Efficient Approximate Solution of Sparse Linear Systems

J. H. REIF
Department of Computer Science, Duke University
Durham, NC 27708-0129, U.S.A.
reif@cs.duke.edu.

Abstract—We consider the problem of approximate solution $\tilde{x}$ of of a linear system $Ax = b$ over the reals, such that $\|A\tilde{x} - b\| \leq \epsilon\|b\|$, for a given $\epsilon, 0 < \epsilon < 1$. This is one of the most fundamental of all computational problems. Let $\kappa(A) = \|A\|\|A^{-1}\|$ be the *condition number* of the $n \times n$ input matrix A. Sparse, Diagonally Dominant (DD) linear systems appear very frequently in the solution of linear systems associated with PDEs and stochastic systems, and generally have polynomial condition number. While there is a vast literature on methods for approximate solution of sparse DD linear systems, most of the results are empirical, and to date there are no known proven linear bounds on the complexity of this problem. Using iterative algorithms, and building on the work of Vaidya [1] and Gremban *et al.* [2–4] we provide the best known sequential work bounds for the solution of a number of major classes of DD sparse linear systems. Let $\pi = \log(\kappa(A)/\epsilon)$. The *sparsity graph* of A is a graph whose nodes are the indices and whose edges represent pairs of indices of A with nonzero entries. The following results hold for a DD matrix A with nonzero off-diagonal entries of bounded magnitude:

(1) if A has a sparsity graph which is a regular d-dimensional grid for constant d, then our work is $O(n\pi^2)$,
(2) if A is a stochastic matrix with fixed $s(n)$-separable graph as its sparsity graph, then our work is $O((n + s(n)^2)\pi)$.

The following results hold for a DD matrix A with entries of unbounded magnitude:

(3) if A is sparse (i.e., $O(n)$ nonzeros), our work is less than $O(n(\pi + \log n))^{1.5}$,
(4) if A has a sparsity graph in a family of graphs with constant size forbidden graph minors (e.g., planar graphs), then our work is bounded by $O(n(\pi+\log n)^{1+o(1)})$ in the case $\log n = o(\log \pi)$ and $O(n(\pi + \log n))^{1+o(1)}$ in the case $\log \pi = o(\log n)$.

We use approximate preconditioned iterations to compute a sequence of iterative improvements to the preconditioned linear system. For class (1) of matrices (and class (2) with $s(n) = O(\sqrt{n})$), we construct in n) work preconditioners, which reduce the condition number of the resulting preconditioned linear system condition number to $O(1)$; and our resulting total work bounds to approximately solve the linear systems are $O(n)$, if $\pi = O(1)$. For class (4), we are within a small increasing factor of these bounds. © 1998 Elsevier Science Ltd. All rights reserved.

Keywords—Linear systems, Sparse matrices, Approximate solution, Condition number, Preconditioners, Iterations.

Thanks to D. Armon, P. Klein, S. Guattary, K. Gremban, B. Hendrickson, M. Procopiuc, O. Procopiuc, K. Redding, K. Robinson, H. Shao, and X. Sun for edits to this paper. Also thanks to A. Pothen, who provided a proof of Proposition 1.1.

A postscript version of this paper is at http://www.cs.duke.edu/~reif/paper/solve.ps. Supported by Grants NSF CCR-9725021, NSF IRI-9619647, NSF/DARPA CCR-92725021, and ARO Contract DAAH-04-96-1-0448.

Preprint of paper appearing in Computers and Mathematics with Applications, Vol. 36, No. 9, Nov. 1998, pp 37-58.

1. INTRODUCTION

This paper is concerned with the approximate solution of a linear system $Ax = b$ over the reals, where $A = \{a_{i,j}\}$ is assumed throughout this paper to be an $n \times n$ symmetric nonsingular matrix and b is a column vector of size n. All of our algorithms will assume unit cost arithmetic scalar operations. The arithmetic bounds for the solution of dense linear systems are known to be within a constant factor of the bounds for $n \times n$ dense matrix product: $O(n^\omega)$. The currently best known bound is $\omega \leq 2.376\ldots$ [5], however, a practical bound for ω is at best 2.84.

1.1. Sparse Linear Systems

Recall from the abstract that the sparsity graph of a symmetric A is a graph with node set $V = \{1, \ldots, n\}$ and edge set $E = \{(i,j) \mid a_{i,j} \neq 0\}$. Let m be the number of nonzero elements in A. We will assume throughout that $m \geq n$. A is *sparse* if $m = O(n)$. A matrix is *symmetric* if $A = A^\top$, where $A^\top$ is the transpose of A. Our algorithms will provide competitive performance only in the case of sparse, symmetric matrices. Fortunately, the large linear systems found in practice are generally sparse and symmetric and have additional properties which allow for more efficient solution.

1.2. Diagonally Dominant Linear Systems

A is *diagonally dominant* (DD), if, for each $i, 1 \leq i \leq n$, we have $|a_{i,i}| \geq \sum_{j,j\neq i} |a_{i,j},|$. That is, at each row, the diagonal element has magnitude greater than the sum of the magnitudes of the off-diagonal elements of that row.

In practice, DD linear systems frequently arise from discretization of elliptic Partial Differential Equations (PDEs), with bounded coefficients, (e.g., Laplace's equation) and the approximate numerical solution of these PDEs is given by the approximate solution of corresponding DD linear systems. Stochastic linear systems used in statistical computation and probabilistic analysis generally have the property that the sum of the magnitudes of the off-diagonal elements of each row is equal to the diagonal element (usually of value 1), due to the requirement that probabilities of exhaustive events sum to 1, and thus these stochastic linear systems are DD.

1.3. Vector Norms and Condition

Let $x^\top$ and $A^\top$ denote the transpose of vector x and matrix A. Let $\|x\| = \sqrt{\sum_{i=1}^n x_i^2}$ denote the L_2 *(Euclidean) norm* of n-vector x. The *matrix norm* is $\|A\| = \sup_{x\neq 0}(\|Ax\|/\|x\|)$. Recall from the abstract that $\kappa(A) = \|A\|\|A^{-1}\|$ is the condition number of A. We will also define the generalized norm $\|x\|_M = \sqrt{x^\top Mx}$ for vector x with given matrix M.

1.4. Positive Definite Matrices

Matrix A is *Positive Definite* (PD) if $x^\top Ax > x$ for all $x \neq 0$. A is SPD if A is symmetric and PD.

Proposition 1.1. *If A is symmetric, nonsingular and DD and all the diagonal elements are positive, then A is SPD.*

Proof. (Attributed to Pothen; also see [6, p. 140].) The eigenvalues of a symmetric A are within the *gershgorin disks* (see [7]), which for each $i, 1 \leq i \leq n$, are centered at $a_{i,i}$ and have radius $\sum_{j,j\neq i} |a_{i,j}|$. Since A is symmetric, all the eigenvalues are real, and since A is DD, this implies that all the eigenvalues are positive, which implies that A is SPD. ∎

Throughout the paper we assume that the input matrix A is symmetric, nonsingular, and DD. We can also assume, w.l.o.g., A has all positive diagonal elements (by multiplication of A by a diagonal matrix whose i^{th} diagonal element is 1 if $a_{i,i} \geq 0$, and else -1 if $a_{i,i} < 0$, and, noting that multiplication by a diagonal matrix always preserves the symmetric matrix property), so A is SPD.

1.5. The Normal Reduction to SPD Matrices

Given a nonsingular matrix A' that is not necessarily PD, then the *normal form* $A'^{\top}A'$ is SPD. We can approximately solve a linear system $A'x = b'$ by approximately solving the SPD linear system $(A'^{\top}A')x = A'^{\top}b'$. If A' is symmetric, the condition number is $\kappa(A'^{\top}A') \leq \kappa(A')^2$. This increase in condition number does not much affect the complexity of our algorithms, which grow as $\log \kappa(A')$ or $\log^{1.5} \kappa(A')$.

1.6. Forbidden Graph Minors and Graph Separators

A family $\mathcal{F}$ of graphs has *$s(n)$-separators* if for sufficiently large n, for each n node graph G in the family, there is a separator set V' of $s(n)$ nodes which, when deleted, disconnects the graph into subgraphs G_1, G_2 each of $\leq cn$ nodes, for some constant $c, 0 < c < 1$, and such that the graph G/G_j derived by removing subgraph G_j from G is in $\mathcal{F}$, for $j = 1, 2$. We say a graph is *$s(n)$-separable* if it is in a family $\mathcal{F}$ of graphs with $s(n)$-separators. A graph H is a *minor* of graph G if there is a subgraph of G that can be contracted to H by edge contractions. Forbidden graph minors can be used to characterize large classes of sparsity graphs associated with sparse linear systems. For example, the planar graphs are exactly the family of graphs whose forbidden minors are the clique of 5 nodes K_5 and the bipartite graph $K_{3,3}$. Similarly, constant size sets of forbidden minors can be used to characterize sparsity graphs arising from various discretizations of 2D PDEs. Fix a (minor closed) family $\mathcal{F}$ of graphs defined by a constant size set $S_{\mathcal{F}}$ of forbidden graph minors. $\mathcal{F}$ is closed under edge contractions [8]; that is, if we apply a sequence of edge contraction to a graph of $\mathcal{F}$, the result is a graph in $\mathcal{F}$. Then by [8,9], there is some fixed graph that is not the minor of any graph in $\mathcal{F}$, and hence, the graphs in $\mathcal{F}$ are sparse, with $m = O(n)$ edges and they have $O(\sqrt{n})$ separators.

1.7. Direct Methods

The arithmetic bounds for the solution of dense linear systems are known to be within a constant factor of the bounds for $n \times n$ dense matrix product: $O(n^{\omega})$. The currently best known bound is $\omega \leq 2.376\ldots$ [5], however a practical bound for ω is at best 2.84.

There are many diverse techniques for the solution of sparse linear systems. *Direct methods* generally use a carefully chosen ordering to apply Gaussian elimination of variables, yielding a decomposition $A = LL^{\top}$ (where L is lower triangular) known as a *$LL^{\top}$-factorization*, which always exists assuming A is SPD, or symmetric DD. (If A is SPD, the $LL^{\top}$-factorization is called a *Cholesky $LL^{\top}$-factorization* and L is lower triangular with positive diagonal entries.) For general sparse linear systems, the bounds of Gaussian elimination are at least $\Omega(n^2)$. However, for sparse linear systems that have sparsity graphs with separators of size $s(n)$, the work bound to construct a sparse $LL^{\top}$-factorization using nested dissection orderings [10,11] is $O(n + s(n)^{\omega})$.

For example, for matrices with planar sparsity graphs, where $s(n) = O(\sqrt{n})$, the work bounds for sparse $LL^{\top}$-factorization are $O(n^{\omega/2})$; and for d-dimension grids (that is an n node grid graph in d dimensions), where $s(n) = n^{d-1/d}$, the work bounds for sparse $LL^{\top}$-factorization are $O(n^{\omega(d-1)/d})$. Known lower bounds for fill-in [10] imply that these are the best bounds possible by use of such direct Gaussian elimination techniques (up to improvement in ω). However, once a sparse $LL^{\top}$-factorization is constructed for a fixed A, then the system $Ax = b$ can be solved (by back-solving) in work $O(n + s(n)^2)$ for any column n-vector b.

1.8. Our Results and Previous Preconditioned Iterations

OUR METHODS. We use *Preconditioned Iterative* (PI) methods (see Section 2) to compute the approximate solution a number of major classes of DD sparse linear systems. We provide with considerably better *provable* bounds than known previous bounds. A summary of our improved work bounds for various sparse linear systems is given in the Abstract and a detailed description of the results are given in Section 4.

Building on the work of Vaidya [1] and Gremban *et al.* [2–4], we apply an interesting combination of combinatorial and algebraic methods to construct our preconditioners and to bound the condition number of the resulting preconditioned matrices. The key to our performance improvement is to obtain preconditioned matrices with small condition number,using a variety of techniques including *multigrid preconditioners* and *maximum spanning tree preconditioners*, as well as *recursion*, in certain cases.

COMPARISON WITH PREVIOUS PRECONDITIONED ITERATIONS. There is a vast amount of literature on preconditioned iteration (PI) methods, however, much of the literature is empirical; there is only a very small number of proven complexity results (which we will significantly improve on). This area represents a prime area where theoretical computer scientists might have impact, particularly since good preconditioners require one to juggle two contrasting requirements:

(i) fast construction, and
(ii) low condition number,

and moreover, the methodologies for construction and analysis of preconditioners involve an interesting hybrid of combinatorial and algebraic techniques.

Reference [6] provides a short introduction to preconditioning, [12–14] provide analysis of preconditioned conjugate gradient, multilevel preconditioners are presented in [15,16], and [17–20] provide efficient implementations of preconditioned iterative methods. Preconditioners have been constructed using a wide variety of methods, including diagonal scaling, partial Gaussian elimination resulting in sparse partial $LL^{\top}$-factorizations, and algebraic transforms. For example, consider preconditioners for a class of matrices we will denote BGRID_n, consisting of symmetric DD matrices of size $n \times n$ which are $O(1)$-bounded, and where the sparsity graph is an n node regular d-dimensional grid of constant dimension d. Such matrices frequently arise in the discrete solution (via uniform grids) of PDEs with bounded coefficients. Multilevel preconditioners based on symmetric successive over-relaxation [15] and modified incomplete Cholesky [21] have been constructed for this class BGRID_n of matrices, with upper bounds of $O(n^{1/d})$ on the condition number of the preconditioned matrix, but they may require the tuning of certain parameters to achieve this performance. A *Laplacian matrix* is a real symmetric DD matrix where each off-diagonal element is nonpositive. For the subclass of Laplacian matrices in BGRID_n, Gremban *et al.* [2,3] defined another effective class of preconditioners they call support trees resulting in a preconditioned matrix with condition number $O(n^{1/d} \log^{O(1)} n)$ (their work was also generalized [4] to all real symmetric DD matrices in BGRID_n). These methods compute an ϵ-solution of linear systems with this matrix class BGRID_n and subclass in work $O(n^{1+1/d} \log^2(\kappa(A)/\epsilon))$, or $O(n^{1+1/d}(\log^{O(1)} n) \log^2(\kappa(A)/\epsilon))$, respectively. In contrast, for the class of matrices BGRID_n, by Theorem 4.1, our resulting preconditioned matrices have a constant condition number, and by Corollary 4.1, our total work to compute an ϵ-solution is linear in n.

Vaidya [1] proposed the use of preconditioners based on *maximum spanning trees*, which do not require an input matrix to be $O(1)$-bounded. For certain results, we will apply Vaidya's techniques for those of our preconditioners based on minimum spanning trees, and improve both the condition number and the resulting work bounds (a detailed comparison of his work bounds with ours are given just after the statement of our results: Corollaries 4.3 and 4.4). However, we note that his results do not extend (as ours do) to d dimensional grids with $d > 2$. More significantly, for our main results (Corollary 4.1) for grid graphs, we use instead Multigrid Preconditioners, which were not considered by Vaidya.

One further significant aspect of the work of both Vaidja [1] and Gremban [4] is the use of a powerful combination of algebraic methods (initiated by Axelsson [13]) and combinatorial methods for the analysis of preconditioners. Their approach is to define certain mappings (known as support mappings) to and from the sparsity graph of the input matrix A and the sparsity graph of the preconditioner matrix B, and to bound the condition number of the preconditioned matrix $B^{-1}A$ by use of certain key parameters (namely, the congestion and dilation) of these mappings.

1.9. Organization

We give in this section definitions for *sparse linear systems* in Section 1.1, *diagonally dominant* (DD) linear systems in 1.2, *vector norms* and *condition number* in 1.3, *positive definite* (PD) and *Symmetric Positive Definite* (SPD) matrices in 1.4, the *normal reduction to SPD matrices* in 1.5, *forbidden graph minors* and *graph separators* in 1.6, *direct methods* for solution of linear systems in 1.7, a *comparison with previous preconditioned iterations* in the Appendix, and the organization of our paper in 1.9. Next, in Section 2, we describe iterative methods for the solution of linear systems, including *base iterative methods* (e.g., Conjugate Gradient and Chebyshev Semi-iterative) in Section 2.1, *preconditioned iterations* in 2.2, and *approximate preconditioned iterations* in 2.3. In Section 3, we describe combinatorial methods, known as *support mappings*, for bounding the condition number of preconditioned matrices. In Section 4, we state our main results (also summarized in the abstract), which give improved work bounds for approximate solution of sparse linear systems. We prove, in Section 4, some results for sparse matrices with bounds on the magnitude of the coefficients of the input matrix. In Section 5, we prove our results for the case where these coefficients have no magnitude bounds. Section 6 concludes the paper with a brief summary, open problems, and acknowledgments. The Appendix provides more details of the *recursive preconditioned iterations* used in Section 5.

2. ITERATIVE METHODS

For a given *relative error* bound $\epsilon, 0 < \epsilon < 1$, let an ϵ-solution of linear system $Ax - b$ be a n-vector $\tilde{x}$ such that $\|A\tilde{x} - b\| \leq \epsilon\|b\|$. A wide variety of *iterative methods* can be used to provide ϵ-solution of sparse linear systems. Newton methods for inverse of A within relative error ϵ (provided with a good initial approximation) can quickly converge with second order convergence in $O(\log\log(\kappa(A)/\epsilon))$ iterations, but require matrix product on each iteration, with high cost $O(n^{\omega})$. Multigrid methods [22–27] cost $O(n)$ per iteration but can be proved to converge quickly only for certain restricted classes of matrices, e.g., those derived from discretization of elliptic PDEs with bounded coefficients multigrid. (Note: When multigrid methods do converge fast, they work well in practice, but they may not very robust. A small perturbation of a linear system may result in nonconvergence or very slow convergence. Other iteration methods, such as conjugate gradient, often have the greater robustness and direct methods can generally be much more robust.)

2.1. Base Iterative Methods

For these reasons, many practitioners use highly optimized versions of classical iterative methods such as Jacobi, Gauss-Seidel, SOR, conjugate gradient and Chebyshev. Semi-iterative methods to approximately solve general sparse linear systems. A succinct introduction to these iterative methods is given in [6]; also see [28–32] for further details, see [33] for sequential implementations, see [34,35] for parallel implementations, and see [36–38] for detailed analysis of the Conjugate Gradient method. These methods have the advantage that the most costly work per iteration consists of an inner product of the input matrix with a vector, but have the disadvantage of requiring potentially more iterations than Newton's method. In our algorithms, we will choose and fix one of these iterative methods IMPROVE and call it the *base iterative method.* (Later, we will restrict the base iterative method to be one of the last two listed, and consider enhancements of the base method so as to decrease the number of iterations.) Each iterative stage of a base iterative method consists of an iterative improvement procedure of the form

$$(x_{i+1}, y_{i+1}) = \text{IMPROVE}\,(A, b, x_i, y_i)\,,$$

where x_i is the approximation to the solution on the i^{th} iteration (where, with no other information, we may initially set $x_0 = 0$), and y_i is a list of $O(1)$ auxiliary n-vectors computed on the i^{th} iteration within the same relative error.

SOME BASE ITERATIVE METHODS. In certain very simplistic iterative methods, including Jacobi, Gauss-Seidel, and SOR, we require no auxiliary vectors in y_i. Let $D = \text{diag}(A)$ be the $n \times n$ matrix containing the diagonals of A, and let L, U be the proper lower and upper triangular submatrix of A. For the *Jacobi iteration*, we define $x_{i+1} = D^{-1}((A - D)x_i + b)$, for the *Gauss-Seidel iteration*, we define $x_{i+1} = U^{-1}((D + L)x_i + b)$, and for the *SOR iteration*, we define $x_{i+1} = (D + \omega L)^{-1}(((1 - \omega)D - \omega U)x_i + \omega b)$, for some $0 < \omega < 1$.

In the conjugate gradient and Chebyshev semi-iterative iterations, y_i consists of two auxiliary vectors q_i, p_i. For the *conjugate gradient iteration* we initialize $x_0 = 0$, $q_1 = p_0 = b$, and we define $x_{i+1} = x_i + c_i q_i$, where $p_{i+1} = p_i - c_i A q_i$, $c_i = (p_i^T p_i / q_i^T A q_i)$, and for $i \geq 2$, $q_{i+1} = p_i + (p_i^T p_i / p_{i-1}^T p_{i-1}) q_i$

The *Chebyshev semi-iterative iteration* is also a three term recurrence with similar convergence bounds; see [6, p. 513].

ERRORS IN BASE ITERATIVE METHODS. The (absolute) *error* on the i^{th} iteration is defined to be

$$\epsilon_i \|b\|_M = \|Ax_i - b\|_M,$$

where M is a fixed matrix depending on the base iterative method (see [6]), and the *relative error* is ϵ_i. Each such base iterative improvement procedure IMPROVE (A, b, x_i, y_i) requires an inner product of the input matrix A times an n-vector. Assuming A has m nonzeros, this costs $O(m)$ steps per iteration (again, see [6]). The *improved relative error ratio* per iteration is $\epsilon_i/\epsilon_{i-1}$. Assuming that the input matrix A is SPD, $\epsilon_i/\epsilon_{i-1} \leq 1 - 1/\kappa(A)^\beta$, where β depends on the base iterative method (for example, $\beta = 1$ for Jacobi, Gauss-Seidel, and SOR, and $\beta = 1/2$ for conjugate gradient and Chebyshev semi-iterative methods; see [6]). The relative error of these base iterative methods is reduced geometrically: $\epsilon_i \leq 2(1 - (1/\kappa(A)^\beta))^i$. Thus, we have Proposition 2.1.

PROPOSITION 2.1. *Assuming A is SPD, these base iterative methods converge to an ϵ-solution in $O(\kappa(A)^\beta \log(1/\epsilon))$ iterations.*

Note that these base iterative methods provide no bits of accuracy until $\Omega(\kappa(A)^\beta)$ iterations; and thus, their performance is strongly dependent on the condition number $\kappa(A)$ of the input matrix. Hence, a key approach to speeding up such an iterative method is to decrease the condition number $\kappa(A)$.

PRECONDITIONED ITERATIVE METHODS. The number of iterations of a base iterative method can be reduced by the computation of a *preconditioner matrix* B such that $\kappa(B^{-1}A)$ is small. The *preconditioned linear system* $A'x = b'$ derived from the original linear system $Ax = b$ with *preconditioned matrix* $A' = B^{-1}A$ and $b' = B^{-1}b$, requires $O(\kappa(B^{-1}A)^\beta \log(1/\epsilon'))$ iterations to converge to an ϵ'-solution.

PROPOSITION 2.2. *If we compute an ϵ'-solution $\tilde{x}$ of the preconditioned linear system such that $\|A'\tilde{x} - b'\| \leq \epsilon'\|b'\|$, where $\epsilon' = \epsilon/(\kappa(A')\kappa(A))$ then $\|A\tilde{x} - b\| \leq \epsilon\|b\|$.*

PROOF. Note that $\kappa(BA^{-1}) = \kappa(A^{-1}B) = \kappa(A')$, so $\kappa(B) = \kappa(BA^{-1}A) \leq \kappa(BA^{-1})\kappa(A) = \kappa(A')\kappa(A)$, hence, $\epsilon' \leq \epsilon/\kappa(B)$. Then, by definition of A' and b', we have $\|A\tilde{x} - b\| \leq \|B(A'\tilde{x} - b')\| \leq \|B\|\|A'\tilde{x} - b'\| \leq \|B\|(\epsilon'\|b'\|) \leq \epsilon\|b\|$ since $\|B\|(\epsilon'\|b'\|) \leq \|B\|\epsilon'\|B^{-1}\|\|b\| = \kappa(B)\epsilon'\|b\| \leq \epsilon\|b\|$. ∎

2.2. Preconditioned Iterations

The preconditioned iterative (PI) procedure is

$$(x_{i+1}, y_{i+1}) = \text{IMPROVE}\,(A', b', x_i, y_i),$$

where x_i is the approximation to the solution on the i^{th} iteration (where, again, generally initially $x_0 = 0$), and IMPROVE is a base iterative method.

For example, for the *preconditioned conjugate gradient iteration* with preconditioned matrix $A' = B^{-1}A$, we define $x_{i+1} = x_i + c_i q_i$, where $p_{i+1} = p_i - c_i A' q_i$, $c_i = (p_i^{\mathrm{T}} p_i / q_i^{\mathrm{T}} A' q_i)$, and for $i \geq 2$, $q_{i+1} = p_i + (p_i^{\mathrm{T}} p_i / p_{i-1}^{\mathrm{T}} p_{i-1}) q_i$. The relative error of the PI procedure is $\epsilon_i' = \|A'x_i - b'\|_M / \|b'\|_M$, (where again M is a fixed matrix depending on the iterative method) and so the relative error is reduced geometrically $\epsilon_i' \leq 2(1 - 1/\kappa(A')^{\beta})^i$, where $\kappa(A')$ is the condition number of A' and β again depends on the base iterative method. Again, y_i is a list of $O(1)$ auxiliary vectors computed on the i^{th} iteration within the same relative error. It is important to note that the preconditioned matrix $A' = B^{-1}A$ need not be explicitly computed. Nevertheless, the base iterative improvement procedure IMPROVE (A', b', x_i, y_i) requires an inner product of the preconditioned matrix $A' = B^{-1}A$ times an n-vector vector in y_i. These inner products can be done first by a multiplication by A, resulting in an n-vector, say, r_i, and then a multiplication by B^{-1}. Assuming the input matrix A has m nonzero entries, this inner product cost $O(m)$ steps per iteration.

2.3. Approximate Preconditioned Iterations

The computation $z_i = B^{-1} r_i$ at first appears to imply the need for an exact solution z_i to the *induced linear system* $Bz_i = r_i$. However, that is not necessary. The induced linear system $Bz_i = r_i$ need not be computed exactly. Instead, let an *approximate PI algorithm* be a PI method exactly as detailed above. $(x_{i+1}, y_{i+1}) = \text{IMPROVE}(A', b', x_i, y_i)$ (utilizing one of the previously listed base iterative methods), modified so that we *use an approximation* $\tilde{b}'$ to b' with relative error $O(\epsilon/\kappa(B))$ and we *compute an approximate solution* $\tilde{z}_i$ to the induced linear system $Bz_i = r_i$ with relative error $\epsilon_{1,i} = O(\epsilon_i'/\kappa(B))$. (Note that an approximate PI is often used by multigrid solvers, except that we restart the iteration of the $\tilde{z}_i$ from 0, rather than from $\tilde{z}_{i-1}$, due to the possible near-orthogonality of $\tilde{z}_i$ and $\tilde{z}_{i-1}$.) It is well known (e.g., see [6]) that the resulting iterations still reduce the relative error geometrically, as $\epsilon_i' \leq O(1 - (1/\kappa(A')^{\beta}))^i$.

WORK BOUNDS. Let W_B be the work to do this computation of an approximate solution $\tilde{z}_i$ with relative error ϵ_i'. Let P_B be the work to construct the preconditioner matrix B. The error bounds given by Proposition 2.1 on the number of iterations require that the preconditioned matrix A' be SPD. (If this is not so, we may detect a violation of the error bounds or exceed the required number of iterations. If $A' = B^{-1}A$ is not SPD, we instead redefine the preconditioned matrix to be the *normal form preconditioned matrix* $A' = (B^{-1}A)^{\mathrm{T}}(B^{-1}A)$. We have noted above that any such normal form is SPD. We will bound the condition number of the normal form preconditioned matrix by $\kappa(A') = \kappa((B^{-1}A)^{\mathrm{T}}(B^{-1}A)) \leq \kappa((B^{-1}A)^{\mathrm{T}})\kappa(B^{-1}A)$. (Each iteration may now require two approximate solutions of linear systems with matrix B, but the work per iteration is asymptotically the same.) In general, the work of an approximate PI algorithm depends on W_B, P_B, and on the condition $\kappa(A')$ of the preconditioned matrix A'. The total precomputation and initialization work for an approximate PI algorithm in either case is P_B.

By Proposition 2.1, we have Proposition 2.3.

PROPOSITION 2.3. *If A' is SPD, an ϵ'-solution to the preconditioned linear system $A'x = b'$ can be computed in $O(\kappa(A')^{\beta} \log(1/\epsilon'))$ iterations.*

By Proposition 2.2, we have the following.

PROPOSITION 2.4. *If A' is SPD, an ϵ'-solution to the preconditioned linear system $A'x = b'$ provides an ϵ-solution to the given linear system $Ax = b$, with $O(\kappa(A')^{\beta} \log(\kappa(A')\kappa(A)/\epsilon))$ iterations.*

The work of each iteration is $O(m + W_B)$ arithmetic steps, assuming the input matrix A has m nonzeros. Hereafter, we assume that we use a base iterative method such as conjugate gradient or the Chebyshev semi-iterative method.

LEMMA 2.1. *Assuming that the preconditioned matrix A' is SPD, by use of a base iterative method such as conjugate hradient or the Chebyshev semi-iterative method, with $\beta = 1/2$ the total work for an approximate PI algorithm is $P_B + O(m + W_B)\sqrt{\kappa(A')}\log(\kappa(A')\kappa(A)/\epsilon)$, to compute an ϵ-solution to the given linear system $Ax = b$.*

3. BOUNDING PRECONDITIONER CONDITION BY SUPPORT MAPPINGS

In the following, we define a *network* to be a graph with each edge e labeled by a nonnegative real number *weight* $w(e)$. Given two networks G, G' with the same node set, a *support mapping* from G to G' provides a mapping f from each edge $e = (i, j)$ of G to a path $f(e)$ from i to j in G'. The edges of the path $f(e)$ in G' *supports edge* e in G. The *supported weight* of edge e' of G' is $\sum_{e \in E(G), e' \in f(e)} w(e)$, that is, the supported weight of e' is the sum of the weights mapped to paths on G' containing edge e'. The *dilation* of edge e of G is the number of edges of the longest path $f(e)$ appearing in the support mapping. The *congestion* of an edge e' of G' is the ratio of the supported weight of e' to the weight of e' in G'. We define the *support number* of the support mapping to be maximum, for any edge e of G and any edge e' of G' supported by e, of the product of the dilation of e times the congestion of e'. A *weight partitioning* of a network G consists of a set of networks $G_1, \dots, G_k$ such that the weight of each edge e of G is the sum of the weights of e in the networks $G_1, \dots, G_k$ and there is no edge appearing in networks $G_1, \dots, G_k$ that does not appear in G. (Note that the nodes and edges of G may be repeated in the networks $G_1, \dots, G_k$.) We define the *support* $\sigma(G, G')$ of network G by G' to be the minimum number $\sigma > 0$ which upper bounds the support number of each $f_1, \dots, f_k$ for some weight partitioning $G_1, \dots, G_k$ of network G and some weight partitioning $G'_1, \dots, G'_k$ of network G', for some support mappings f_i from G_i to G'_i, for $i = 1, \dots, k$. Recall that the *sparsity graph* $G(A)$ of a real $n \times n$ symmetric matrix $A = \{a_{i,j}\}$ is a graph (V, E) with node set $V = \{1, \dots, n\}$ and edge set $E = \{(i, j) \mid a_{i,j} \neq 0\}$. We will augment the sparsity graph with the nonnegative real value $|a_{i,j}|$. For compactness of notation, we will intentionally confuse notation and also denote the resulting network $G(A)$. Given two matrices, A, B of the same size, we define the *support* $\sigma(A, B)$ of A by B to be $\sigma(G(A), G(B))$. Recall that a Laplacian matrix is a real symmetric DD matrix where each off-diagonal element is nonpositive. Laplacian matrices are known to be equivalent to resistive matrices associated with resistive networks of electrical circuits [39,40], and have been investigated in the determination of the expected length of a random walk in resistive network [40]. Let $B \leq A$ if A, B have the same size, say $n \times n$, and for each i, j we have $|b_{i,j}| \leq |a_{i,j}|$. In this case, note that $\sigma(B, A) = 1$. The following Lemma 3.1 is a known result, which seems to be first due to Vaidya [1].

LEMMA 3.1. *For any Laplacian $n \times n$ matrices $A, B, \kappa(B^{-1}A) \leq O(\sigma(A, B)\sigma(B, A))$, and if $B \leq A$, then $\kappa(B^{-1}A) \leq O(\sigma(A, B))$.*

A full proof of Lemma 3.1, and further enlightening discussions of support mappings and resistive networks, appears in the recent Ph.D. thesis of Gremban [4] (his statement of Lemma 3.1 appears without reference to the prior work of Vaidya [1], but may be considered to be an improvement, since it does not require constant factor, due to a redefinition of the mapping σ). The previous results of Vaidya [4] bounding condition numbers were generally restricted to Laplacian matrices.

Fix an $n \times n$ real symmetric matrix A. Let A^+ be the $n \times n$ matrix derived from A by setting to 0 all negative off-diagonal elements. Let A^- be the $n \times n$ matrix derived from A by setting to 0 all positive off-diagonal elements. Let $D(A)$ be the $n \times n$ matrix consisting of the diagonal elements of A, where $D(A)$ has 0 value at the off-diagonal elements. Define the $2n \times 2n$ matrix

$$\breve{A} = \begin{bmatrix} D(A) + A^- & -A^+ \\ -A^+ & D(A) + A^- \end{bmatrix}.$$

Then $\breve{A}\left[\begin{smallmatrix} x \\ -x \end{smallmatrix}\right] = \left[\begin{smallmatrix} Ax \\ -Ax \end{smallmatrix}\right]$ for any n-element vector x, since $A = D(A) + A^+ + A^-$.

Gremban has shown Proposition 3.1 [4, Lemma 7.2].

PROPOSITION 3.1. *$\breve{A}$ has eigenvalue λ if A has eigenvalue λ.*

The following provides an extension of Lemma 3.1 to symmetric DD matrices.

LEMMA 3.2. *For any symmetric DD matrices A, B, where $B \le A$, $\kappa(B^{-1}A) \le O(\sigma(A,B))$, and $\kappa((B^{-1}A)^{\mathrm{T}}(B^{-1}A)) \le O(\sigma(A,B)^2)$.*

PROOF. By Proposition 3.1, $\breve{A}$ has all the eigenvalues of A. Note that each off-diagonal element of $\breve{A}$ is nonpositive. If A is symmetric and DD, then $\breve{A}$ is a Laplacian matrix.

Our proof of Lemma 3.2 uses an interesting argument, where we show by a graph embedding argument using support mappings, that $\sigma(\breve{A}, \breve{B})$ can be upper bounded by $O(\sigma(A,B))$, for two Laplacian matrices $\breve{A}, \breve{B}$ derived from A, B, with $\kappa(B^{-1}A) \le \kappa(\breve{B}^{-1}\breve{A})$, and then we apply Lemma 3.1, thus bounding $\kappa(\breve{B}^{-1}\breve{A}) \le O(\sigma(\breve{A}, \breve{B}))$. This implies $\kappa(B^{-1}A) \le \kappa(\breve{B}^{-1}\breve{A}) \le O(\sigma(\breve{A}, \breve{B})) \le O(\sigma(A,B))$, as claimed.

Let A, B, be nonsingular symmetric DD matrices where $B \le A$. Let $\breve{A}$, $\breve{B}$ be the Laplacian matrices derived from A, B as defined above. Then $B^{-1}Ax = \lambda x$, if $\breve{B}^{-1}\breve{A}\left[\begin{smallmatrix} x \\ -x \end{smallmatrix}\right] = \lambda\left[\begin{smallmatrix} x \\ -x \end{smallmatrix}\right]$. Hence, $\breve{A}$ has eigenvalue λ if A has eigenvalue λ. Thus we conclude $\breve{B}^{-1}\breve{A}$ has all the eigenvalues of $B^{-1}A$, so $\kappa(B^{-1}A) \le \kappa(\breve{B}^{-1}\breve{A})$.

Fix a path $p = e_1, \dots, e_j$ in $G(A)$ where $e_i = (u_{i-1}, u_i)$ for $i = 1, \dots, j$. Then we can construct a path $\breve{p} = \breve{e}_1, \dots, \breve{e}_j$ in $G(\breve{A})$ starting at node u_0 with the same edge weight magnitudes: the weight of e_i in $G(A)$ has the same magnitude as the weight of $\breve{e}_i$ in $G(\breve{A})$, for $i = 1, \dots, j$. Let P_i be the number of positive entries, $A_{e_{i'}} > 0$, that are indexed by the edges $e_{i'}, 1 \le i' < i$, preceding edge e_i on the path p. To construct $\breve{p}$, we let for $i = 1, \dots, j$ its edges be $\breve{e}_i = (\breve{u}_{i-1}, \breve{u}_i)$ where $\breve{u}_i = u_i$ if P_i is even, and otherwise $\breve{u}_i = u_i + n$. We can also similarly construct a path $\breve{p}' = \breve{e}'_1, \dots, \breve{e}'_j$ in $G(\breve{A})$, with the same edge weight magnitudes, that starts instead at node $u_0 + n$ where for $i = 1, \dots, j$ its edges are $\breve{e}_i = (\breve{u}'_{i-1}, \breve{u}'_i)$, where $\breve{u}'_i = u_i + n$ if P_i is even, and otherwise $\breve{u}'_i = u_i$.

Consider a support mapping f from $G(A)$ to $G(B)$, with dilation (f) congestion $(f) = \sigma$. For each edge $e = (u,v)$ of $G(A)$, consider the path $p = f(e)$. If p has an even number of negative edges, then let $\breve{f}(u,v)$ be the path $\breve{p}$ in $G(\breve{B})$ defined above and let $\breve{f}(u+n, v+n)$ be the path $\breve{p}'$ also defined above. If p has an odd number of negative edges, then let $\breve{f}(u, v+n)$ be the path $\breve{p}$ in $G(\breve{B})$ defined above and let $\breve{f}(u+n, v)$ be the path $\breve{p}'$ also defined above. Thus, dilation$(\breve{f}) \le O($dilation$(f))$ and congestion$(\breve{f}) \le O($congestion$(f))$. Thus, we can construct a support mapping $\breve{f}$, from $G(\breve{A})$ to $G(\breve{B})$ with the product dilation$(\breve{f})$congestion$(\breve{f}) = O(\sigma)$. The support $\sigma(G(A), G(B))$ of network $G(A)$ by $G(B)$ is the maximum of the support number of each $f_1, \dots, f_k$ for some weight partitioning $G_1, \dots, G_k$ of network $G(A)$ and some weight partitioning $G'_1, \dots, G'_k$ of network $G(B)$, for some support mappings f_i from G_i to G'_i, for $i = 1, \dots, k$. We have just shown that for each $i = 1, \dots, k$ we can construct a support mapping $\breve{f}_i$ from $G(\breve{A})$ to $G(\breve{B})$ with dilation $(\breve{f}_i)$ congestion $(\breve{f}_i) \le$ dilation (f_i) congestion $(f_i) \le O(\sigma(G(A), G(B)))$. Thus, the support $\sigma(G(\breve{A}), G(\breve{B}))$ of network $G(\breve{A})$ by $G(\breve{B})$ is $O(\sigma(A,B))$. By Lemma 3.1, $\kappa(\breve{B}^{-1}\breve{A}) \le O(\sigma(\breve{A}, \breve{B}))$. Hence, $\kappa(B^{-1}A) = \kappa(\breve{B}^{-1}\breve{A}) \le O(\sigma(\breve{A}, \breve{B})) \le O(\sigma(A,B))$.

Since A, B are symmetric, $(B^{-1}A)^{\mathrm{T}} = A^{\mathrm{T}}(B^{-1})^{\mathrm{T}} = AB^{-1}$, so $\kappa((B^{-1}A)^{\mathrm{T}}) = \kappa(AB^{-1})$. The proof techniques of Lemma 3.1 also imply $\kappa(AB^{-1}) \le O(\sigma(A,B))$, and hence, we have $\kappa((B^{-1}A)^{\mathrm{T}}(B^{-1}A)) \le O(\sigma(A,B)^2)$. ∎

4. IMPROVED ALGORITHMS FOR APPROXIMATE SOLUTION OF SPARSE LINEAR SYSTEMS

GOOD PRECONDITIONERS. Given a symmetric DD matrix A, our main goal is to find a preconditioner B, where

(i) the resulting preconditioned matrix A' is SPD and has $O(1)$ condition number,
(ii) where B is constructed in $P_B = O(n)$ work,
(iii) the work W_B to compute an approximate solution of a linear system over B is $O(n)$.

If we can find a such a preconditioner, the resulting approximate PI algorithms have linear total work to find an ϵ-solution to the given linear system. (Note: in the following, we first let the preconditioned matrix be $A' = B^{-1}A$, but if the iterations do not converge as stated, then we conclude $B^{-1}A$ is not SPD, so we instead redefine $A' = (B^{-1}A)^{\top}(B^{-1}A)$ which is always SPD.)

OUR RESULTS. We show that we can construct good preconditioners for a number of cases, including the important case of grid graphs with nonzero entries of bounded magnitude (here we use multigrid preconditioners). While we certainly build on the work of Vaidya [1] and Gremban *et al.* [2–4], we also improve on these previous techniques:

(i) our condition number bounds on the preconditioned matrices are smaller than those of [1], which grow as n^{γ}, for $\gamma > 0$, and those of [2–4], which are $O(n^{1/d})$,
(ii) our arithmetic bounds are smaller than those of [1] which grow as $n^{1+\gamma}$, for $\gamma > 0$, and those of [2–4] which grow as $n^{1.5}$ in many cases (for example 2D grid graphs).

See further comments on the relation with previous work of [1] just after the statement of Corollaries 4.3 and 4.4.

4.1. Our Results for μ-Bounded Matrices

We have a number of results for matrices where the off-diagonals elements have bounded magnitude. The results given by Corollaries 4.1 and 4.2 are the best known to date for solution of certain systems of linear equations derived by higher dimensional PDEs with bounded coefficients. Corollary 4.1 has wide application to the solution of such PDEs, which are often discretized as regular d-dimensional grids with bounded coefficients (see also the description in Section 1.8 of previous work). The further result of Corollary 4.2 has application to the solution of stochastic PDEs discretized as irregular d-dimensional grids with small separators. Let a matrix A be *μ-bounded* if the nonzero off-diagonal entries of A have magnitude at most a factor μ more than the smallest magnitude off-diagonal nonzero entry. A is thus *$O(1)$-bounded* if it is μ-bounded for $\mu = O(1)$. Note that we can assume without loss of generality (by use of a diagonal preconditioner matrix $\mathrm{diag}(a_{11}^{-1}, \dots, a_{nn}^{-1})$) that each diagonal element of a μ-bounded A is 1. Let B be the matrix derived from A by substituting for each nonzero off-diagonal element the smallest magnitude nonzero off-diagonal of A, and with the same diagonal elements as A, so B is DD assuming A is DD. The preconditioner matrix B is thus computed in $P_B(n) \le O(n)$ work.

PROPOSITION 4.1. *$\kappa(B^{-1}A) \le O(\mu)$ and $\kappa((B^{-1}A)^{\top}(B^{-1}A)) \le O(\mu^2)$.*

PROOF. Recall matrix A is *μ-bounded* if the nonzero off-diagonal entries of A have magnitude at most a factor μ more than the smallest magnitude nonzero off-diagonal entry, and we can assume (by premultiplication of a diagonal matrix) that all the diagonal elements have the same value. Each edge of $G(A)$ can be supported by the same edge of $G(B)$ with dilation 1 and congestion at most μ. Thus, the support number $\sigma(A, B)$ is the product of these bounds on dilation and congestion, namely μ. By Lemma 3.2, it follows that $\kappa(B^{-1}A) \le O(\mu)$. Furthermore, by Lemma 3.2, $\kappa((B^{-1}A)^{\top}(B^{-1}A)) \le O(\sigma(A, B)^2) \le O(\mu^2)$. ∎

In the following, fix $\pi = \log(\kappa(A)/\epsilon)$. By the use of a multigrid preconditioner, we obtain the following Theorem.

THEOREM 4.1. *Given a symmetric DD and μ-bounded matrix A of size $n \times n$ with a sparsity graph which is an n node regular d-dimensional grid, for constant d, we can compute a preconditioner matrix B in $P_B = O(n)$ work, resulting in the preconditioned matrix A' of condition number $\kappa(A') \le O(1)$, and such that $W_B = O(n\pi)$.*

PROOF. Fix a symmetric DD $O(1)$-bounded $n \times n$ matrix A, with a sparsity graph which is n node regular d-dimensional grid, for any constant d. We use Proposition 4.1 to obtain the preconditioner B. If $B^{-1}A$ is PD, we let $A' = B^{-1}A$ be the preconditioned matrix of condition number $\kappa(A') \le O(\mu)$, and otherwise let $A' = (B^{-1}A)^{\mathrm{T}}(B^{-1}A)$ be the preconditioned matrix of condition number $\kappa(A') \le O(\mu^2)$. Assuming $\mu = O(1)$, then in either case $\kappa(A') \le O(1)$. We use multigrid to compute (via approximate PI), in work $W_B(n)$, an approximate solution to a linear system with matrix B from a given approximate solution with relative error $O(\epsilon_i'/\kappa(B))$. By construction, B is symmetric DD with a sparsity graph which is a regular d-dimensional grid, for any constant d, and where every off-diagonal element has the same fixed value. We have defined $\epsilon' = \epsilon/(\kappa(A')\kappa(A))$, so $1/\epsilon' \le (\kappa(A')\kappa(A))/\epsilon$, and hence, $1/\epsilon' \le (\mu\kappa(A))/\epsilon$, if $B^{-1}A$ is PD, and otherwise $1/\epsilon' \le (\mu^2\kappa(A))/\epsilon$. It is known [25–27] that multigrid methods obtain this relative error in a number of multigrid iterations bounded by $O(\sqrt{\kappa(A')}\log(\kappa(A')/\epsilon_i'))$, which is $\le O(\sqrt{\mu}\log(\mu/\epsilon')) \le O(\sqrt{\mu}\log(\mu\kappa(A)/\epsilon))$, if $B^{-1}A$ is PD, and otherwise we have $\le O(\sqrt{\kappa(A')}\log(\kappa(A')/\epsilon_i')) \le O(\mu\log(\mu/\epsilon')) \le O(\mu\log(\mu\kappa(A)/\epsilon))$ multigrid iterations. Since each multigrid iteration takes $O(n)$ time, $W_B(n) \le O(n\sqrt{\mu}\log(\mu\kappa(A)/\epsilon))$, if $B^{-1}A$ is PD, and otherwise $W_B(n) \le O(n\mu\log(\mu\kappa(A)/\epsilon))$. In either case, $W_B(n) \le O(n\log(\kappa(A)/\epsilon)) = O(n\pi)$, for $\mu = O(1)$. This completes the proof of Theorem 4.1. ∎

COROLLARY 4.1. *The arithmetic work is $O(n\pi^2)$ to compute an ϵ-solution to a linear system with an $O(1)$-bounded symmetric DD matrix A of size $n \times n$ with a sparsity graph which is an n node regular d-dimensional grid, for any constant d.*

PROOF. By the upper bound on $W_B(n)$ given by the above proof of Theorem 4.1, Lemma 2.1 implies that the approximate PI algorithm costs work in this case,

$$\begin{aligned}
&P_B + O(O(m) + W_B)\sqrt{\kappa(A')}\log\left(\frac{\kappa(A')\kappa(A)}{\epsilon}\right) \\
&\le \left(O(n) + On\sqrt{\mu}\log\left(\frac{\mu\kappa(A)}{\epsilon}\right)\right) O\left(\sqrt{\mu}\log\left(\frac{\mu\kappa(A)}{\epsilon}\right)\right) \\
&\le O\left(n\mu\log^2\log\left(\frac{\mu\kappa(A)}{\epsilon}\right)\right),
\end{aligned}$$

if $B^{-1}A$ is PD, and else

$$\begin{aligned}
&P_B + O(O(m) + W_B)\sqrt{\kappa(A')}\log\left(\frac{\kappa(A')\kappa(A)}{\epsilon}\right) \\
&\le \left(O(n) + On\mu\log\left(\frac{\mu\kappa(A)}{\epsilon}\right)\right) O\left(\mu\log\left(\frac{\mu\kappa(A)}{\epsilon}\right)\right) \\
&\le O\left(n\mu^2\log^2\left(\frac{\mu\kappa(A)}{\epsilon}\right)\right).
\end{aligned}$$

In either case, the total work is $\le O(n\log^2(\kappa(A)/\epsilon)) = O(n\pi^2)$, for $\mu = O(1)$. Thus, Theorem 4.1 immediately implies Corollary 4.1. ∎

Also, we have some further results for stochastic matrices. A matrix is *stochastic* if all diagonal entries are 1 and all off-diagonal entries are negative and their magnitudes sum to 1. A matrix is *weakly stochastic* if all diagonal entries are 1 and for each row, the magnitudes of the off-diagonal entries of the row sum to 1.

THEOREM 4.2. *Fix an n node sparse $s(n)$-separable graph G. Also, fix a certain preconditioner matrix B with sparsity graph G and with a precomputed sparse $LL^\top$-factorization. Then given any SPD DD, weakly stochastic, $O(1)$-bounded matrix A of size $n \times n$ with sparsity graph $G(A) = G$ we can compute a preconditioned matrix A' with condition number $\kappa(A') \leq O(1)$, where $P_B = O(n)$ and $W_B = O(n + s(n)^2)$.*

PROOF. As in the proof of Theorem 4.1, we use Proposition 4.1 to obtain the preconditioner B. Since we have assumed A is SPD, the preconditioned matrix $A' = B^{-1}A$ is SPD and has condition number $\kappa(A') \leq O(\mu) \leq O(1)$, for $\mu = O(1)$. Use of the precomputed sparse $LL^\top$-factorization of B [10] allows us to solve a linear system $Bz = r$, for any n-vector r, in work $W_B = O(n + s(n)^2)$. Since the $LL^\top$-factorization of B is fixed and precomputated (without cost), the initialization cost is $P_B = O(n)$. ∎

COROLLARY 4.2. *Fix a sparse $s(n)$-separable graph G. The arithmetic work is $O((n + s(n)^2)\pi)$ to compute an ϵ-solution to a linear system with an $O(1)$-bounded SPD DD weakly stochastic matrix A of size $n \times n$. Thus, the arithmetic work is $O(n\pi)$, if $s(n) \leq O(\sqrt{n})$.*

PROOF. Lemma 2.1 implies that the preconditioned iteration algorithm has work in this case

$$P_B + O(O(m) + W_B)\sqrt{\kappa(A')}\log\left(\frac{\kappa(A')\kappa(A)}{\epsilon}\right) \leq O\left((n + s(n)^2)\right)$$
$$\log\left(\frac{\kappa(A)}{\epsilon}\right) = O\left(\left(n + s(n)^2\right)\pi\right). \quad ∎$$

4.2. Our Results for Sparse Matrices with Unbounded Magnitude

In Section 5 we will prove Theorem 4.3.

THEOREM 4.3. *Given a sparse symmetric nonsingular DD matrix A, we can compute a preconditioner matrix B and its Cholesky $LL^\top$-factorization in $P_B = O(n + (n')^\omega)$ work, resulting in the SPD preconditioned matrix $A' = B^{-1}A$, of condition number $\kappa(A') \leq O((n/n')^2)$, and $W_B \leq O(n + (n')^2)$.*

Let $\pi_1 = \log(n\kappa(A)/\epsilon) = \pi + \log n$. Theorem 4.3 implies that use of a preconditioned iteration (in this case, without need of approximate PI) applied to such a sparse matrix A, requires at most $\sqrt{\kappa(A')}\log(\kappa(A')\kappa(A)/\epsilon) \leq O(n/n')\pi_1$, iterations, each with work $O(n)$.

Setting $n' = (n^2\pi_1)^{1/\omega+1}$, and applying Lemma 2.1, Corollary 4.3 follows.

COROLLARY 4.3. *We can compute an ϵ-solution to a linear system with a sparse symmetric DD matrix A of size $n \times n$ in arithmetic work*

$$O(n(\pi + \log n))^{\omega/\omega+1}.$$

Note that our work is $\leq O(n\pi_1)^{1.5}$ for $\omega = 3$ and is $\leq O(n\pi_1)^{1.408}$ for $\omega = 2.376$ [5]. In contrast, Vaidya [1] gave a bound of $O(n\pi_1)^{1.75}$ for this case.

Let $\rho = 1 - 2/\omega$ (note that $\rho = 1/3$ for $\omega = 3$ and $\rho = 0.15825$ for $\omega = 2.376$ [5]). By the recursive use of preconditioners, we will prove Theorem 4.4 in Section 5.

THEOREM 4.4. *Given a symmetric DD matrix A of size $n \times n$, with a sparsity graph in a family of graphs with constant size forbidden graph minors (for example, planar graphs), let $\phi(n)$ be any increasing function. We can compute a preconditioner matrix B in $P_B = O(n)$ work, resulting in the preconditioned matrix A', of condition number $\kappa(A') = O(\pi_1^{2\phi(n)})$, with $W_B = O(n^{1+(\rho/\phi(n)-O(1))}{\pi_1}^{O(1)})$.*

Lemma 2.1 and Theorem 4.4 imply the recursive PI algorithm requires work

$$\begin{aligned} P_B + O(O(m) + W_B)\sqrt{\kappa(A')}\pi_1 &\leq O\left(n^{1+\rho/\phi(n)-O(1)}\pi_1^{o(1)}\right) O\left(\pi_1^{\phi(n)+1}\right) \\ &= O\left(n\pi_1^{\rho\log n/(\log\pi_1)(\phi(n)-O(1))+\phi(n)+1+O(1)}\right). \end{aligned}$$

To minimize our work bounds, we set $\phi(n) = \sqrt{\rho \log n / \log \pi_1} + o(1)$, and obtain the work bound $O(n\pi_1^{1+2\sqrt{\rho \log n / \log \pi_1}+O(1)})$. Since $\pi_1 = \pi + \log n$, in the case that $\log \pi = \theta(\log n)$ (that is when both $\log \pi \le O(\log n)$ and $\log n \le O(\log \pi)$), then our work bound is

$$O\left(n\pi_1^{1+2\sqrt{\rho \log n / \log \pi_1}+o(1)}\right) = O\left(n(\pi + \log n)^{O(1)}\right).$$

In the case $\log n = O(\log \pi)$, our work bound is

$$O\left(n\pi_1^{1+2\sqrt{\rho \log n / \log \pi_1}+o(1)}\right) \le O\left(n(\pi + \log n)^{1+o(1)}\right).$$

Furthermore, for the case $\log \pi = O(\log n)$, our work bound is

$$\begin{aligned} O\left(n\pi_1^{1+2\sqrt{\rho \log n / \log \pi_1}+o(1)}\right) &= O\left(n^{1+2\sqrt{\log \pi_1 / \rho \log n}}\pi_1^{1+o(1)}\right) \\ &\le O(n(\pi + \log n))^{1+o(1)}. \end{aligned}$$

Summarizing our results, we have Corollary 4.4.

COROLLARY 4.4. *If A is a symmetric nonsingular DD matrix of size $n \times n$ with a sparsity graph in a family of graphs with constant size forbidden graph minors, the arithmetic work to compute an ϵ-solution of a linear system with matrix A is $O(n\pi_1^{1+2\sqrt{\rho \log n / \log \pi_1}+o(1)})$, which is*

- *$O(n(\pi + \log n)^{O(1)})$ in the special case that $\log \pi = \theta(\log n)$, and more generally is*
- *$O(n(\pi + \log n)^{1+o(1)})$ if $\log n = O(\log \pi)$, and*
- *$O(n(\pi + \log n))^{1+o(1)}$ if $\log \pi = O(\log n)$.*

NOTE. In comparison, previously Vaidya [1] gave for this case a bound of

$$O\left(n(\pi + \log n)\right)^{1+\gamma}, \qquad \text{for } \gamma > 0.$$

Also note that these iterative methods are much faster than direct methods (for example nested dissection) that compute a sparse LL^{T}-factorization; in contrast the work bound of nested dissection [10,11] is lower bounded as $\Omega(n^{\omega/2})$ for SPD matrices with planar sparsity graphs.

5. PROOF OF OUR RESULTS FOR SPARSE MATRICES WITH NO MAGNITUDE BOUNDS

5.1. Reduction from Sparse to Bounded Degree Matrices

Let A be a symmetric nonsingular DD (and thus SPD) $n \times n$ matrix with $m = O(n)$ nonzeros. The *degree* of a node of $G(A)$ is the number of adjacent nodes. Let the *degree* of A be the maximum degree of any node of $G(A)$.

PROPOSITION 5.1. *We can expand an $n \times n$ matrix A with m nonzeros into a matrix $\bar{A}$ of size $\bar{n} \times \bar{n}$ where $\bar{n} \le 2n$ such that the degree of $\bar{A}$ is $\le \max(3, m/n)$, and such that after Gaussian elimination of the added subtrees, the resulting linear system is equivalent to A.*

PROOF. Fix a sparse linear system $Ax = b$ with matrix $A = \{a_{i,j}\}$ of size $n \times n$ with m nonzero coefficients. We now describe a reduction to an equivalent linear system of size $2n \times 2n$ with a sparsity graph that has degree $\le \delta = \max(3, m/n)$. This reduction will done by expanding each vertex of degree $\delta' > \delta$ in the sparsity graph into a tree of at most $1 + \delta'/(\delta - 1)$ nodes of degree $\le \delta$ such that the resulting expanded linear system is equivalent to the original linear system after Gaussian elimination of all but the root of that tree. This expansion can be done in, at most, $\delta'/(\delta - 1)$ stages, where on each stage we decrease the degree by $\delta - 1$ and add a new variable and linear constraint to the linear system. A vertex of degree $\delta' > \delta$ will correspond to a variable, say x_k, with the linear constraint: $a_{k,k}x_k + \sum_{i=1}^{\delta'} a_{k,j_i}x_{j_i} = b_k$, where $a_{k,j_1}, \ldots, a_{k,j_{\delta'}}$ are the nonzero off-diagonal elements of the k^{th} row of A.

Choose some c such that if $|a_{k,k}| > \sum_{i=1}^{\delta'} |a_{k,j_i}|$, then $1 < c < |a_{k,k}|/\sum_{i=1}^{\delta'} |a_{k,j_i}|$ and else $c = 1$. Let us define $a_{k,n+1} = c\sum_{i=1}^{\delta} |a_{k,j_{\delta'-i-1}}|$, and let $a_{k,s} = 0$ for $s \neq n+1$. We will introduce the new variable x_{n+1} (x_{n+1} corresponds to a new interior node of the tree with children corresponding to variables $x_{\delta'-\delta+1}, \ldots, x_{\delta'}$) and replace the linear constraint

$$a_{k,k}x_k + \sum_{i=1}^{\delta'} a_{k,j_i}x_{j_i} = b_k$$

with two linear constraints: $a_{k,k}x_k + a_{k,n+1}x_{n+1} + \sum_{i=1}^{\delta'-\delta-1} a_{k,j_i}x_{j_i} = b_k$, and $x_{n+1} = (\sum_{i=1}^{\delta} a_{k,j_{\delta'-i-1}}x_{j_{\delta'-i-1}})/a_{k,n+1}$. Note that the resulting linear system is of size $(n+1) \times (n+1)$, has the same solution for the variables $x_1, \ldots, x_n$ as the original system, and on this stage we have decreased the degree of the sparsity graph to $\delta' - \delta + 1 = \delta' - (\delta - 1)$. If A is DD, then by definition $|a_{k,k}| > \sum_{i=1}^{\delta'} |a_{k,j_i}|$, so for $k \leq n$, $|a_{k,k}| \geq a_{k,n+1} + \sum_{i=1}^{\delta'-\delta-1} |a_{k,j_i}| \leq c\sum_{i=1}^{\delta'} |a_{k,j_i}| < 1$, and hence, the resulting linear system is also DD. Also, it is easy to verify that if A is symmetric and nonsingular, then the resulting linear system is also symmetric and nonsingular. After repeated stages at all applicable vertices, the resulting equivalent linear system has degree $\leq \delta$ and is of size at most $\bar{n} \times \bar{n}$, where $\bar{n} \leq n + 2m/\delta \leq 2n$. ∎

5.2. Maximum Spanning Tree Preconditioners

By Proposition 5.1, we will assume, without loss of generality, that the input matrix A has maximum degree not more than $\max(3, m/n)$. We will execute (and later improve upon) the following *Maximum Spanning Tree* (MST) *preconditioner procedure* due to Vaidya [1].

Input $n \times n$ symmetric nonsingular DD matrix A, and number $n' \leq n$.

[1] Compute a maximum spanning tree T of $G(A)$.

[2] Compute a partitioning of T into n' node-disjoint trees $T_1, \ldots, T_{n'}$ each of at most $2n/n'$ nodes. Let E' initially be the empty set.

[3] For each i, j where $1 \leq i < j \leq k$, if there is an edge (u, v) in $G(A)$ such that u is in T_i and v is in T_j, then add the maximum weight such edge (u, v) to E'.

[4] **Output** the preconditioner matrix B represented by the network $G(B)$ consisting of the union of E' and the edges of the trees $T_1, \ldots, T_{n'}$, and also all loop edges of $G(A)$ (corresponding to the diagonal elements of A).

Using the linear work minimum spanning tree algorithm of [41] the work bound to construct the preconditioner matrix B is $O(|E'| + m)$. By use of support mappings, we prove:

PROPOSITION 5.2. $\kappa(B^{-1}A) \leq O(nm/(n')^2)$.

PROOF. We shall show that there is a support mapping f from A to B with support number $\leq O(nm/(n')^2)$. Consider any edge $e = (u, v)$ of $G(A)$. If u, v are in the same tree T_i, then e can be supported by $f(e)$ = the tree path p of T_i from u to v of length at most $2n/n'$. Note that the weight of e is upper bounded by the weight of each edge of this tree path p, or else the tree T is not a maximum spanning tree (otherwise, we could have created a more weighty spanning tree by inserting instead the edge e). Otherwise, if u, v are on distinct trees T_i, T_j then let (u', v') be the edge of E' such that u' is in T_i and v' is in T_j, and let e be supported by a path $f(e)$ constructed as follows:

(i) the tree path p_1 of T_i from u to u' of length at most $2n/n'$,

(ii) the edge (u', v'),

(iii) the tree path p_2 of T_i from v' to v of length at most $2n/n'$.

Note that the weight of e is upper bounded by the weight of each edge of the first p_1 and last p_2 portions of this path or else the tree T is not a maximum spanning tree, and also by construction of E' the weight of e is upper bounded by the weight of edge (u', v'). Hence, the dilation of the support mapping is the length of path $f(e) = p_1(u', v')p_2$, which is at most $4n/n' + 1$. Since the degree of $G(A)$ is assumed to be at most $\max(3, m/n)$, and each tree T_i is of size at most $2n/n'$, and the weight of e is upper bounded by the weights of all the edges of the support path $f(e)$, the congestion is at most $\max(3, m/n)(2n/n') \leq O(m/n')$. Thus, the support number is the product of these bounds on dilation and congestion, namely $(4n/n' + 1)\max(3, m/n)(2n/n') \leq O(nm/(n')^2)$. By Lemma 3.2, it follows $\kappa(B^{-1}A) \leq O(\sigma(A, B)) \leq O(nm/(n')^2)$. ∎

The resulting B is nonsingular symmetric DD, since A is nonsingular symmetric DD. Hence, B is SPD. For each T_i, let T_i' be the forest derived from T_i by deleting the nodes that appear at the ends of edges in E'. To solve a linear system with preconditioner matrix B, we first compute a sparse partial $LL^\top$-factorization of B, (which exists since B is SPD) resulting in a reduced matrix B^*, where the sparsity graph $G(B^*)$ is derived from the sparsity graph $G(B)$ by collapsing each connected component (which will be a tree) of T_i' into a single node, for $i = 1, \ldots, k$. Note that this can be done in $O(n)$ work since the eliminated nodes are all within the trees $T_1, \ldots, T_{n'}$ (it is well known [10], that since a tree is $O(1)$-separable, computing a sparse $LL^\top$-factorization costs linear work). The resulting matrix B^* is of size $(c'n') \times (c'n')$, where c' is a constant multiple of the average degree of $G(A)$, and the number of edges of $G(B^*)$ is $O(|E'|)$.

5.3. The Sparse Preconditioned Linear Solver Without Recursion

We will assume a sparse SPD and DD matrix A, with $m = O(n)$ nonzeros. We apply Proposition 5.1 and so assume, w.l.o.g., that the input matrix A has constant maximum degree not more than $\max(3, m/n) = O(1)$. We execute the above MST preconditioner procedure. By Proposition 5.2, $\kappa(B^{-1}A) \leq O(nm/(n')^2) \leq O(n/(n')^2)$. The work bound to construct the preconditioner matrix B is $O(|E'| + m) \leq O((n')^2 + n)$. Using a technique similar to domain decomposition, we compute a sparse Cholesky $LL^\top$-factorization of B (where L is lower triangular) in two stages:

(a) first in $O(n)$ time, we compute an incomplete Cholesky $LL^\top$-factorization of B, resulting in a reduced matrix B^*, as previously described, then

(b) we compute a Cholesky $LL^\top$-factorization of the $O(n') \times O(n')$ matrix B^*.

The work to compute a Cholesky $LL^\top$-factorization of B^* is at most $O((n')^\omega)$ work. Thus, $P_B(n) \leq O(n + (n')^\omega)$. Once this Cholesky $LL^\top$-factorization of B^* has been computed, a linear system with matrix B^* can be solved in $W_B(n) \leq O(n + (n')^2)$ work, using the well known back-solving procedure.

5.4. The Sparse Preconditioned Linear Solver With Recursion

We will prove Theorem 4.4 by the use of *recursive* PI, as described in detail in Appendix 7. To construct the preconditioners, we make recursive application of the MST construction. (*Vaidya [1] previously proposed a similar use of recursion for a more restrictive class of matrices, but he was unable to reduce the condition number below $O(n^\gamma)$, for $\gamma > 0$.*) We assume that we are given a symmetric nonsingular DD (and hence also SPD) $n \times n$ matrix A, with a sparsity graph $G(A)$ contained in a family $\mathcal{F}$ of graphs with constant size forbidden graph minors. It is known [8,9] that $G(A)$ is sparse, with $m = O(n)$ edges. Observe, [8] that $\mathcal{F}$ is closed under edge contractions (that is, if we apply a sequence of edge contractions to a graph of $\mathcal{F}$, the result is a graph in $\mathcal{F}$). Recall the separator bound for planar graphs [42] is $O(\sqrt{n})$ and [9] provides $O(\sqrt{n})$ bounds on separators for any graph family with constant size forbidden graph minors.

We will construct a sequence of preconditioners $B_0 = B, \ldots, B_L$ for matrices $A_0 = A, A_1, \ldots, A_L$ such that B_ℓ is a preconditioner matrix for A_ℓ and for $\ell = 0, \ldots, L$ A_ℓ is an $n_\ell \times n_\ell$ matrix (which will be derived from $B_{\ell-1}$ by a partial Gaussian elimination of certain subtrees followed by the constant degree reduction) and $n_\ell \leq O(n_{\ell-1}/H(n))$, for a positive function $H(n) \geq 2$. The sparsity graphs of the matrices $A_1, \ldots, A_L$ of the recursive subproblems will be derived by edge contraction, and so will be also in $\mathcal{F}$. The symmetric, nonsingular DD properties (and hence also SPD) are inherited in matrices $A_1, \ldots, A_L$ of the recursive subproblems.

We can compute a sparse $LL^\top$-factorization of A_L by direct methods within work $s(n_L)^\omega \leq n_L^{\omega/2}$, since $s(n) = O(\sqrt{n})$ is the separator bound for the family $\mathcal{F}$ and we have defined n^ω to be the cost for $n \times n$ matrix multiplication. Hence, we can terminate the recursion when the work $n_L^{\omega/2}$ to compute a sparse $LL^\top$-factorization of A_L is $\leq n \leq$ the total work bounds for all the iterations. Thus, we terminate the recursion when we on that level $\ell = L$ where $n_L \leq n^{2/\omega}$.

Now we give the details of the recursive preconditioner procedure. Let c' be the constant defined in the MST preconditioner procedure. Following the definitions of Section 4.2, recall is $\phi(n) \geq 1$ any positive, monotonically increasing function, and $\pi_1 = \log(n\kappa(A)/\epsilon) = \pi + \log n$. Let $H(n) = (4c'\pi_1)^{\phi(n)}$.

We will execute the above MST preconditioner procedure, where we fix here $n' = n/H(n)$. We assume, w.l.o.g., the diagonal elements of A are positive. We will apply Proposition 5.1 and so assume, without loss of generality, that the input matrix A is sparse with constant maximum degree no larger than $\max(3, m/n) = O(1)$. (This reduction to constant degree will need to be done on each recursion.)

The MST preconditioner procedure given A yields a preconditioner matrix B. Let $A' = B^{-1}A$ be the preconditioned matrix. Since we have assumed that A is symmetric nonsingular DD (and so SPD), it follows that B is also. By Proposition 5.2, $\kappa(A') \leq O(\sigma(A, B)) \leq O(nm/(n')^2) \leq O(H^2(n))$. We first compute, as described above, a sparse partial $LL^\top$-factorization of B of the subtrees $T'_1, \ldots, T'_{n'}$ in $O(n)$ time, resulting in the reduced matrix B^* of size $(c'n') \times (c'n')$, where c' is the constant defined in the MST preconditioner procedure. We apply the constant degree reduction of Proposition 5.1 to B^*, resulting in the $n_1 \times n_1$ matrix A_1, where $n_1 \leq 4c'n'$. The number of nodes in the recursive problem is $n_1 \leq 4c'n' = 4c'(n/H(n))$. We will assume n is above a sufficiently large constant so that $H(n) \geq 8c'$ to insure $n_1 \leq n/2$. Since the resulting sparsity graph $G(A_1)$ is obtained by edge contraction from $G(B)$, A_1 is also symmetric nonsingular DD and so SPD, and $G(A_1)$ is also in the same family $\mathcal{F}$ as $G(A)$. Thus, $|E'| \leq O(n')$ and the number of edges of $G(A_1)$ is $O(|E'|) \leq O(n')$. We recursively compute an ϵ'-solution to a linear system with matrix A_1 by the recursive PI algorithm, making recursive application of this MST preconditioner procedure. The work bound to construct preconditioner matrix B is $O(|E'| + m) \leq O(n' + n) \leq O(n)$. Since precomputation of the preconditioner B requires work $O(n)$, the recurrence equation for precomputation of all the recursive preconditioners costs work $P_B(n) \leq P_{B_1}(n_1) + O(n) \leq P_B(n/2) + O(n)$ so $P_B(n) \leq O(n)$.

On recursion level ℓ, A'_ℓ is a $n_\ell \times n_\ell$ matrix where $n_\ell \leq 4c'(n_{\ell-1}/H(n_\ell))$. We have defined $H(n) = 4c'{\pi_1}^{\phi(n)}$, so $\log H(n) = \phi(n) \log 4c'\pi_1$. Also, recall that we terminate the recursion when we on that level $\ell = L$ where $n_L \leq n^{2/\omega}$. From this it follows that: $L \leq \rho \log_{H(n)-o(1)} n$, where $\rho = 1 - 2/\omega$, as defined in Theorem 4.4. Since $H(n) = 4c'{\pi_1}^{\phi(n)}$, so $\log H(n) = \phi(n) \log 4c'\pi_1$, we have

$$L \leq \rho \log_{H(n)-o(1)} n \leq \frac{\rho \log n}{(\phi(n) \log \pi_1) - o(1)}.$$

Thus, ${\pi_1}^L \leq {\pi_1}^{\rho \log n/(\phi(n) \log(4c'\pi_1)) - 4c'} \leq n^{\rho/\phi(n) - o(1)}$ and $L^L \leq {\pi_1}^{o(1)}$.

Hence, Proposition 5.3 follows.

PROPOSITION 5.3. *The number of levels of recursions is*

$$L \leq \rho \log_{H(n)-o(1)} n \leq \frac{\rho \log n}{(\phi(n) \log \pi_1) - o(1)}.$$

Also,

$$H(n)^L \leq n^{O(1)}, \qquad \pi^L \leq n^{\rho \log n/(\phi(n) \log \pi_1) - O(1)}, \qquad \text{and} \qquad L^L \leq \pi^{O(1)}.$$

PROPOSITION 5.4. *The recursive matrices A_ℓ and preconditioner matrices B_ℓ have condition number $\kappa(A_\ell), \kappa(B_\ell) \leq O(n^{O(1)}\kappa(A))$, and the recursive preconditioned matrices $A'_\ell = B_\ell^{-1}A_\ell$ have condition number $\leq O(H^2(n))$.*

PROOF. First, observe that if A_ℓ, B_ℓ are nonsingular, then

$$\begin{aligned}\kappa(B_\ell) &= \|B_\ell\| \, \|B_\ell^{-1}\| \\ &= \kappa\left(B_\ell^{-1}\right) \\ &\leq \kappa\left(B_\ell^{-1}A_\ell A_\ell^{-1}\right) \\ &\leq \kappa\left(B_\ell^{-1}A_\ell\right)\kappa\left(A_\ell^{-1}\right) \\ &\leq \kappa\left(B_\ell^{-1}A_\ell\right)\kappa(A_\ell),\end{aligned}$$

so

$$\kappa(B_\ell) \leq \kappa\left(B_\ell^{-1}A_\ell\right)\kappa(A_\ell).$$

Since

$$\begin{aligned}\kappa\left(B_\ell^{-1}A_\ell\right) &\leq O\left(H^2(n)\right), \\ \kappa(B_\ell) &\leq \kappa\left(B_\ell^{-1}A_\ell\right)\kappa(A_\ell) \\ &\leq O\left(H^2(n)\kappa(A_\ell)\right).\end{aligned}$$

By definition of A_ℓ,

$$\kappa(A_\ell) = \kappa(B_{\ell-1}) \leq O\left(H^2(n)\kappa(A_{\ell-1})\right) \leq O\left(H^{2\ell}(n)\kappa(A)\right).$$

By Proposition 5.3, $H(n)^L \leq n^{O(1)}$, so we can bound

$$\kappa(A_\ell) \leq O\left((H(n))^{2L}\kappa(A)\right) \leq O\left(n^{O(1)}\kappa(A)\right).$$

Similarly, $\kappa(B_\ell) \leq O(n^{O(1)}\kappa(A))$. By Proposition 5.2, the condition number bound for the recursive preconditioned matrices $A'_\ell = B_\ell^{-1}A_\ell$ is

$$\kappa\left(B_\ell^{-1}A_\ell\right) \leq O\left(\frac{n_\ell}{n_{\ell-1}}\right)^2 \leq O\left(H^2(n)\right). \qquad \blacksquare$$

To complete the proof of Theorem 4.4, we now provide an upper bound on $W_B(n)$, the work of the recursive computation of an approximate solution to a linear system with matrix B with relative error $\epsilon'/\kappa(B)$ using the recursively computed preconditioners.

By Propositions 2.4 and 7.1, we do at most

$$\tau_\ell = O\left(\sqrt{\kappa\left(A'_\ell\right)}\pi_1\ell\right) \leq O(H(n)\pi_1\ell)$$

iterations of the chosen base iterative method IMPROVE to achieve the required relative error.

Let $\tau_\ell = \max_i \tau_{\ell,i}$.

The total number of level ℓ preconditioned iterations we do is $\leq \Pi_{j=1}^{\ell}\tau_j$. If the level ℓ preconditioned matrix A'_ℓ is sparse and of size $n_\ell \times n_\ell$, the cost to do each iteration is $O(n_\ell)$, so Proposition 5.5 follows.

PROPOSITION 5.5. *The total cost to do all the level ℓ preconditioned iterations to the required relative error $\epsilon'_{\ell,i}$ is $O(n_\ell \Pi_{j=1}^{\ell} \tau_j)$.*

By Proposition 5.5, for each $\ell, 1 \le \ell \le L$, the cost to do all of the level ℓ preconditioned iterations is

$$\begin{aligned} O\left(n_\ell \Pi_{j=1}^{\ell} \tau_\ell\right) &\le O\left(n\left(\frac{4c'}{H(n)}\right)^{\ell}\right) O(H(n)\pi_0)^{\ell} \\ &\le O\left(n(4c'\pi_1\ell)^{\ell}\right) \\ &\le O\left(n\left(\pi_1 L\right)^{L}\right) \\ &\le O\left(n{\pi_1}^{L+O(1)}\right) \\ &\le O\left(n^{1+\rho/\phi(n)-O(1)}{\pi_1}^{o(1)}\right), \end{aligned}$$

since by Proposition 5.5, $\pi^L \le n^{\rho \log n/(\phi(n)\log \pi_1)-O(1)}$, and $L^L \le \pi^{O(1)}$.

6. CONCLUSION

We have presented preconditioned iterative methods for the solution of sparse linear systems. For a number of cases, we achieve near linear work bounds. Our methods, building on previous work, combine the use of algebraic and combinatorial methods for bounding condition number of preconditioners. It remains an open problem to reduce these work bounds strictly linear in important cases, for example for the case of linear systems with planar sparsity graphs.

7. APPENDIX: RECURSIVE PRECONDITIONED ITERATIONS

Here we give a detailed definition of the recursive PI algorithm of level $\ell \ge 0$. On level $\ell > 0$, we apply an iteration and recurse. Level 0 is the final level of recursion where we apply a direct method.

```
ALGORITHM PI_ℓ.
INPUT: sparse $n \times n$ matrix $A$, $n$-vector $b$, and relative error bound $\epsilon$, $0 < \epsilon < 1$.
[0] If $\ell = 0$ then apply a direct method to compute and output $A^{-1}b$ and exit.
[1] Construct preconditioner matrix $B$ for matrix $A$. Compute an approximation $\tilde{b}'$ to $b' = B^{-1}b$ by a recursive call to the algorithm $\mathrm{PI}_{\ell-1}$. This is to be done with relative error $\epsilon'/\kappa(B)$.
[2] INITIALIZE:
    [2.1] $i := 1$
    [2.2] $\tilde{x}_0 := 0$ and initialize the auxiliary vectors $\tilde{y}_0$ as prescribed by the base iteration method IMPROVE.
    [2.3] Also similarly initialize, for the induced linear system over $B$, its approximate solution $\tilde{z}_0 := 0$ and auxiliary vectors.
    [2.4] Let $\epsilon' = \epsilon/(\kappa(A')\kappa(A))$.
[3] WHILE $\|A'\tilde{x} - b'\| \le \epsilon'\|b'\|$ DO
    Apply one stage of the chosen base iterative method $(\tilde{x}_i, \tilde{y}_i) :=$ IMPROVE$(A', \tilde{b}', \tilde{x}_{i-1}, \tilde{y}_{i-1})$, which requires an inner product of the preconditioned matrix $A' = B^{-1}A$ times an $n$-vector, say $u_i$. We do not explicitly compute $A'$. Instead, we compute this inner product as follows.
    [3.1] Compute the inner product $r_i := Au_i$.
    [3.2] Apply a recursive call to the algorithm $\mathrm{PI}_{\ell-1}$ to compute an approximate solution $\tilde{z}_i$ of the induced linear system $Bz_i = r_i$, starting from the zero vector. The computation of $\tilde{z}_i$ is to be done with relative error $\epsilon_{1,i} = O(\epsilon'_i/\kappa(B))$, where $\epsilon'_i = \|A'\tilde{x} - b'\|/\|b'\|$.
    [3.3] i :=i+1
    OD
[4] $\tilde{x} := B\tilde{x}_i$
[5] OUTPUT: $\epsilon$-solution $\tilde{x}$ of the linear system $Ax = b$ such that $\|A\tilde{x} - b\| \le \epsilon\|b\|$.
```

The relative error at the i^{th} stage of the level 0 preconditioned iteration procedure PI_0 is at most

$$\epsilon'_i \|b'\|_M \le \|A'\tilde{x}_i - b'\|_M ,$$

(where M is a fixed matrix depending on the iterative method). Thus ϵ'_i is the relative error given by $\tilde{x}$, that is $\epsilon'_i = \|A'\tilde{x} - b'\|/\|b'\|$. The relative error at the i^{th} stage of the level ℓ preconditioned iteration procedure PI_ℓ is at most $\epsilon'_i = O(1 - 1/\kappa(A')^{\beta})^i$, where $\kappa(A')$ is the condition number of A' and β again depends on the base iterative method (we assume $\beta = 1/2$).

A Detailed Definition of the PI_1 Algorithm. As an example, we describe in detail the PI_1 algorithm. To **INITIALIZE,** we construct preconditioner matrix B_1 for matrix $A_1 = B$. The recursive preconditioned matrix is $A'_1 = B_1^{-1}A_1$. Also initialize $\tilde{z}'_0 := 0$ and initialize an auxiliary vector $\tilde{y}'_0$ as prescribed by the base iteration method IMPROVE.

We wish to $\epsilon_{1,i}$-solve the linear system $Bz_i = A_1 z_i = r_i$, where $\epsilon_{1,i} = \kappa(A_1)\epsilon'_i$. To do this, we $\epsilon'_{1,i}$-solve the preconditioned linear system $A'_1 z'_i = r'_i$, where $r'_i = B_1^{-1}r_i$, and by the proof of Proposition 2.4, it suffices that

$$\epsilon'_{1,i} = \frac{\epsilon_{1,i}}{(\kappa(A_1)\kappa(A'_1))} = \frac{\epsilon'_i}{(\kappa^2(A_1)\kappa(A'_1))}.$$

We use the zero vector as the starting iterate. (Note that we set $\tilde{z}'_{i,0}$ to 0, rather than to $\tilde{z}_{i-1}$, due to the possible near-orthogonality of $\tilde{z}_i$ and $\tilde{z}_{i-1}$.) We then apply to A'_1 the level 1 preconditioned matrix,

$$\tau_{1,i} = \sqrt{\kappa(A'_1)}\log\left(\frac{\kappa^2(A_1)\kappa(A'_1)\epsilon'_{1,0}}{\epsilon'_i}\right)$$

iterations of the chosen base iterative method IMPROVE.

```
[0] Let $\tilde{z}'_{i,0} = 0$, and initialize the auxiliary vectors $\tilde{y}'_{i,0}$ as prescribed by the base iteration
    method IMPROVE.
[1] FOR $j = 1, \ldots, \tau_{1,i}$ DO
        $(\tilde{z}'_{i,j}, \tilde{y}'_{i,j}) :=$ IMPROVE $(A'_1, \tilde{r}'_i, \tilde{z}'_{i,j-1}, \tilde{y}'_{i,j-1})$.
[2] Let $\tilde{z}'_i = \tilde{z}'_{i,\tau_{1,i}}$, and $\tilde{y}'_i = \tilde{y}'_{i,\tau_{1,i}}$.
```

This requires $O(\tau_{1,i})$ inner products of the preconditioned matrix $A'_1 = B_1^{-1}A_1$ times an n-vector. If this is the final level 0 of recursion, we compute these inner products exactly. Otherwise, we apply the preconditioned procedure recursively to the matrix B_1.

We now bound the relative error for the level 1 preconditioned iterations over matrix A'_1. The relative error is $\epsilon'_{1,i} = \|A'_1\tilde{z}'_i - r'_i\|_M/\|r'_i\|_M$, so if we use $\tilde{z}'_i$ to approximate the solution of the preconditioned linear system $A'_1 z'_i = r'_i$, the *error at the i^{th} stage of the level* 1 *preconditioned iteration* is $\epsilon'_{1,i}\|r'_i\|_M = \|A'_1\tilde{z}'_i - r'_i\|_M$ (where again M is a fixed matrix depending on the iterative method). We have to show that this relative error is at most $\epsilon'_{1,i} = \epsilon'_i/\kappa^2(A_1)\kappa(A'_1)$. Note that if we apply the conjugate gradient or Chebyshev semi-iterative method, the sequence of iterates $\tilde{z}'_0, \tilde{z}'_1, \ldots, \tilde{z}'_i$ defined above **do not** necessarily satisfy the usual properties of multistep iterates generated by these methods (since the r'_i vary depending on i). However, for a fixed i, the sequence of iterates $\tilde{z}'_{i,0}, \tilde{z}'_{i,1}, \ldots, \tilde{z}'_{i,\tau_{1,i}}$ do satisfy the usual properties of multistep iterates generated by these methods (since the r'_i are the same on each of these iterations) and this is all we need to establish our error bounds. In particular, the known error bounds (see [6, p. 525]) state that the computation of $\tilde{z}'_i = \tilde{z}_{i,\tau_{1,i}}$ from $\tilde{z}_{i,0} = 0$ via $\tau_{1,i}$ iterations of the conjugate gradient or Chebyshev semi-iterative methods with the same r'_i is done with relative error

$$\epsilon'_{1,i} \le \epsilon'_{1,0}\left(1 - \frac{1}{\sqrt{\kappa(A'_1)}}\right)^{\tau_{1,i}}.$$

Note that

$$\log_e\left(1 - \frac{1}{\sqrt{\kappa(A'_1)}}\right) \approx \frac{-1}{\sqrt{\kappa(A'_1)}}.$$

Thus, by definition of

$$\tau_{1,i} = \sqrt{\kappa(A'_1)}\log\left(\frac{\kappa^2(A_1)\kappa(A'_1)\epsilon'_{1,0}}{\epsilon'_i}\right),$$

we have that the relative error for the level 1 preconditioned iterations over matrix A'_1 is at most

$$\epsilon'_{1,i} = \epsilon'_{1,0}\left(1 - \frac{1}{\sqrt{\kappa(A'_1)}}\right)^{\tau_{1,i}} \approx \frac{\epsilon'_i}{(\kappa^2(A_1)\kappa(A'_1))},$$

satisfying the required relative error bounds for stage i.

Case of Multiple Levels of Recursion of Agorithm PI$_\ell$. In the case of multiple levels of recursion, the error analysis is similar. For each level ℓ on the i^{th} stage, we have the recursively defined matrix A_ℓ and we construct preconditioner matrix B_ℓ. We use the zero vector as the starting iterate, and then apply $\tau_{\ell,i}$ iterations of the chosen base iterative method IMPROVE to the level ℓ preconditioned matrix $A'_\ell = B_\ell^{-1}A_\ell$. Let $\epsilon'_{\ell,i}$ be an upper bound on the relative error of the i^{th} stage of the iterations over preconditioned matrix A'_ℓ. Then the corresponding approximate solution over matrix A_ℓ at the i^{th} stage has relative error upper bounded by $\epsilon_{\ell,i} \le \epsilon'_{\ell,i}\kappa(A_\ell)$, so by the proof of Proposition 2.4, to preserve the convergence of the iterations at level $\ell - 1$ (that is, the iterations over the preconditioned matrix $A'_{\ell-1}$), it suffices to show that this relative error over matrix A_ℓ is upper bounded by

$$\epsilon_{\ell,i} = \frac{\epsilon'_{\ell-1,i}}{(\kappa(A_\ell)\kappa(A'_\ell))}.$$

Hence, it suffices to show that the i^{th} stage of the iterations over A'_ℓ have relative error $\le \epsilon'_{\ell,i} = \epsilon'_{\ell-1,i}/(\kappa^2(A_\ell)\kappa(A'_\ell))$. The error bounds of [6, p. 525] imply that $\tau_{\ell,i}$ iterations of the conjugate gradient or Chebyshev semi-iterative methods over the preconditioned matrix A'_ℓ give relative error

$$\epsilon'_{\ell,i} \le \epsilon'_{\ell,0}\left(1 - \frac{1}{\sqrt{\kappa(A'_\ell)}}\right)^{\tau_{\ell,i}}.$$

Since

$$\log_e\left(1 - \frac{1}{\sqrt{\kappa(A'_\ell)}}\right) \approx \frac{-1}{\sqrt{\kappa(A'_\ell)}},$$

if we define

$$\tau_{\ell,i} = \sqrt{\kappa(A'_\ell)}\log\left(\frac{\kappa^2(A_\ell)\kappa(A'_\ell)\epsilon'_{\ell,0}}{\epsilon_{\ell-1,i}}\right),$$

then the iterations over matrix A'_ℓ have relative error at most

$$\epsilon'_{1,i} = \epsilon'_{\ell,0}\left(1 - \frac{1}{\sqrt{\kappa(A'_\ell)}}\right)^{\tau_{\ell,i}} \approx \frac{\epsilon'_i}{(\kappa^2(A_\ell)\kappa(A'_\ell))},$$

satisfying the required relative error bounds for stage i on level ℓ.

Suppose

$$\kappa(A_\ell), \kappa(B_\ell) \leq O\left(n^2\kappa(A)\right),$$

then

$$\begin{aligned}\epsilon'_{\ell,i} &= \frac{\epsilon'_{\ell-1,i}}{(\kappa^2(A_\ell)\,\kappa(A'_\ell))} \\ &\geq \Omega\left(\frac{\epsilon'_{\ell-1,i}}{(n^4\kappa^3(A))}\right) \\ &\geq \Omega\left(\frac{\epsilon'_{0,i}}{(n\kappa(A))}\right)^{4\ell} \\ &\geq \Omega\left(\frac{\epsilon}{(n\kappa(A))}\right)^{4\ell}\end{aligned}$$

since

$$\epsilon'_{0,i} = \epsilon'_i = \frac{\epsilon}{(\kappa(A)\kappa(A'))},$$

so

$$\begin{aligned}\tau_{\ell,i} &= \sqrt{\kappa(A'_\ell)}\log\left(\frac{\kappa^2(A_\ell)\kappa(A'_\ell)}{\epsilon_{\ell-1,i}}\right) \\ &\leq O\left(\sqrt{\kappa(A'_\ell)}\ell\log\left(\frac{n\kappa(A)}{\epsilon}\right)\right),\end{aligned}$$

and by Proposition 2.4, Proposition 7.1 follows.

PROPOSITION 7.1. *If each*

$$\kappa(A_\ell), \kappa(B_\ell) \leq O\left(n^2\kappa(A)\right),$$

then to satisfy the required relative error bound $\epsilon'_{\ell,i}$ for stage i on level ℓ, it suffices to set

$$\tau_{\ell,i} \leq O\left(\sqrt{\kappa(A'_\ell)}\ell\log\left(\frac{n\kappa(A)}{\epsilon}\right)\right).$$

REFERENCES

1. P. Vaidya, Invited Talk *Workshop on Graph Theory and Sparse Matrix Computation, Special Year Algebra* (October 14–18, 1991), Institute for Mathematics and Its Applications, University of Minnesota, MN.
2. K.D. Gremban, G.L. Miller and M. Zagha, Performance evaluation of a new parallel preconditioner, Technical Report CMU-CS-94-205, Carnegie Mellon University, (1994).
3. K.D. Gremban, G.L. Miller and M. Zagha, Performance evaluation of a new parallel preconditioner, In *Proc. 9th International Parallel Processing Symposium*, pp. 65–69, (1995).
4. K.D. Gremban, Combinatorial preconditioners for large sparse, diagonally dominant linear systems CMU Technical Report CMU-CS-96-123, Ph.D. Thesis, Carnegie Mellon University, (October 1996).
5. D. Coppersmith and S. Winograd, Matrix multiplication via arithmetic progressions, *J. Symbolic Comput.* 9 (3), 251–280, (1990).
6. G.H. Golub and C.F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, (1989).
7. J.M. Ortega, *Numerical Analysis, A Second Course*, Academic Press, New York, (1972).
8. B. Bollobas, *Extremal Graph Theory*, Academic Press, (1978).
9. N. Alon, P. Seymour and R. Thomas, A separator theorem for graphs with excluded minor and its applications, In *Proc. 2nd Annual ACM Symp. Theory of Computing*, pp. 293–299, (1990).
10. R.J. Lipton, D.J. Rose and R.E. Tarjan, Generalized nested dissection, *SIAM J. Num. Anal.* **16**, 346–358, (1979).
11. J.R. Gilbert and R.E. Tarjan, The analysis of a nested dissection algorithm, *Numer. Math.* **50** (4), 377–404, (1987).
12. S.F. Ashby, Polynomial preconditioning for conjugate gradient methods, Technical Report UIUCDCS-R-87-1355, Univ. Illinois, (1987).

13. O. Axelsson, Bounds of eigenvalues of preconditioned matrices, *SIAM J. Matrix Anal. Appl.* **13** (3), 847–862, (1992).
14. O. Axelsson and G. Lindskog, On the rate of convergence of the preconditioned conjugate gradient method, *Numer. Math.* **48**, 499–523, (1986).
15. O. Axelsson and P.S. Vassilevski, Algebraic multilevel preconditioning methods, *Numer. Math.* **56**, 157–177, (1989).
16. J.H. Bramble, J.E. Pasciak and J. Xu, Parallel multilevel preconditioners, *Math. Comp.* **55**, 1–22, (1990).
17. S.C. Eisenstat, Efficient implementation of a class of preconditioned conjugate gradient methods, *SIAM J. Sci. Stat. Comput.* **2**, 1–4, (1981).
18. I.S. Duff and G.A. Meurant, The effect of ordering on preconditioned conjugate gradients, *BIT* **29**, 635–657, (1989).
19. H.A. van der Vorst, High performance preconditioning, *SIAM J. Sci. Stat. Comput.* **10** (6), 1174–1185, (1999).
20. M.A. Heroux, P. Vu and C. Yang, A parallel preconditioned conjugate gradient package for solving sparse linear systems on a Cray Y-MP, *Appl. Num. Math.* **8**, 93–115, (1991).
21. X.-Z. Guo, Multilevel preconditioners: Analysis, performance enhancements, and parallel algorithms, In Technical Report CS-TR-2903, University of Maryland, (1992).
22. A. Brandt, Multi-level adaptive solutions to boundary-value problems, *Math. Comput.* **31**, 333–390, (1977).
23. K. Stuben, Algebraic multigrid (AMG): Experiences and comparisons, *Appl. Math. Comp.* **13**, 419–451, (1983).
24. D. Braess and W. Hackbusch, A new convergence proof for the multigrid method including the V-cycle, *SIAM J. Numer. Anal.* **20**, 975–986, (1983).
25. R.B. Ewing, Editor, *Frontiers in Applied Mathematics: Multigrid Methods*, SLANS, (1987).
26. W.L. Briggs, A multigrid tutorial, *SIAM*, (1987).
27. W.H. Press and S.A. Teukolsky, Multigrid methods for boundary value problems, *Computers in Physics*, 514–519, (1991).
28. R.S. Varga, *Matrix Iterative Analysis*, Prentice Hall, (1962).
29. D.M. Young, *Iterative Solution of Large Linear Systems*, Academic Press, (1971).
30. O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, (1984).
31. L.A. Hageman and D.M. Young, *Applied Iterative Methods*, Academic Press, (1981).
32. W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, Springer-Verlag, (1994).
33. W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, (1988).
34. G. Golub and J.M. Ortega, *Scientific Computing: An Introduction with Parallel Computing*, Academic Press, (1993).
35. M.T. Heath, E. Ng and B.W. Peyton, Parallel algorithms for sparse linear systems, In *Parallel Algorithms for Matrix Computations*, SIAM, (1990).
36. M.R. Hestenes and E. Stiefel, Methods of conjugate gradients for solving linear systems, *J. Res. Nat. Bur. Standards* **49**, 409–436, (1952).
37. J.K. Reid, On the method of conjugate gradients for the solution of large sparse systems of equations, In *Large Sparse Sets of Linear Equations*, pp. 231–254, Academic Press, (1971).
38. G. Golub and D. O'Leary, Some history of the conjugate gradient and Lanczos algorithms: 1948-1976, *SIAM Rev.* **31**, 50–102, (1989).
39. P.G. Doyle and J.L. Snell, Random walks and electric networks, *Carus Mathematical Mono-Graphs* **22**, (1984).
40. A.K. Chandra, P. Raghavan, W.L. Ruzzo, R. Smolensky and P. Tiwari, The electrical resistance of a graph captures its commute and cover times, In *Proc. 21st Annual ACM Symposium on the Theory of Computing*, pp. 574–586, (1989).
41. P.N. Klein and R.E. Tarjan, A randomized linear-time algorithm for finding minimum spanning trees, In *Proc. 26th ACM Symposium on Theory of Computing*, Montreal, Canada, pp. 9–15, (1994).
42. R.J. Lipton and R.E. Tarjan, A separator theorem for planar graphs, *SIAM J. Applied Math.*, 177–189, (1979).
43. M. Arioli, I. Duff and D. Ruiz, Stopping criteria for iterative solvers, *SIAM J. Matrix Anal. Appl.* **13** (1), 138–144, (1992).
44. K.D. Gremban, G.L. Miller and S.-H. Teng, Moments of inertia and graph separators, In *Proc. 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 452–461, (1994).
45. V. Pan and J.H. Reif, On the bit complexity of discrete solutions of PDEs: Compact multigrid, *SIAM Journal of Scientific and Statistical Computing*, 612–625, (1991).
46. J.H. Reif, Communication to K.D. Gremban, (1996).