Iteratively Constructing Preconditioners
via the Conjugate Gradient Method

John Dunagan
Microsoft Research
Redmond, WA
jdunagan@microsoft.com

Nicholas J.A. Harvey
MIT
Cambridge, MA
nickh@mit.edu

ABSTRACT
We consider the problem of solving a symmetric, positive definite system of linear equations. The most well-known and widely-used method for solving such systems is the preconditioned Conjugate Gradient method. The performance of this method depends crucially on knowing a good preconditioner matrix. We show that the Conjugate Gradient method itself can produce good preconditioners as a by-product. These preconditioners allow us to derive new asymptotic bounds on the time to solve multiple related linear systems.

Categories and Subject Descriptors
G.1.3 [Numerical Analysis]: Numerical Linear Algebra—
Linear systems (direct and iterative methods)

General Terms
Algorithms, Theory

Keywords
Conjugate Gradient Method, Preconditioning

1. INTRODUCTION
One of the most basic and useful computational problems is to solve a system of linear equations $Ax = b$, where $A$ is a given matrix, $b$ is a given vector, and $x$ is a vector of unknowns. In this paper, we consider solvers for linear systems over the reals, in which the matrix $A$ is symmetric and positive definite. Such systems have applications in a wide range of areas, including computer graphics [9], machine learning [16], scientific computing, and many branches of engineering.

A comprehensive introduction to linear system solvers can be found in standard references [7, 11, 17, 19]. Roughly speaking, solvers can be divided into two categories.

Direct solvers. These typically work by factoring the matrix $A$ into a canonical form such that the system becomes easy to solve. The prototypical examples are Gaussian elimination (LU factorization), Cholesky factorization, etc.

Iterative solvers. These produce a sequence of “guesses” for the solution. Each iteration of the algorithm adjusts the previous guess, obtaining a new guess which is closer to the final solution. This process is similar to function minimization by gradient descent. The prototypical iterative solvers are Steepest Descent and CG (Conjugate Gradient).

Iterative solvers are of interest for a number of reasons. First, there are numerous circumstances in which it is preferable to rapidly compute an approximate solution to a problem instead of waiting the amount of time needed to compute an exact solution. Next, iterative solvers often have superior performance when the system being solved is sparse. For example, if $A$ has size $n \times n$ and only $m$ non-zero entries, then the CG solver requires only $O(mn)$ operations to produce an exact solution. In contrast, the best theoretical running time known for any direct solver is $O(n^{2+\omega})$, where $0.37 < \omega \leq 1$.

Iterative solvers are so widely used that specialized versions have been developed for various problem families, e.g., the Multigrid method [20] and the algorithm of Spielman-Teng [18]. For a lengthier discussion of iterative solvers, we refer the interested reader to the literature [2, 11, 12, 19].

A qualitative difference between direct and iterative solvers is that the efficiency of iterative solvers usually depends more crucially on the spectrum of the matrix $A$. This is usually formalized by defining a condition number, a quantity derived from $A$ which captures the attractiveness of its spectrum, then analyzing the convergence of the iterative solver as a function of the condition number.

This motivates the study of preconditioning. Instead of solving the system $Ax = b$, one introduces a matrix $P$ (called a preconditioner), then solves the related system $PAx = Pb$. The intention is that $PA$ should have a smaller condition number than $A$, so the problem has become easier to solve, and that introducing $P$ does not impose significant additional costs. The importance of preconditioning techniques is emphasized by Trefethen and Bau [19]:

Nothing will be more central to computational science in the next century than the art of transforming a problem that appears intractable into another whose solution can be approximated rapidly.

For [iterative solvers], this is preconditioning.
While provably good pre-conditioners have been derived for many problems with special structure [12, 13, 18], pre-conditioners for the general symmetric positive definite case are typically based on heuristics [2, 3].

This paper investigates a new approach for constructing preconditioners. We present an algorithm that iteratively constructs a good preconditioner for $A$ by observing the execution of the Conjugate Gradient method (or Steepest Descent) on the system $Ax = b$.

### 1.1 Overview

We begin with an informal overview of our algorithm. The problem $Ax = b$ amounts to finding the minimum of $\|Ax - b\|^2$, which is a convex, quadratic function (its level sets are ellipsoids). Steepest Descent and CG work roughly as follows: given a guess $x$, they compute an improved guess by computing the gradient at $x$, then moving $x$ in the direction of that gradient. This works since the gradient at $x$ points roughly towards the minimum. If the level sets are nearly spherical then this procedure works very well because the gradients will point closely towards the minimum. However, if the level sets are far from spherical, then more iterations will be required. Having nearly spherical level sets amounts to $A$ having small condition number.

Our algorithm repeatedly performs iterations of CG, at each step measuring how much progress was made towards the minimum. If an iteration fails to make much progress, this yields a certificate that the level sets are not spherical in the direction of $x$. This certificate can then be used to find a new preconditioner for $A$ which improves the condition number. The new preconditioner acts on $A$ by squashing the level sets to make them more spherical.

Stated more generally, our algorithm’s approach is to monitor the progress of a basic algorithm (CG). When the basic algorithm is slow, this indicates that the given instance has some undesirable properties (bad condition number). One then takes some alternative action to improve the instance (update the preconditioner), thereby accelerating the performance of the basic algorithm. A similar approach underlies the linear program solvers of Dunagan-Vempala [8] and Kelner-Spielman [14]. In [8], the basic algorithm is the perceptron method, which has running time that is exponential in the input size, and the improvement process accelerates the runtime to polynomial. In [14], an analogous improvement is obtained when the basic algorithm is a certain variant of the simplex method.

For the case of linear systems, our iterative preconditioner updates do not provably accelerate the performance of the already “diabolically fast” [19] CG algorithm. Instead, the iterative preconditioner construction makes the following three contributions. First, it gives an entirely different approach to deriving asymptotic bounds comparable to some of the known asymptotic bounds for Conjugate Gradient. Second, these bounds hold even if the basic algorithm used to construct the preconditioners is Steepest Descent (see Section 6); this is in marked contrast to Steepest Descent’s inferior performance relative to CG in the absence of iterative preconditioner updates. Finally, regardless of whether CG or Steepest Descent is used, the preconditioner serves an additional purpose: it can be used to accelerate the solution of related linear systems (same matrix $A$, new vector $b$), analogous to how direct solvers can reuse a matrix factorization after computing it once.

### 1.2 Designing Preconditioners

As mentioned above, an ideal preconditioner should improve the condition number of the given system without introducing much additional computational cost. Naturally there is a tradeoff between these two objectives, and it is not well understood what tradeoffs can be achieved for general linear systems. Our algorithm gives a framework for making precise tradeoffs between these objectives.

As outlined above, our algorithm produces a sequence of suggested improvements to the preconditioner $P$. Each improvement amounts to a “rank-1 update” of $P$, which increases the cost of a computing matrix-vector products with $P$ (see Section 6.1). On the other hand, the algorithm can precisely determine the extent to which each suggested update decreases a condition number of the system. Thus our algorithm allows an implementer to design an arbitrary policy for deciding which suggested improvements to accept and which to reject. To the best of our knowledge, no such framework was previously known.

There do exist other solvers similar to CG which, in some sense, iteratively compute a new “preconditioner” as each step of the algorithm. These are the so-called quasi-Newton or variable metric methods from non-linear optimization, such as the BFGS and DFP algorithms [4, 5, 6, 10, 15]. The preconditioners computed by these algorithms do successively improve in quality in the sense that they converge towards $A^{-1}$. However, unlike our framework, these algorithms do not show how the condition number improves at each step.

### 2. OVERVIEW OF THE ALGORITHM

We denote the system of linear equations we are trying to solve by $Mx = d$, where $M$ is symmetric positive definite, and has size $n \times n$. If $x^*$ is the actual solution to the system, and $x$ is a guess for the solution, then the error is defined to be $e = x - x^*$. The residual is defined to be $r = x - d = Mx - d$. We will solve the system by incorporating a preconditioner $P$. Consider the new system

$$P^TMP\hat{x} = P^Td.$$  

The new system is equivalent to the original one because, given any solution $\hat{x}$ to the new system, we obtain a solution $x$ to the original one by setting $x = P\hat{x}$. Furthermore, the matrix $P^TMP$ is symmetric positive definite, assuming that $P$ is non-singular.

Pseudocode for our algorithm is shown in Algorithm 1. It generates a sequence of guesses for the solution $x_1, x_2, \ldots$ and a sequence of preconditioners $P_1, P_2, \ldots$. At iteration $i$ of the algorithm, define

$$A_i = P_i^TMP_i \quad \text{and} \quad b_i = P_i^Td.$$  

At step $i$, the error and residual (after preconditioning) are respectively defined to be $e_i = x_i - x^*$ and $r_i = A_ix_i - b_i$.

#### 2.1 Potential Functions

Our algorithm and its analysis are based on two potential functions, which we now define. Let $X$ be a positive definite matrix. We define a new condition number for $X$, called its eccentricity, as follows.

$$\mathcal{E}(X) = \det \left( \frac{X^{1/2} + X^{-1/2}}{2} \right).$$  

The two potential functions are $\phi(i) = \|r_i\|^2$ and $\psi(i) = $
Algorithm 1: An overview of our main algorithm.

Let $x_0$ be an arbitrary initial guess, and let $P_0 = I$.

Repeat

Initialize the Conjugate Gradient Algorithm.

Repeat

Increment $i$.

Step 1: Perform one iteration of the Conjugate Gradient algorithm to compute a better guess $x_{i+1}$, so that $\|r_{i+1}\| < \|r_i\|$.

If $\|r_{i+1}\|$ is sufficiently small then Halt.

Until $\|r_{i+1}\|$ is not much smaller than $\|r_i\|$.

Now $r_i$ is a certificate that $A_i$ is “eccentric”.

Step 2: Compute a better preconditioner $P_{i+1}$ from $P_i$ so that $A_{i+1}$ is less “eccentric” than $A_i$.

Modify $x_{i+1}$ to match the new preconditioner.

Forever

Algorithm 2: A single iteration of our variant of the Conjugate Gradient algorithm (which is Step 1 in Algorithm 1). Initially $x_i$ is the current guess for the solution, $r_i$ is the current residual, and $d_i$ is the current search direction. The new vectors are $x_{i+1}$, $r_{i+1}$, and $d_{i+1}$ respectively.

\[
\alpha_i = -\frac{d_i^T Ar_i}{d_i^T A^2 d_i} \quad \text{[Equivalently, } \alpha_i = -\frac{(r_i^T Ar_i)}{(d_i^T A^2 d_i)} \text{ by Eq. (C.3)}] \]
\[
x_{i+1} = x_i + \alpha_i d_i \quad r_{i+1} = r_i + \alpha_i A d_i \quad \beta_i = \frac{(r_{i+1}^T A^2 d_i)}{(d_i^T A^2 d_i)} \quad \text{[Equivalently, } \beta_i = \frac{(r_{i+1}^T A r_{i+1})}{(r_i^T Ar_i)} \text{ by Eq. (C.4)}] \]
\[
d_{i+1} = r_{i+1} + \beta_i d_i \]

$\mathcal{E}(A_i)$. The function $\phi(i)$ measures the magnitude of the residual in the transformed space, i.e., after preconditioning.

The purpose of $\psi(i)$ is to measure the quality of the current preconditioner, i.e., the extent to which the level sets of $\|A_i x_i - b_i\| = \|r_i\|$ are not spherical. The following lemma makes this precise. To illustrate this lemma, consider the examples shown in Figure 1.

Lemma 2.1. $\mathcal{E}(X)$ is uniquely minimized when $X$ is the identity matrix, which has $\mathcal{E}(I) = 1$.

This lemma shows that $\psi(i)$ measures the extent to which the level sets of $\|r_i\|$ are not the unit sphere. That is, our eccentricity function has the property that it is not invariant under scaling, i.e., $\mathcal{E}(X) \neq \mathcal{E}(cX)$ for most scalars $c \neq 1$. We discuss this issue further in Section 7.

3. THE CONJUGATE GRADIENT ALGORITHM

Step 1 of Algorithm 1 is based on a variant of the Conjugate Gradient (CG) algorithm. Algorithm 2 presents pseudocode for a single iteration of our variant of CG. Readers familiar with CG will notice several differences between our variant and the usual formulation; we discuss these differences in Section 3.1. Readers unfamiliar with CG may learn more from the highly readable exposition of Shewchuk [17].

An Iteration: For convenience, let us drop the subscripts on $A_i$, and $b_i$. Also, define the inner product $\langle x, y \rangle = x^T A y$. The CG algorithm maintains three vectors: $x$, $r$ and $d$. Each iteration updates $x$ according to the formula

\[
x_{i+1} = x_i + \alpha_i d_i, \quad (3.1) \]

where

\[
\alpha_i = -\frac{(d_i, e_i)}{(d_i, d_i)} = \frac{d_i^T A r_i}{d_i^T A^2 d_i}, \quad (3.2) \]

(The value of $\alpha_i$ is chosen to minimize $\phi(i+1)$, as can be seen by Eq. (3.10), although this fact is not used in the proofs.)

The vectors $d_0, d_1, \ldots$ are called search directions, and they are defined by the formula

\[
d_0 = r_0, \quad d_{i+1} = r_{i+1} + \beta_i d_i \quad \forall i \geq 0, \quad (3.3) \]

where

\[
\beta_i = \frac{(r_{i+1}, d_i)}{(d_i, d_i)}. \quad (3.4) \]

Eq. (3.1) implies similar equations for the error and residual:

\[
e_{i+1} = e_i + \alpha_i d_i, \quad (3.5) \]
\[
r_{i+1} = r_i + \alpha_i A d_i. \quad (3.6) \]

A standard proof of our variant of CG is in Appendix B.

Lemma 3.1. In Algorithm 2, we have

\[
d_i^T A r_i = r_i^T A r_i = (3.7) \]
\[
\alpha_i = -\frac{r_i^T A r_i}{(d_i, d_i)} = (3.8) \]
\[
(d_i, d_i) \leq (r_i, r_i). \quad (3.9) \]

Change in Potential: How do the potential functions change after performing a CG iteration? Clearly $\psi$ is unchanged. The decrease in $\phi$ is given by

\[
\frac{\phi(r_{i+1})}{\phi(r_i)} = \frac{r_i^T r_{i+1}}{r_i^T r_i} = (3.10) \]
\[
= 1 + \frac{2\alpha_i d_i^T A r_i + \alpha_i^2 d_i^T A^2 d_i}{r_i^T r_i}. \]

By Lemma 3.1, we obtain

\[
1 - \frac{2}{(r_i, r_i)} = \frac{(r_i^T A r_i)^2}{(d_i, d_i) \cdot r_i^T r_i} + \frac{(r_i^T A r_i)^2}{(d_i, d_i) \cdot r_i^T r_i} = (3.11) \]

\[
\leq 1 - \frac{(r_i^T A r_i)^2}{(r_i, r_i) \cdot r_i^T r_i} = \frac{(r_i^T A r_i)^2}{r_i^T A^2 r_i \cdot r_i^T r_i} \leq 1. \]

To summarize this computation, Fact 2 in Appendix A implies that $\phi$ indeed decreases, unless $r_i = 0$. That is,

\[
0 \leq \frac{\phi(r_{i+1})}{\phi(r_i)} < 1. \]

If $\frac{\phi(r_{i+1})}{\phi(r_i)} \leq 1$, then the residual has decreased substantially, and we have made progress towards solving the linear system. If $\phi$ does not decrease substantially then the following inequality holds:

\[
\frac{(r_i^T A r_i)^2}{(r_i^T A r_i)(r_i^T r_i)} \leq 1. \]
In this case, the vector \( r_i \) is a certificate of eccentricity. This is because \( \frac{(r_i^T A r_i)^2}{(r_i^T A^2 r_i)(r_i^T r_i)} = 1 \) if and only if \( r_i \) is an eigenvector of \( A \). Thus the certificate of eccentricity proves that \( A \) has at least two distinct eigenvalues, and hence the level sets of \( f \) are eccentric (elliptical rather than spherical). When Step 1 of the algorithm fails to decrease the norm of the residual by more than a prescribed amount, we are guaranteed to have such a certificate of eccentricity for use in Step 2.

Informally, the certificate of eccentricity is a proof of “non-sphericeness” of the ellipsoid corresponding to \( A_i \), which means that it is also a certificate of “non-unit-sphericness”, the property captured by \( \psi(i) \).

### 3.1 Comparison to Usual CG

There are three main differences between the our formulation of CG and the usual one.

1. We use a non-standard potential function for analyzing progress, namely \( \phi(i) = \| r_i \|^2 = x^T A^2 x - 2b^T A x + b^T b \).

2. We use a different definition of conjugacy, namely orthogonality relative to \( A^2 \) rather than \( A \).

3. Our initial search direction is the residual vector, which is not the gradient of our potential function \( \phi(i) \), but is the gradient of the usual CG potential function.

Our choice of this CG variant was not by accident. We found that the usual formulation of CG did not produce the guarantees needed for our later analysis, namely, the certificates of eccentricity in Eq. (3.11). Interestingly, we are not aware of previous work finding such a need to depart from the usual formulation of CG.

As mentioned in point (3) above, our search directions are not based on the gradient of our potential function \( \phi \), but instead on the gradient of the usual CG potential function. Rather remarkably, we obtain stronger certificates of eccentricity by using a mismatched gradient and potential function.

### 4. Improving the Preconditioner

Step 2 of our algorithm updates the preconditioner by applying a rank-1 update. Specifically, we set

\[
P_{i+1} = P_i (I + \frac{\sigma}{v^T v}vv^T),
\]

where \( \sigma > -1 \) is a scalar and \( v \) is a non-zero vector, both to be specified later. The algorithm also sets \( x_{i+1} \) appropriately for the new preconditioner. We discuss these changes in detail below.

**Choosing \( v \) and \( \sigma \):** Note that \( A_{i+1} \) becomes

\[
(I + \frac{\sigma}{v^T v}vv^T) A_i (I + \frac{\sigma}{v^T v}vv^T).
\]

We now analyze how this change affects \( \psi \).

**Lemma 4.1.** The change in \( \psi \) is given by

\[
\frac{\psi(i+1)}{\psi(i)} = \frac{\mathcal{E}(A_{i+1})}{\mathcal{E}(A_i)} = (1 + \sigma)^{-1} \cdot \left(1 + \frac{2\sigma + \sigma^2}{v^T v} v^T A_i (A_i + I)^{-1} v\right).
\]

The change in \( \psi \) could be less than or greater than 1, depending on the choice of \( \sigma \) and \( v \). Ideally the change would be less than 1, because this means that the eccentricity decreased. How can we ensure that this happens? Suppose that the previous Step 1 produced a certificate of eccentricity. Then by Fact 2, we have

\[
1 \gg \epsilon := \frac{(r_i^T A r_i)^2}{(r_i^T A_i^2 r_i) \cdot (r_i^T r_i)} \geq \frac{(r_i^T A^2 r_i)^2}{(r_i^T A_i^2 r_i) \cdot (r_i^T r_i)}. \quad (4.1)
\]

This certificate will allow us to choose \( \sigma \) and \( v \) such that \( \psi \) decreases significantly.

First, define the scalars

\[
\zeta = \frac{v^T A (A + I)^{-1} v}{v^T v} \quad \text{and} \quad \sigma = -1 + \frac{\sqrt{1 - \zeta}}{\sqrt{\zeta}}. \quad (4.2)
\]

The eigenvalues of \( A (A + I)^{-1} \) are obviously \( \{\lambda_i/(1 + \lambda_i)\} \), where \( \{\lambda_i\} \) are the eigenvalues of \( A \). Since \( A \) is positive
definite, we obtain $0 < \zeta < 1$, so $\sigma$ is indeed well-defined. For any choice of $v$, the definition of $\sigma$ in Eq. (4.2) is actually the value that minimizes $\frac{\sigma}{\sigma_i(x_i)}$.

We choose $v$ by considering two cases that result from the last inequality of Eq. (4.1).

Case 2a: $\frac{r_i^T A^2 r_i}{r_i^T r_i} < \sqrt{\epsilon}$.

In this case, pick $v = (A_i + I)r_i = A_i r_i + r_i$.

Case 2b: $\frac{r_i^T A^2 r_i}{r_i^T r_i} < \sqrt{\epsilon}$.

In this case, pick $v = (A_i + I)A_i r_i = A_i^T r_i + A_i r_i$.

Using these definitions, together with the formula given in Lemma 4.1, one may design a variety of policies for deciding when Algorithm 1 should perform a Step 2. One possible policy is to perform a Step 2 whenever $\epsilon$ is sufficiently small, say $\epsilon \leq 2^{-16}$. This policy will be assumed for the remainder of the paper. As shown in the following lemma, this policy ensures that each Step 2 provably decreases $\psi$ by a factor of at least $e^{1/16}$. Though this policy is sufficient for our asymptotic analysis, an actual implementation might benefit from using a different policy.

**Lemma 4.2.** If Step 2 is performed with $\epsilon \leq 2^{-16}$ then the following claims hold:

- In Case 2a, $\zeta < e^{1/4} \sqrt{1 - \zeta} - 1$.
- In Case 2b, $1 - \zeta < e^{1/4}$ and $-1 < \sigma < 0$.
- In both cases, $\frac{v_i(1+\epsilon)}{v_i(1-\epsilon)} < 2\sqrt{1-\zeta}-\zeta < e^{1/16}$.

Choosing $x_{i+1}$: Modifying the preconditioner changes the linear system that is being solved. It might be the case that $x_i$ is a good guess for $A_i x = b_i$ but a bad guess for $A_{i+1} x = b_{i+1}$. Therefore, we specify a new value for $x_{i+1}$ as follows:

$$x_{i+1} = (I + \frac{\sigma}{v_i v_i^T})^{-1} x_i = (I - \frac{\sigma(1+\epsilon)}{v_i v_i^T} v_i v_i^T)^{-1} x_i.$$  

**Lemma 4.3.** The choice of $x_{i+1}$ affects $\phi$ as follows:

$$\frac{\phi(i+1)}{\phi(i)} \leq \begin{cases} \sqrt{1/\zeta} & \text{(in Case 2a)} \\ 1 & \text{(in Case 2b).} \end{cases}$$

5. **Runtime Analysis**

In this section, we analyze the number of iterations needed to reduce the residual to a fraction $\nu$ of its original magnitude. Recall that there are two residuals under consideration.

**Transformed residual:** $r_i = P_i^T M P_i x_i - P_i^T d$

**Untransformed residual:** $r = M(P x_i) - d$

As is clear from these expressions, the relationship between the transformed and untransformed residuals is $r = P_i^T r_i$. Our algorithm seeks to minimize $\|r\|$ by performing steps that decrease $\|r_i\|$. Of course, both norms are uniquely minimized when $r = r_i = 0$, but when $r_i$ is non-zero, a step which decreases its norm might actually increase $\|r\|$. So to analyze the progress in decreasing $\|r\|$, one must consider $P_i^T$, the matrix which relates $r$ and $r_i$. Specifically, we have $\|r\| \leq \|P_i^T\| \cdot |r_i|$. Recall that the preconditioner matrix $P_i$ is modified only during Step 2. When Step 2 modifies the preconditioner, this impedes the effort to reduce the untransformed residual in two ways.

**Issue 1:** The norm of the transformed residual might increase, as Lemma 4.3 suggests.

**Issue 2:** The norm of $P_i^T$ might increase. In other words, Step 2 might increase the upper bound on the gap between the transformed residual and the untransformed residual.

To analyze these issues, let $\epsilon_i$, $\sigma_i$, and $\zeta_i$ denote their respective values during the $i$th iteration of Step 2. First we deal with Issue 1. Lemma 4.3 implies that the total amount by which the norm of the transformed residual increases is at most $\sqrt{1/\zeta}$. This quantity is bounded in the following lemma.

**Lemma 5.1.** $\Pi_i$ in Case 2a, $\sqrt{1/\zeta} < E(M)^2$.

The second issue is dealt with by the following lemma.

**Lemma 5.2.** For any $k$, we have $\|P_k\| < E(M)^2$.

From our analysis of issues 1 and 2, we find that Step 1 must decrease the norm of the residual by an additional factor of at most $E(M)^4$. This discussion yields the following result.

**Theorem 5.3.** The number of iterations required by Algorithm 1 to decrease the untransformed residual by a factor of $\nu$ is at most $O\left(\log(1/\nu) + \log E(M)\right)$.

**Proof.** Since each Step 1 decreases $\|r_i\|$ by a constant factor, we obtain the following upper bound on the number of times that our algorithm needs to perform Step 1:

$$O\left(\log(1/\nu) + \log E(M)\right).$$

Step 2 is performed at most $O\left(\log E(M)\right)$ times.

6. **Comparison of Asymptotics**

As mentioned in Section 1, the best known asymptotic analysis for a direct solver is $O(n^{2+\omega})$ where $\omega > 0.37$. The asymptotic analysis of CG does not admit such a simple closed form, and instead requires finding low-degree polynomials $f$ such that $f(\lambda_i)$ is small for all the eigenvalues $\lambda_i$ of $M$. In particular, most analyses rely on properties of the Chebyshev polynomials. One interesting aspect of our present work is that it provides a self-contained analysis for a variant of CG, without resorting to arguments based on polynomials.

Several useful bounds on CG are given in the article of Axelsson and Lindskog [1]. For example, the number of iterations required to decrease the error by a factor of $\nu$ can be bounded by the following expressions.

$$O\left(\sqrt{\frac{\Lambda_n}{\Lambda_1}} \cdot \log(1/\nu)\right)$$

$$O\left(\sqrt{\frac{\Lambda_n}{\Lambda_1}} \cdot \log(1/\nu) + b \right)$$

for any $b$

$$O\left(\sqrt{\frac{\Lambda_n}{\Lambda_1}} \cdot \log(1/\nu) + \log \Pi_{a=1}^n \frac{1}{\Lambda_i} + a \right)$$

for any $a$

The classical bound that is used in most contexts is Eq. (6.1). Other extensions and combinations of these bounds are possible. Additionally, it is known that CG finds an exact solution in at most $n$ iterations, as mentioned in Appendix B, although this bound frequently fails to hold in practice because of the use of inexact arithmetic.
The bound of Algorithm 1 stated in Theorem 5.3 is similar to the bound on CG given by Eq. (6.3), in the special case \( a = n - 1 \). To see this, suppose for example that the matrix is normalized so that \( \lambda_n = 1 \). Then we have \( \log \prod_{i=1}^{n} \lambda_i = \Theta(\log(M)) \).

We remark that Theorem 5.3 remains valid even if Algorithm 1 is simplified to use ordinary Steepest Descent iterations instead of CG in Step 1. The standard bound [17] on Steepest Descent is that it requires \( O((\lambda_n/\lambda_1) \cdot \log(1/\nu)) \) iterations to decrease the error by \( \nu \). The iterative preconditioner updates accelerate the Steepest Descent algorithm such that its performance is comparable to CG, i.e., the bound of Eq. (6.3) with \( a = n - 1 \).

6.1 Iteration Cost

The preceding discussion considers only iteration counts and neglects the computational cost of each iteration. Each iteration of CG involves only \( O(1) \) matrix-vector products, and hence requires \( O(m) \) time, where \( m \) is the number of non-zero entries of \( M \). If one also uses a preconditioner, the added overhead is the cost of multiplying a vector by the preconditioner.

The cost of using the preconditioners produced by our algorithm depends on how they are represented. The simplest option is to store \( P \) explicitly; it will typically have \( \Theta(\log(M)) \) non-zero entries of \( M \). If one also uses a preconditioner, the bound of Algorithm 1 stated in Theorem 5.3 is similar on CG are also \( \Omega(\log M) \), and all stated bounds on CG are also \( \Omega(n^{-\alpha/2}) \). Let \( k = n^{-\alpha/2} \), and \( \nu = 1/\nu > 2^{\alpha^{-2}} \).

7. DISCUSSION

There are many interesting questions left open by our work.

• Our definition of eccentricity is not scale-invariant, as noted in Section 2.1. This is undesirable for several reasons. Intuitively, this causes the algorithm to favor wasteful updates which push the eigenvalues towards \( 1 \), rather than useful updates which push the eigenvalues closer together. A natural way to resolve this problem would be to run the algorithm on the rescaled matrix \( M/\prod_{i=1}^{n} \lambda_i^{1/n} \) rather than \( M \). Is there a way to estimate \( \prod_{i=1}^{n} \lambda_i^{1/n} \) (the geometric mean of the eigenvalues) more rapidly than computing \( \det M \)? Alternatively, is there an alternative approach to updating the preconditioner which does not affect the scaling (i.e., det \( P_i = 1 \) for all \( i \))?

• Suppose one needs to solve two systems \( Ax = a \) and \( By = b \), where \( A \approx B \). Intuitively, a good preconditioner for \( A \) should be a good preconditioner for \( B \). That is, if \( E(P^TAP) \) is small then \( E(P^TB) \) should also. Can this be made rigorous? Does this lead to interesting analyses of interior point LP solvers or quasi-Newton algorithms?

• Can Step 1 of Algorithm 1 be modified to use the standard conjugate gradient algorithm, rather than our variant? For example, can the inner product \( \langle x, y \rangle = x^TAy \) be used instead of \( x^TA^Ty \)?

• Can the preconditioner updates be modified so that they maintain conjugacy? In other words, is it necessary to reinitialize CG after each Step 2?
• Do the iteratively updated preconditioners have attractive numerical stability properties?
• Can Algorithm 1 be extended to handle rank-deficient matrices?

We are cautiously optimistic that progress on these questions could positively impact solving linear systems in practice.

Acknowledgements
The authors thank Alan Edelman, Steven G. Johnson, Daniel Spielman, Gilbert Strang and Shang-Hua Teng for helpful discussions. The second author was supported by a Natural Sciences and Engineering Research Council of Canada PGSDS scholarship, by NSF contract CCF-0515221 and by ONR grant N00014-05-1-0148.

8. REFERENCES

APPENDIX
A. GENERAL FACTS

FACT 1. Let $M$ be a matrix. The matrix $M(I + \frac{\sigma}{v^Tw}vv^T)$ is called a rank-1 update of $M$. The matrix $(I + \frac{\sigma}{v^Tv}vv^T)$ has several useful properties.

- It is symmetric, positive definite (assuming $\sigma > -1$).
- Its inverse is $(I - \frac{\sigma/(1+\sigma)}{v^Tv}vv^T)\cdot
- Its determinant is $1 + \sigma$.
- Its norm is $\|I + \frac{\sigma}{v^Tv}vv^T\| = \begin{cases} 1 + \sigma & (\text{if } \sigma > 0) \\ 1 & (\text{if } -1 < \sigma \leq 0) \end{cases}$

PROOF. The first property is trivial. The second property is the Sherman-Morrison formula [11]. Let us now consider the eigenvalues of $(I + \frac{\sigma}{v^Tv}vv^T)$. The eigenvalue in the direction of $v$ is $1 + \sigma$, and the eigenvalues are all 1 in the subspace orthogonal to $v$. This implies the third and fourth properties.

FACT 2. Let $X$ be positive definite and let $w$ be a non-zero vector.

$$0 < \frac{(w^T X w)^2}{(w^T w)(w^T X^2 w)} \leq 1. \quad (A.1)$$

Furthermore,

$$\frac{(w^T X^2 w)^2}{(w^T w)(w^T X^2 w)} \leq \frac{(w^T X w)^2}{(w^T w)(w^T X w)}. \quad (A.2)$$

PROOF. Recall Hölder’s inequality, which states that, for any $p, q \in \mathbb{R}^+$ with $\frac{1}{p} + \frac{1}{q} = 1$,

$$\left| \sum_i x_i y_i \right| \leq \left( \sum_i |x_i|^p \right)^{1/p} \left( \sum_i |y_i|^q \right)^{1/q}. \quad (A.3)$$

Let $\{v_1, \ldots, v_n\}$ be orthonormal eigenvectors of $X$ and let $\lambda_i$ be the eigenvalue associated with $v_i$. We may write
By Eq. (3.3), if 0

This quantity is positive for all j, proving the lower bound in Eq. (A.1). Apply Hölder’s inequality with \( x_i = c_i, y_i = c_i \lambda_i \), and \( p = q = 2 \) (i.e., Cauchy-Schwarz). Squaring Eq. (A.3) yields \((w^T X^2 w)^2 \leq (w^T w) (w^T X w)^2\), establishing the upper bound of Eq. (A.1).

To prove Eq. (A.2), we show the following upper bound.

\[
1 \geq \frac{(w^T X^2 w)^2}{(w^T w)(w^T X^2 w)} = \frac{(w^T X^2 w)^3}{(w^T X^2 w)(w^T X w)^2}.
\]

Applying Hölder’s inequality with \( p = 3, q = 3/2, x_i = c_i^2 / \lambda_i^{1/3}, \) and \( y_i = c_i^4 / \lambda_i^{2/3} \). Taking the cube of Eq. (A.3) yields

\[
\left( \sum_i c_i^2 \lambda_i^3 \right)^3 \leq \left( \sum_i c_i^2 \lambda_i^3 \right) \left( \sum_i c_i^2 \lambda_i^2 \right)^2 \Rightarrow (w^T X^2 w)^3 \leq (w^T X^2 w) (w^T X w)^2,
\]

which is our desired upper bound.

\[\square\]

**B. PROOF OF CG VARIANT**

In this appendix we show that the inner loop of Algorithm 2 is indeed a variant of CG. The defining criteria of a CG algorithm are: (1) the search directions are conjugate (orthogonal relative to the specified inner product, and (2) the error at the \( i \)th step is conjugate to the subspace spanned by the previous search directions. (Note that these criteria imply convergence of the CG algorithm in at most \( n \) iterations.) We formalize these criteria with the following predicates.

\[D(i) : \{ d_i, d_i \} = 0 \quad \forall j < i\]
\[E(i) : \{ d_i, e_i \} = 0 \quad \forall j < i\]

Both are trivial for \( i = 0 \). Assuming both are true for \( i \), we will inductively prove them for \( i + 1 \).

**CLAIM B.1.** \( E(i + 1) \) is true.

**PROOF.** For \( j = i \), we have

\[
\{ d_i, e_{i+1} \} = \{ d_i, e_i \} - \{ d_i, e_i \} \{ d_i, d_i \} = 0
\]

directly from Eq. (3.5) and Eq. (3.2). For \( j < i \), we have

\[
\{ d_j, e_{i+1} \} = \{ d_j, e_i \} + \alpha_j \{ d_j, d_i \} = 0.
\]

\[=0 \text{ by } D(i) \quad =0 \text{ by } D(i)\]

**CLAIM B.2.** \( r_j^T A r_k = 0 \) for all \( j < k \leq i + 1 \).

**PROOF.** By Eq. (3.3), if \( 0 < j \leq k \), we have

\[
\{ d_j, e_k \} = \{ r_j, e_k \} + \beta_j \{ d_{j-1}, e_k \} = \{ r_j, e_k \}, \quad (B.1)
\]

and similarly if \( j = 0 \). By our assumption that \( j < k, E(k) \) implies that \( 0 = \{ d_j, e_k \} = \{ r_j, e_k \} \). Expanding \( \{ r_j, e_k \} \) as \( r_j^T A e_k \) and using \( p_k = A e_k \) yields the desired result, \( r_j^T A r_k = 0 \).

**CLAIM B.3.**

\[
\{ r_{i+1}, d_j \} = \begin{cases} 0 & \text{if } i > j \\
\frac{r_{i+1}^T A r_{i+1}}{\alpha_i} & \text{if } i = j \\
-\frac{r_{i+1}^T A r_{i+1}}{\alpha_i} & \text{if } i = j - 1
\end{cases}
\]

**PROOF.** For arbitrary \( i \) and \( j \) we have

\[
\{ r_{i+1}, d_j \} = \{ r_{i+1}^T A r_{i+1} / \alpha_i \} \quad \text{if } i = j
\]

\[=0 \text{ by Claim B.3} \quad =0 \text{ by } D(i)\]

**C. PROOFS OF LEMMAS**

**PROOF (of Lemma 2.1).** For any matrix \( X \), let \( \lambda_j(X) \) denote the \( j \)th eigenvalue of \( X \). If \( X \) is positive semi-definite, then \( \lambda_j(X^{1/2} + X^{-1/2}) = \sqrt{\lambda_j(X)} + 1/\sqrt{\lambda_j(X)} \). Clearly \( x+1/x \geq 2 \) whenever \( x > 0 \), and equality holds if \( x = 1 \); this follows from \( 0 \leq (x - 1)^2 = x^2 - 2x + 1 \). Thus we obtain

\[
\det \left( \frac{A_i^{1/2} + A_i^{-1/2}}{2} \right) = \prod_{j=1}^{n} \lambda_j \left( A_i^{1/2} + A_i^{-1/2} \right)/2
\]

\[= \prod_{j=1}^{n} \left( \left( \sqrt{\lambda_j(A_i)} + 1/\sqrt{\lambda_j(A_i)} \right) / 2 \right) \geq 1.
\]

Furthermore, equality holds if \( \lambda_j(A_i) = 1 \). But the unique matrix with all eigenvalues equal to 1 is the identity matrix, so the lemma is proven.

**LEMMA C.1.** \( \{ d_i, r_{i+1} \} + \beta \{ d_i, d_i \} = 0 \).

**PROOF.** The following calculations prove the lemma.

\[
\beta_i = -\frac{\{ r_{i+1}, d_i \} \{ d_i, d_i \}}{\{ d_i, d_i \}},
\]

\[
= -\frac{\alpha_i}{\{ d_i, d_i \}}, \quad (C.1)
\]

where the second equality follows from Claim B.3.

\[
\{ d_i, r_{i+1} \} + \beta \{ d_i, d_i \} = \frac{\{ d_i, r_{i+1} \} - \alpha_i r_{i+1}^T A r_{i+1}}{\alpha_i} \alpha_i,
\]

by the second expression for \( \beta \) in Eq. (C.1),

\[
= \left( \{ d_i, r_i \} + \alpha_i \{ d_i, A d_i \} \right) - \frac{\alpha_i}{\{ d_i, d_i \}} \left( r_i^T A r_i + 2 \alpha_i \{ d_i, r_i \} + \alpha_i^2 d_i^T A d_i \right) / \alpha_i,
\]

\[=0 \text{ by Lemma B.4} \quad =0 \text{ by } D(i)\]
by Eq. (3.6),

\[ - \left( r_i^T A r_i / \alpha_i + \langle d_i, r_i \rangle \right) = 0, \]

by Claim B.3.

PROOF (of Lemma 3.1). From Eq. (B.1), we have

\[ d_i^T A r_i = d_i^T A^2 e_i = \left( r_i, e_i \right) = r_i^T A r_i. \quad (C.2) \]

This leads to the desired expression for \( \alpha_i \).

\[ \alpha_i = - r_i^T A r_i \quad / \quad (d_i, d_i) \quad (C.3) \]

For \( \beta_i \), we obtain

\[ \beta_i = - \left( r_{i+1}, d_i \right) / \alpha_i (d_i, d_i) = r_{i+1}^T A r_{i+1} / r_i^T A r_i, \quad (C.4) \]

as in the derivation of Eq. (C.1), and also using Eq. (C.3).

By Eq. (3.3), we have \( d_i = r_i + \beta_i (d_{i-1}, d_{i-1}) \).

Then

\[ \langle d_i, d_i \rangle = \left( r_i, r_i \right) + 2 \beta_{i-1} (d_{i-1}, r_i) + \beta_i^2 (d_{i-1}, d_{i-1}). \]

Therefore it suffices to show that

\[ 2 \beta_{i-1} (d_{i-1}, r_i) + \beta_i^2 (d_{i-1}, d_{i-1}) \leq 0. \]

This follows straightforwardly:

\[ \begin{align*}
2 \beta_{i-1} (d_{i-1}, r_i) + \beta_i^2 (d_{i-1}, d_{i-1}) \\
= 2 \beta_{i-1} (d_{i-1}, r_i) + \beta_{i-1} (d_{i-1}, d_{i-1}) \\
- \beta_i^2 (d_{i-1}, d_{i-1}) \\
= - \beta_i^2 (d_{i-1}, d_{i-1}) \quad \text{(by Lemma C.1)} \\
\leq 0.
\end{align*} \]

Thus the proof is complete.

LEMMA C.2. Let \( B = (I + \gamma v v^T) A (I + \gamma v v^T) \). Then

\[ \det \left( B^{1/2} + B^{-1/2} \right) = \det \left( (I + \gamma v v^T) A^{1/2} + (I + \gamma v v^T)^{-1} A^{-1/2} \right). \]

PROOF. The proof uses only the fact that the determinant is a homomorphism. First, note that

\[ \det \left( B^{1/2} + B^{-1/2} \right) = \det \left( B + I \right) \cdot \det \left( B^{1/2} \right)^{-1}. \]

But

\[ \det \left( B^{1/2} \right) = \left( \det B \right)^{1/2} = \left( \det(I + \gamma v v^T) \cdot \det A \cdot \det(I + \gamma v v^T) \right)^{1/2} = \left( \det A \cdot \det(I + \gamma v v^T) \right)^{1/2} = \det \left( A^{1/2} (I + \gamma v v^T) \right). \]

Thus,

\[ \begin{align*}
\det \left( B^{1/2} + B^{-1/2} \right) & = \det \left( (I + \gamma v v^T) A^{1/2} + (I + \gamma v v^T)^{-1} A^{-1/2} \right) \\
& = \det \left( (I + \gamma v v^T) A^{1/2} + (I + \gamma v v^T)^{-1} A^{-1/2} \right). \quad \text{■}
\end{align*} \]

PROOF (of Lemma 4.1). For convenience, we drop \( i \) subscripts, letting \( A \) denote \( A_i \) and \( r \) denote \( r_i \). By Lemma C.2, we have

\[ \begin{align*}
E(i + 1) & \quad E(i) \\
\frac{\psi(i + 1)}{\psi(i)} & = \frac{\det \left( (I + \frac{\sigma^2}{\gamma v^T} A v^T \right)^{-1/2} + (I + \frac{\sigma^2}{\gamma v^T} v v^T) \cdot A + I \right)^{-1/2}}{\det \left( (I + \frac{\sigma^2}{\gamma v^T} v v^T \right)^{-1} + (I + \frac{\sigma^2}{\gamma v^T} v v^T) \cdot A + I \right)^{-1/2}} \\
& = \frac{(1 + \sigma)^{-1} \cdot \det \left( I + \frac{2 \sigma^2 + \sigma}{\gamma v^T} \cdot A + I \right)^{1/2}}{(1 + \sigma)^{-1} \cdot \det \left( I + \frac{2 \sigma^2 + \sigma}{\gamma v^T} \cdot A + I \right)^{-1/2}} \\
& = \frac{(1 + \sigma)^{-1} \cdot \left( 1 + \frac{2 \sigma^2 + \sigma}{\gamma v^T} \cdot A + I \right)^{-1/2}}{(1 + \sigma)^{-1} \cdot \left( 1 + \frac{2 \sigma^2 + \sigma}{\gamma v^T} \cdot A + I \right)^{1/2}}. \quad \text{by Fact 1.} \quad \text{■}
\end{align*} \]

PROOF (of Lemma 4.2). Again, we drop \( i \) subscripts, letting \( A \) denote \( A_i \) and \( r \) denote \( r_i \). By Lemma 4.1, we wish to analyze

\[ \frac{\psi(i + 1)}{\psi(i)} = \frac{(1 + \sigma)^{-1} \cdot \left( 1 + \frac{2 \sigma^2 + \sigma}{\gamma v^T} \cdot A + I \right)^{-1/2}}{(1 + \sigma)^{-1} \cdot \left( 1 + \frac{2 \sigma^2 + \sigma}{\gamma v^T} \cdot A + I \right)^{1/2}}. \quad \text{(C.5)} \]

by Eq. (4.2).

Case 2a: In this case, we set \( v = (A + I) r \), implying that

\[ \zeta = \frac{v^T A (A + I)^{-1} v}{v^T v} = \frac{r^T A (A + I)^{-1} r}{r^T r}. \quad \text{(C.7)} \]

The condition of Case 2a is

\[ r^T A^2 r < \epsilon^{1/2} \cdot r^T r \quad \implies \quad r^T A r \leq \sqrt{\epsilon^{1/2} \cdot r^T A^2 r} < \epsilon^{1/4} \cdot r^T r, \quad \text{by Eq. (A.1)}. \]

CLAIM C.3. Assuming the hypothesis of Case 2a and that \( \epsilon \leq 2^{-4} \), we have \( \zeta < \epsilon^{1/4} \) and \( \frac{\psi(i + 1)}{\psi(i)} < 2 \epsilon^{1/8} \).
Proof. First we show that \( \zeta < \epsilon^{1/4} \). By Eq. (C.7), this is equivalent to showing:

\[
r^T A (A + I) r < \epsilon^{1/4} (r^T (A + I)^2 r)
\]

\[
\Longleftrightarrow r^T A^2 r + r^T A r < \epsilon^{1/4} (r^T A^2 r + 2 r^T A r + r^T r)
\]

\[
\Longleftrightarrow (1 - \epsilon^{1/4}) r^T A^2 r + (1 - 2 \epsilon^{1/4}) r^T A r - \epsilon^{1/4} r^T r < 0
\]

\[
\Longleftrightarrow \epsilon^{1/4} r^T r + (1 - 2 \epsilon^{1/4}) r^T r - \epsilon^{1/4} r^T r < 0
\]

by \( \epsilon \leq 2^{-4} \) and Eq. (C.8),

\[
\Longleftrightarrow -3^{1/4} - \epsilon^{1/4} < 0,
\]

which holds for all \( \epsilon > 0 \). From Eq. (C.6), we therefore obtain

\[
\frac{\psi(i+1)}{\psi(i)} < 2\sqrt{\zeta} < 2^{1/8}.
\]

Recall that \( \psi = -1 + \sqrt{1/\zeta - 1} \). Since \( \zeta < \epsilon^{1/4} \) and \( \epsilon \leq 2^{-4} \), we have \( \psi < 1/2 \) and therefore \( 0 < \sigma < \sqrt{1/\zeta} - 1 \), as desired.

Case 2b: In this case, we set \( v = (A + I) A r \), implying that

\[
\zeta = \frac{r^T A (A + I)^{-1} v}{v^T v} = \frac{r^T A^3 (A + I) r}{v^T A^3 (A + I) r}. \tag{C.9}
\]

The condition of Case 2b is

\[
r^T A^2 r < \epsilon^{1/2}, r^T A^4 r
\]

\[
\Rightarrow r^T A^3 r \leq \sqrt{r^T A^2 r \cdot r^T A^4 r} < \epsilon^{1/4} \cdot r^T A^4 r, \tag{C.10}
\]

by Eq. (A.1).

Claim C.4. Assuming the hypothesis of Case 2b and that \( \epsilon \leq 2^{-4} \), we have \( 1 - \zeta < \epsilon^{1/4} \) and \( \frac{\psi(i+1)}{\psi(i)} < 2\sqrt{\zeta} < 2^{1/8} \).

Proof. Using Eq. (C.9) and Eq. (C.10), an argument similar to Claim C.3 shows that \( 1 - \zeta < \epsilon^{1/4} \). From Eq. (C.6), we obtain

\[
\frac{\psi(i+1)}{\psi(i)} < 2\sqrt{\zeta} < 2^{1/8}.
\]

Since \( 1 - \zeta < \epsilon^{1/4} \) and \( \epsilon \leq 2^{-4} \), we have \( \zeta > 1/2 \). Since \( \sigma = -1 + \sqrt{1/\zeta - 1} \), it follows that \( -1 < \sigma < 0 \), as desired.

To conclude the proof of Lemma 4.2, if \( \epsilon \leq 2^{-16} \), then from Claim C.3 and Claim C.4 we obtain the following bound on the decrease in \( \psi \):

\[
\frac{\psi(i+1)}{\psi(i)} < 2\epsilon^{1/4} \leq \epsilon^{1/16}.
\]

Proof (of Lemma 4.3). The new residual becomes

\[
r_{i+1} = A_{i+1} x_{i+1} - b_{i+1}
\]

\[
= \left( I + \frac{\sigma}{\psi(i)} v^T \right) A_i \left( I + \frac{\sigma}{\psi(i)} v^T \right)^{-1} x_i
\]

\[
- \left( I + \frac{\sigma}{\psi(i)} v^T \right) b_i
\]

\[
= \left( I + \frac{\sigma}{\psi(i)} v^T \right) r_i.
\]

Thus \( \| r_{i+1} \| \leq \| I + \frac{\sigma}{\psi(i)} v^T \| \cdot \| r_i \| \). By Fact 1, we have

\[
\| r_{i+1} \| \leq \begin{cases} (1 + \sigma) \cdot \| r_i \| & (\text{if } \sigma > 0) \\ \| r_i \| & (\text{if } -1 < \sigma \leq 0) \end{cases}
\]

By Lemma 4.2, the proof is complete.

Proof (of Lemma 5.1). We have

\[
\mathcal{E}(M) \geq \prod_{i \in \text{Case 2a}} \frac{1}{2\sqrt{\zeta_i} \sqrt{\mathcal{E}(M)}}. \tag{C.11}
\]

from Lemma 2.1 and Lemma 4.2,

\[
> \prod_{i \in \text{Case 2a}} \frac{1}{2\sqrt{\zeta_i}}
\]

since \( \sqrt{\frac{1}{\zeta_i} - 1} < 1 \) and \( 2\sqrt{\zeta_i} \sqrt{\mathcal{E}(M)} - \zeta_i < 1 \). Since \( \epsilon_i < 2^{-16} \) and, by Lemma 4.2, \( \zeta_i < \epsilon^{1/4} \), we have \( \zeta_i < 2^{-4} \). Thus \( 2\sqrt{\zeta_i} < \zeta_i^{1/4} \). Consequently,

\[
\mathcal{E}(M) > \prod_{i \in \text{Case 2a}} \zeta_i^{-1/4}, \tag{C.12}
\]

which proves the lemma.

Proof (of Lemma 5.2). As in Eq. (C.11), we have

\[
\mathcal{E}(M) \geq \prod_{i \in \text{Case 2b}} \frac{1}{2\sqrt{\zeta_i} \sqrt{\mathcal{E}(M)}}.
\]

In Case 2b, we have \( 1 - \zeta_i \leq \epsilon^{1/4} \) by Lemma 4.2. Since \( \epsilon \leq 2^{-16} \), it follows that \( \frac{1}{2\sqrt{\zeta_i} - \zeta_i} \leq \frac{1}{2\sqrt{\epsilon^{1/4}} - \epsilon^{1/4}} \). Thus

\[
\mathcal{E}(M)^2 > \prod_{i \in \text{Case 2b}} \frac{1}{\sqrt{1 - \zeta_i}}. \tag{C.13}
\]

For the \( \ell_2 \)-norm, we have \( \| P_k^{-\tau} \| = \| P_k^{-1} \| \). We may write

\[
P_k = \prod_{i=1}^{k} \left( I + \frac{\sigma_i}{\psi(i)} v_i v_i^T \right)
\]

\[
\Rightarrow P_k^{-1} = \prod_{i=1}^{k} \left( I - \frac{\sigma_i/(1 + \sigma_i)}{\psi(i)} v_i v_i^T \right) \quad \text{(by Fact 1)}
\]

\[
\Rightarrow \left\| P_k^{-\tau} \right\| \leq \prod_{i=1}^{k} \left\| I - \frac{\sigma_i/(1 + \sigma_i)}{\psi(i)} v_i v_i^T \right\| \leq \prod_{i \in \text{Case 2b}} \frac{1}{1 + \sigma_i} \quad \text{(by Fact 1)}
\]

\[
= \prod_{i \in \text{Case 2b}} \frac{1}{1 + \sigma_i}
\]

\[
< \prod_{i \in \text{Case 2b}} \frac{1}{\sqrt{1 - \zeta_i}} < \mathcal{E}(M)^2,
\]

by Eq. (C.13).