2}b \) is the solution for \( Sx = b \) given in (B)

We now prove that Algorithm 3 indeed provides a good approximation. Note that \( b \) is orthogonal to \( \ker(S) \) iff \( D^{-1/2}b \) is orthogonal to \( D^{1/2} \cdot \ker(S) = E_1(\tilde{A}) = E_1(\tilde{B}) \).

**Claim 5.2.** For \( b \) that is orthogonal to the kernel of \( S \),

\[
|x^*_u - \frac{1}{2}e_u^TD^{-1/2}\sum_{t=0}^{s-1}\tilde{B}^tD^{-1/2}b| \leq \frac{\epsilon}{4}\|D^{-1/2}b\|_\infty.
\]

**Proof.** Observe that

\[ 2\tilde{S}^+ = \left( I - \tilde{A} \right)^+ = \left( I - \frac{\tilde{A}+\tilde{I}}{2} \right)^+ = (I - \tilde{B})^+. \]

Thus, as \( D^{-1/2}b \) is in the span of eigenvectors of \( \tilde{B} \) with associated eigenvalues in \([0,1] \), using the same idea as in Fact 2.1 we get that

\[ \tilde{S}^+D^{-1/2}b = \frac{1}{2}\sum_{t=0}^{\infty}\tilde{B}^tD^{-1/2}b, \]
Algorithm 2 \( \hat{x}_u = \text{SolveLinearSDD}(S,b,||b||_0,u,\epsilon,\tilde{\lambda}) \)

1. Set \( s = \log_1/\tilde{\lambda}(2\epsilon^{-1}(1 - \tilde{\lambda})^{-1}\sqrt{||b||_0 \cdot \sqrt{d_{\text{max}}/d_{\text{min}}}}) \), and \( \ell = O((\frac{\epsilon}{\Delta})^{-2} \log s) \).

2. For \( t = 0,1,\ldots, s - 1 \) do

   (a) Perform \( \ell \) independent (lazy) random walks of length \( t \) starting at \( u \), where in one step from vertex \( v \), the walk stays put with probability \( \frac{1}{2} \), moves to \( v' \) with probability \( \frac{1}{2d_v} \), and terminates with the remaining probability.

   For each walk \( i \in [\ell] \), let \( u_i^{(t)} \) be the vertex where the walk ended, and let \( \sigma_i^{(t)} \) be the product of the signs along the walk where stay-put steps have sign 1 and others have \( \text{sgn}(A_{w'}) \).

   Formally, if the walk consists of \( u = u_0, u_1, \ldots, u_t \) then \( \sigma_i^{(t)} = \prod_{j \in [t]} \text{sgn}((D^{-1}A+I)_{u_{j-1},u_j}) \) and if it terminated earlier then \( \sigma_i^{(t)} = 0 \).

   (b) Set \( \hat{x}_u^{(t)} = \frac{1}{\ell} \sum_{i \in [\ell]} \sigma_i^{(t)} b_{u_i^{(t)}}/||u_i^{(t)}||_1 \).

3. Return \( \hat{x}_u = \frac{1}{\ell} \sum_{t = 0}^{s - 1} \hat{x}_u^{(t)} \).

and hence \((x_u^* = e_u^T D^{-1/2} \tilde{S}^+ D^{-1/2} b)\)

\[
x_u^* - \frac{1}{2} e_u^T D^{-1/2} \sum_{t=0}^{s-1} \tilde{B}^t D^{-1/2} b = \frac{1}{2} e_u^T D^{-1/2} \sum_{t=s}^{\infty} \tilde{B}^t D^{-1/2} b.
\]

Similarly to the proof of Claim 2.2, we now get that

\[
\begin{align*}
\left| \frac{1}{2} e_u^T D^{-1/2} \sum_{t=s}^{\infty} \tilde{B}^t D^{-1/2} b \right| &\leq \frac{1}{2\sqrt{d_u}} \sum_{t=s}^{\infty} \tilde{B}^t D^{-1/2} b
\end{align*}
\]

\[
\leq \frac{1}{2\sqrt{d_{\text{min}}}} \sum_{t=s}^{\infty} \tilde{\lambda}^t D^{-1/2} b
\]

\[
\leq \frac{1}{2\sqrt{d_{\text{min}}}} \cdot \frac{\tilde{\lambda}^s}{1 - \tilde{\lambda}} D^{-1/2} b
\]

\[
\leq \frac{1}{2} \sqrt{\frac{d_{\text{max}}}{d_{\text{min}}}} \cdot \frac{\tilde{\lambda}^s}{1 - \tilde{\lambda}} D^{-1/2} b
\]

\[
\leq \frac{1}{2} \sqrt{\frac{d_{\text{max}}}{d_{\text{min}}}} \cdot \frac{\tilde{\lambda}^s}{1 - \tilde{\lambda}} \sqrt{||b||_0 \cdot ||D^{-1} b||_\infty} = \frac{\epsilon}{4} ||D^{-1} b||_\infty.
\]

\[\square\]

Claim 5.3. With probability at least \( 1 - \frac{1}{s} \),

\[
\left| x_u - \frac{1}{2} e_u^T D^{-1/2} \sum_{t=0}^{s-1} \tilde{B}^t D^{-1/2} b \right| \leq \frac{\epsilon}{4} ||D^{-1} b||_\infty.
\]

24
Proof. Recalling that \( \tilde{A} = D^{-1/2}AD^{-1/2} \), we can write
\[
D^{-1/2} \tilde{B} t D^{-1/2} = D^{-1/2} \left( \frac{\tilde{A} + I}{2} \right) t D^{-1/2} = D^{-1/2} \left( \frac{D^{-1/2}A + I}{2} D^{-1/2} \right) t D^{-1/2} = \left( \frac{D^{-1/2}A + I}{2} \right) D^{-1/2} t D^{-1/2}.
\]
Hence (by induction), for every \( t \in \{0, 1, \ldots, s - 1\} \) and \( i \in [\ell] \), we have
\[
e_u^T D^{-1/2} \tilde{B} t D^{-1/2} b = e_u^T \left( \frac{D^{-1}A + I}{2} \right) b = \mathbb{E} [ \sigma_i \frac{b_{u(t)}}{d_{u(t)}}].
\]
By a union bound over Hoeffding bounds (as \( |\sigma_i \frac{b_{u(t)}}{d_{u(t)}}| \leq ||D^{-1}b||_\infty \)), with probability at least \( 1 - \frac{1}{s} \), for every \( t \in \{0, 1, \ldots, s - 1\} \),
\[
| \frac{1}{\ell} \sum_{i \in [\ell]} \sigma_i \frac{b_{u(t)}}{d_{u(t)}} - e_u^T D^{-1/2} \tilde{B} t D^{-1/2} b | \leq \frac{\epsilon}{2s} ||D^{-1}b||_\infty,
\]
which implies that
\[
| \frac{1}{2} \sum_{t=0}^{s-1} \tilde{\lambda} u(t) - \frac{1}{2} e_u^T D^{-1/2} \tilde{B} t D^{-1/2} b | \leq \frac{\epsilon}{4} ||D^{-1}b||_\infty.
\]
Combining Claim 5.2 and Claim 5.3 we get that (with probability \( 1 - \frac{1}{s} \)) \( |\tilde{\lambda} u - x^*_u| \leq \frac{\epsilon}{4} ||D^{-1}b||_\infty \).
Now, letting \( x \) denote any solution to the system \( Sx = b \), for every \( i \in [n] \) we have
\[
\frac{|b_i|}{d_i} = \frac{\sum_{j \in [n]} S_{ij} x_j}{d_i} \leq \frac{\sum_{j \in [n]} |S_{ij}| \cdot ||x||_\infty}{d_i} \leq \frac{2d_i ||x||_\infty}{d_i} = 2 ||x||_\infty
\]
where the last inequality is because \( S \) is SDD. Therefore, \( ||D^{-1}b||_\infty \leq 2 ||x||_\infty \), and we conclude that (with probability \( 1 - \frac{1}{s} \)) \( |\tilde{\lambda} u - x^*_u| \leq \epsilon ||x||_\infty \) for every solution \( x \) to the system \( Sx = b \) (and in particular for \( x^* \)).
We now turn to the runtime of Algorithm 5, which is dominated by the time it takes to perform the random walks. There are \( s \cdot \ell \) random walks in total. Let \( f \) be the time it takes to make a single step in the random walks of Algorithm 5 (it depends on the access method/representation of \( S \) and/or its sparsity). 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 \( \ell \) respective random walks constructed at iteration \( t - 1 \) by an extra step in time \( f \), obtaining a total runtime \( O(s \cdot \ell \cdot f) = O(\epsilon^{-2} s^3 \log s) \). We conclude the following.

Theorem 5.4. Given access to an SDD matrix \( S \in \mathbb{R}^{n \times n} \), \( b \in \mathbb{R}^n \) that is orthogonal to the kernel of \( S \), \( ||b||_0, u \in [n], \epsilon > 0 \), and \( \tilde{\lambda} = \max \{ \tilde{\lambda}_i : i \in [n], \tilde{\lambda}_i < 1 \} \), with probability at least \( 1 - \frac{1}{s} \), Algorithm 5 outputs a value \( \tilde{x}_u \in \mathbb{R} \) such that \( |\tilde{x}_u - x^*_u| \leq \epsilon ||x||_\infty \) for every solution \( x \) to the system \( Sx = b \) (and in particular for \( x^* \)). Algorithm 5 runs in time \( O(\epsilon^{-2} s^3 \log s) \) where \( f \) is the worst-case time to make a single step in a random walk in the weighted graph formed by the non-zeros of \( S \), and \( s = O(\log_{1/\tilde{\lambda}}(\epsilon^{-1}(1 - \tilde{\lambda})^{-1}||b||_0 : \frac{d_{\max}}{d_{\min}})) \).
Proof of Theorem 5.1. Recall $1 \geq \tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_n \geq -1$ are the eigenvalues of $\tilde{A} = I - \tilde{S}$, and that $\tilde{\lambda} = \max\{\tilde{\lambda}_i + 1 : i \in [n], \tilde{\lambda}_i < 1\}$. Then the smallest non-zero eigenvalue of $\tilde{S}$ is $1 - \tilde{\lambda} = 1 - (2\tilde{\lambda} - 1) = 2(1 - \tilde{\lambda}) > 0$. The largest eigenvalue of $\tilde{S}$ is $1 - \tilde{\lambda}_n \in [1, 2]$ where the lower bound is by the following argument (since all diagonal entries in $\tilde{S}$ are 1, and the off-diagonal entries contribute 0 in expectation)

$$\max_{x \neq 0} \frac{x^T \tilde{S} x}{x^T x} \geq \mathbb{E}_{x \in \{\pm 1\}^n} \left[ \frac{x^T \tilde{S} x}{n} \right] = 1.$$ 

Thus, $\kappa(\tilde{S}) = \Theta(\frac{1}{1 - \tilde{\lambda}})$. Using Fact 2.4 with $\delta = 1 - \tilde{\lambda}$, the expression above for $s$ becomes $s = O(\kappa(\tilde{S}) \log(\epsilon^{-1} \kappa(\tilde{S})) ||b||_0 \cdot \frac{d_{\max}}{d_{\min}})$.

The theorem follows by noting that the analysis in Claims 5.2 and 5.3 holds also when replacing $||b||_0$ by an upper bound $B_{up} \geq ||b||_0$ and $\kappa(\tilde{S})$ by an upper bound $\bar{\kappa}$ (or equivalently $\tilde{\lambda}$ by an upper bound $\tilde{\lambda}_{up}$). \hfill \square

Acknowledgments

The authors thank anonymous reviewers for suggesting additional relevant references.

References

[ABC+08] R. Andersen, C. Borgs, J. T. Chayes, J. E. Hopcroft, V. S. Mirrokni, and S. Teng. Local computation of PageRank contributions. *Internet Mathematics*, 5(1):23–45, 2008. [doi:10.1080/15427951.2008.10129302]

[ACL07] R. Andersen, F. R. K. Chung, and K. J. Lang. Using PageRank to locally partition a graph. *Internet Mathematics*, 4(1):35–64, 2007. [doi:10.1080/15427951.2007.10129139]

[AJ06] J. 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]

[Amb12] A. Ambainis. Variable time amplitude amplification and quantum algorithms for linear algebra problems. In *STACS’12 (29th Symposium on Theoretical Aspects of Computer Science)*, volume 14, pages 636–647. LIPIcs, 2012. [doi:10.4230/LIPIcs.STACS.2012.636]

[AP09] R. Andersen and Y. 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:10.1145/1536414.1536443]

[BBCT14] C. Borgs, M. Brautbar, J. T. Chayes, and S. Teng. Multiscale matrix sampling and sublinear-time PageRank computation. *Internet Mathematics*, 10(1-2):20–48, 2014. [doi:10.1080/15427951.2013.802752]

[BP98] S. Brin and L. 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]

[BPP18] M. Bressan, E. Peserico, and L. 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]

[Can15] C. L. Canonne. A survey on distribution testing: Your data is big, but is it blue? *Electronic Colloquium on Computational Complexity (ECCC)*, 22:63, 2015. Available from: [http://eccc.hpi-web.de/report/2015/063](http://eccc.hpi-web.de/report/2015/063).
[CGS04] Y.-Y. Chen, Q. Gan, and T. 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.

[CKM+14] M. B. Cohen, R. Kyng, G. L. Miller, J. W. Pachocki, R. Peng, A. B. Rao, and S. C. Xu. Solving SDD linear systems in nearly $m \log^{3/2} n$ time. In Proceedings of the 46th Annual ACM Symposium on Theory of Computing, pages 343–352, 2014. doi:10.1145/2591796.2591833.

[CKS17] 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.

[CS15] F. Chung and O. 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.

[CZ10] F. Chung and W. Zhao. PageRank and Random Walks on Graphs, pages 43–62. Springer, 2010. doi:10.1007/978-3-642-13580-4_3.

[DGT17] D. Doron, F. L. Gall, and A. 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.2017.43.

[DHM+18] D. Dervovic, M. Herbster, P. Mountney, S. Severini, N. Usher, and L. Wossnig. Quantum linear systems algorithms: a primer. CoRR, abs/1802.08227, 2018. arXiv:1802.08227.

[DSTS17] D. Doron, A. Sarid, and A. 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.

[FL50] G. E. Forsythe and R. 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.

[Gal14] F. L. 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.

[HHL09] A. W. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for linear systems of equations. Phys. Rev. Lett., 103:150502, Oct 2009. doi:10.1103/PhysRevLett.103.150502.

[HLW06] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. Bull. Amer. Math. Soc., 43(4):439–561, 2006. doi:10.1090/S0273-0979-06-01126-8.

[KLP+16] R. Kyng, Y. T. Lee, R. Peng, S. Sachdeva, and D. A. Spielman. Sparsified Cholesky and multigrid solvers for connection Laplacians. In 48th Annual ACM Symposium on Theory of Computing, pages 842–850. ACM, 2016.

[KP17] I. Kerenidis and A. 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.

[KR11] B. Klartag and O. 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.

[Lee14] Y. T. Lee. Probabilistic spectral sparsification in sublinear time. CoRR, abs/1401.0085, 2014. arXiv:1401.0085.
[LPS88] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988. DOI:10.1007/BF02126799.

[Mar88] G. A. Margulis. Explicit group-theoretic constructions of combinatorial schemes and their applications in the construction of expanders and concentrators. *Problem Peredachi Informatsii*, 24(1):51–60, 1988.

[Raz99] R. 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.

[RTVX11] R. Rubinfeld, G. Tamir, S. Vardi, and N. Xie. Fast local computation algorithms. In *Innovations in Computer Science - ICS 2010*, pages 223–238, 2011. Available from: http://conference.iis.tsinghua.edu.cn/ICS2011/content/papers/36.html.

[SBL16] N. Shyamkumar, S. Banerjee, and P. Loogren. 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.

[Spi10] D. 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.

[SS11] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. *SIAM J. Comput.*, 40(6):1913–1926, December 2011. DOI:10.1137/080734029.

[ST04] 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.

[Suo13] J. Suomela. Survey of local algorithms. *ACM Comput. Surv.*, 45(2):24:1–24:40, March 2013. DOI:10.1145/2431211.2431223.

[Tan18] E. Tang. A quantum-inspired classical algorithm for recommendation systems. *CoRR*, abs/1807.04271, 2018. arXiv:1807.04271.

[Vis13] N. K. Vishnoi. Lx = b. *Foundations and Trends in Theoretical Computer Science*, 8(1–2):1–141, 2013. DOI:10.1561/0400000054.

[Wal77] A. 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.

[Was52] W. 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.