# A RANDOMIZED ALGORITHM FOR
# PRECONDITIONER SELECTION[*]

CONNER DIPAOLO[†] AND WEIQING GU[‡]

**Abstract.** The task of choosing a preconditioner $M$ to use when solving a linear system $Ax = b$ with iterative methods is difficult. For instance, even if one has access to a collection $M_1, M_2, \ldots, M_n$ of candidate preconditioners, it is currently unclear how to practically choose the $M_i$ which minimizes the number of iterations of an iterative algorithm to achieve a suitable approximation to $x$. This paper makes progress on this sub-problem by showing that the preconditioner stability $\|I - M^{-1}A\|_{\mathsf{F}}$, known to forecast preconditioner quality, can be computed in the time it takes to run a constant number of iterations of conjugate gradients through use of sketching methods. This is in spite of folklore which suggests the quantity is impractical to compute, and a proof we give that ensures the quantity could not possibly be approximated in a useful amount of time by a *deterministic* algorithm. Using our estimator, we provide a method which can provably select the minimal stability preconditioner among $n$ candidates using floating point operations commensurate with running on the order of $n \log n$ steps of the conjugate gradients algorithm. Our method can also advise the practitioner to use no preconditioner at all if none of the candidates appears useful. The algorithm is extremely easy to implement and trivially parallelizable. In one of our experiments, we use our preconditioner selection algorithm to create to the best of our knowledge the first preconditioned method for kernel regression reported to never use more iterations than the non-preconditioned analog in standard tests.

**Key words.** Preconditioning, Randomized Algorithms, Sketching Methods

**AMS subject classifications.** 65F08, 68W20, 68Q17, 62G08

**1. Introduction.** A preconditioner $M$ is helpful for solving a linear system $Ax = b$ via iterative methods if it reduces the number of iterations enough to offset the cost of constructing the preconditioner plus the additional cost of taking the iterations (typically an extra computation of the form $M^{-1}z$ per iteration.) Ignoring the latter component of this trade-off, this corresponds to having a small condition number $\kappa(M^{-1}A)$ for the conjugate gradient method [29, 14]; a precise objective is unclear in the indefinite or unsymmetric case with general Krylov methods besides the general desire that the spectrum of $M^{-1}A$ be clustered [8]. For instance, even if we have a small number of candidate preconditioners $M_1, M_2, \ldots, M_n$ for our problem ready to use and assume that they add the same amount of time to compute each iteration, it is unclear how one would go about estimating which preconditioner would reduce the iteration count the most without actually solving a system with each preconditioner or doing a comparable amount of work. This preconditioner selection task is the focus of the present work.

**1.1. Contributions.** The core contribution of this work is the realization that randomized sketching methods make completely practical the computation of a helpful forecaster of preconditioner quality previously thought to be infeasible to compute.

In addition to this primary method for computing preconditioner stability, we provide a number of other results:

- We prove that no practical deterministic algorithm, in a meaningful sense, could possibly be used to estimate the preconditioner stability $\|I - M^{-1}A\|_{\mathsf{F}}$.

- We confirm the conjecture of [5] regarding the true asymptotic sample complexity of the Gaussian trace estimator using a substantially more direct proof than the general result provided in [33], at the same time confirming the tightness of our stability estimation algorithm convergence bound.
- We provide a randomized algorithm which can provably select a preconditioner of approximately minimal stability among $n$ candidate preconditioners using computational resources equivalent to computing on the order of $n \log n$ steps of the conjugate gradients algorithm.
- We give a theoretical speedup to the initial preconditioner selection method which largely decouples the runtime dependence between the number of preconditioners $n$ and the desired accuracy in the situation that the input preconditioner stabilities satisfy an anti-concentration assumption.
- Using our initial preconditioner selection algorithm, we create the first (to the best of our knowledge) method for preconditioning in kernel regression which is reported to never give a worse number of iterations than using no preconditioner in standard tests.

It is important to point out that while our experiments consider positive definite systems and preconditioners, our algorithms and runtime bounds hold as-is for arbitrary matrices $\boldsymbol{A}$ and preconditioners $\boldsymbol{M}$.

**1.2. Prior Art.** Current approaches to forecasting preconditioner quality are not robust in the sense that they can fail to select a highly performant preconditioner in favor of a poorly performing one in many realistic scenarios. This lack of robustness, even in the case of positive definite $\boldsymbol{A}$, results in part from the fact that the research community does not have robust methods for estimating condition numbers of large sparse matrices, which makes proxies for preconditioner quality necessary. For instance, Avron et al. [4] recently produced a condition number estimator in this setting which appears to perform admirably in many situations but does not always converge and at this point does not have rigorous theoretical backing.

The simplest forecasting criterion is that a preconditioner $\boldsymbol{M}$ ought to be an 'accurate' approximation, in the sense that the Frobenius norm distance $\|\boldsymbol{M} - \boldsymbol{A}\|_{\mathsf{F}}$ is small. This preconditioner accuracy criterion $\|\boldsymbol{M} - \boldsymbol{A}\|_{\mathsf{F}}$ can be efficiently computed so long as one has access to the preconditioning matrix $\boldsymbol{M}$ in memory as well as the standard matrix-vector product access to $\boldsymbol{M}^{-1}$. Moreover, for symmetric $M$-matrices this accuracy criterion is a useful proxy for the number of conjugate gradient iterations necessary to solve the preconditioned system in $\boldsymbol{A}$. This point was theoretically confirmed by Axelsson and Eijkhout [6], who showed that the condition number $\kappa(\boldsymbol{M}^{-1}\boldsymbol{A})$ can be bounded in terms of $\|\boldsymbol{M}^{-1}\|_{\mathsf{F}}\|\boldsymbol{M} - \boldsymbol{A}\|_{\mathsf{F}}$. The accuracy criterion for forecasting preconditioner quality was heavily tested on an empirical level in [15].

Even in this setting, though, there exist accurate real-world preconditioners that give a poor iteration count because $\|\boldsymbol{M}^{-1}\|_{\mathsf{F}}$ is very large [8]. Unfortunately, to detect this issue one presumably needs to compute the entire matrix $\boldsymbol{M}^{-1}$ in memory solely from matrix-vector product access to $\boldsymbol{M}^{-1}$. The natural method for this, computing each of the $d$ columns $\boldsymbol{M}^{-1}\boldsymbol{e}_i$ from products with the standard basis vectors $\boldsymbol{e}_i$, takes the same leading-order floating point cost as solving the system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ via conjugate-gradients, at least when matrix-vector multiply access to $\boldsymbol{M}^{-1}$ is comparable to the matrix-vector multiply cost of multiplying by $\boldsymbol{A}$. Perhaps as a result of this intuition, computing $\|\boldsymbol{M}^{-1}\|_{\mathsf{F}}$, or rather its close cousin the $\ell^{\infty}$-norm $\|\boldsymbol{M}^{-1}\|_{\infty}$, was deemed impractical [8, 7]. In practice, Chow and Saad [10] suggested using the lower bound $\|\boldsymbol{M}^{-1}\boldsymbol{e}\|_{\infty}$ for $\|\boldsymbol{M}^{-1}\|_{\infty}$ where $\boldsymbol{e}$ is the vector of all ones. This has proved helpful

for detecting preconditioner instability in some situations, and especially indefinite systems [8].

The quantity known as 'preconditioner stability,' $\|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}}$, is empirically the most reliable indicator of preconditioner performance from this class of forecasts. This is especially true among many non-symmetric problems or problems which are far from diagonally dominant [8]. Unfortunately, prior work has suggested that computing preconditioner stability is 'impractical' [7] for effectively the same reason as why $\|\boldsymbol{M}^{-1}\|_{\infty}$ was considered impractical to compute.

In spite of this, Tyrtyshnikov's influential paper [30] gave a fast algorithm to solve the harder problem considered in this paper of determining the minimal-stability preconditioner over the special class of *circulant* matrices. This algorithm relied heavily on properties of these matrices.

Our results in this paper will suggest that even for generic $\boldsymbol{A}$ and $\boldsymbol{M}$, one can get an arbitrarily accurate estimate of most of these quantities previously thought to be unusable for preconditioner selection. That said, it is crucial to note that the prior impracticality assessment is completely valid in a certain regard: no deterministic algorithms, as we also show, could possibly even approximate the preconditioner stability, for example. Thus an algorithm must fail at some of these tasks with some nonzero probability.

**1.3. Overview.** Section 2 motivates the need for a randomized algorithm for stability estimation with theory, responds with a sketching-based solution, and uses it to create and analyze a method for preconditioner selection. This theory is empirically confirmed in Section 3 where we apply the primary preconditioner selection algorithm from Section 2 to solving generic real-world systems (Section 3.1) and creating more robust preconditioned solvers for kernel regression (Section 3.2.) After this we conclude in Section 4 and discuss open problems that could provide an avenue for future work.

**1.3.1. Notation.** Boldface letters like $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{b}$ denote vectors while their uppercase analogues such as $\boldsymbol{A}, \boldsymbol{P}, \boldsymbol{Q}$ denote matrices. Such matrices and vectors can be either random or deterministic. The underlying scalar field $\mathbb{F}$ is either the real numbers $\mathbb{R}$ or the complex numbers $\mathbb{C}$ unless otherwise specified. The adjoint of a matrix is denoted by $\boldsymbol{A}^*$. The inner product of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is denoted as $\boldsymbol{x}^*\boldsymbol{y}$. The expectation of a random variable will be denoted with $\mathbb{E}$, and the probability of an event $E$ is written $\mathbb{P}(E)$. The norms $\|\boldsymbol{A}\|_{\mathsf{F}}$ and $\|\boldsymbol{x}\|_2$ represent the Frobenius and Euclidean norms, respectively. The condition number $\kappa(\boldsymbol{A}) = \sigma_{\max}(\boldsymbol{A})/\sigma_{\min}(\boldsymbol{A})$ is the $\ell^2$ condition number. The notation $a_n \sim b_n$ represents $\lim_{n \to \infty} \frac{a_n}{b_n} = 1$. $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}_{\mathbb{R}}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\mathcal{N}_{\mathbb{C}}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ are normal and circularly-symmetric complex normal random variables, respectively.

**2. Algorithms.** This section contains all of our core theoretical results and algorithms. In Section 2.1 we show that the only algorithms which can possibly estimate preconditioner stability must be randomized. The follow-up question of whether randomization can indeed work is answered in the affirmative in Section 2.2, where we show that a slight adaptation of a well-known sketching-based algorithm for computing Schatten norms perfectly fits our realistic access model to our preconditioner $\boldsymbol{M}$ and matrix $\boldsymbol{A}$. Once we have a good estimator of preconditioner stability, a natural method for selecting the candidate preconditioner with minimal stability criterion presents itself in Section 2.3. It turns our that our algorithm can be trivially parallelized, and a testament to this fact is given in Section 2.3.1. In Section 2.4 we

take advantage of highly informative results from the literature on trace estimation to provide useful approximation guarantees and runtime bounds for the previously presented algorithms. Section 2.4.1 returns the focus to Theorem 2.2 of Section 2.4, showing that the leading constant is tight and providing a more concise resolution of a conjecture from [5] than the more generalized result from [33]. In that section, we also comment that no randomized algorithm for estimating preconditioner stability which has access to matrix-vector products of the form $(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\boldsymbol{z}$ could possibly do better asymptotically than Algorithm 1 by relying on this result [33, Thm. 3] from the trace estimation literature. Finally, this conversation is wrapped up in Section 2.4.2, where our bounds from Section 2.4 are utilized to provide a theoretical improvement to Algorithm 2.

**2.1. Randomization is Necessary to Compute Preconditioner Stability.** This paper provides a simple randomized algorithm which can accurately estimate the preconditioner stability $\|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}}$ in time faster than running a constant number of iterations of preconditioned conjugate gradients with the matrix $\boldsymbol{A}$ and preconditioner $\boldsymbol{M}$. Through incorporating randomness, however, we must accept that the algorithm fails with some probability. This failure probability can be made arbitrarily small, but it would still be advantageous (for example, in mission-critical applications) to provide a deterministic algorithm for the same task with a comparable approximation guarantee. The purpose of this section is to crush that latter hope, and the following theorem does just that.

THEOREM 2.1. *Fix some $0 \le \epsilon < 1$. Suppose we have a deterministic algorithm which takes as input an arbitrary positive semi-definite matrix $\boldsymbol{A} \in \mathbb{F}^{d\times d}$ and positive definite matrix $\boldsymbol{M} \in \mathbb{F}^{d\times d}$, and returns an estimate $\mathrm{ALG}(\boldsymbol{A}, \boldsymbol{M})$ satisfying*

$$(1-\epsilon)\|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}} \le \mathrm{ALG}(\boldsymbol{A}, \boldsymbol{M}) \le (1+\epsilon)\|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}}$$

*after sequentially querying and observing matrix vector multiplies of the form $(\boldsymbol{M}^{-1}\boldsymbol{q}_i, \boldsymbol{A}\boldsymbol{q}_i)$ for $i = 1, 2, \ldots, k$ where $k$ is a constant depending only on $d$ and $\epsilon$. Then $k \ge d$.*

*Proof.* Fix $\boldsymbol{M} = \boldsymbol{I}$ for the remainder of the proof. Suppose to the contrary that $k = d - 1$ suffices to compute $\mathrm{ALG}(\boldsymbol{A}, \boldsymbol{M})$. Let $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_{d-1}$ be the query vectors used by the algorithm in the case that $(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\boldsymbol{q}_i$ always returns $\boldsymbol{0}$. Write $\boldsymbol{P}$ for the orthogonal projection onto $\mathrm{span}\{\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_{d-1}\}$. Both of the positive semi-definite matrices $\boldsymbol{A} = \boldsymbol{I}$ and $\boldsymbol{A} = \boldsymbol{P}$ will return $(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{I})\boldsymbol{q}_i = (\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{P})\boldsymbol{q}_i = \boldsymbol{0}$ uniformly over $i = 1, 2, \ldots, d - 1$, and thus since the algorithm is deterministic the estimated stabilities $\mathrm{ALG}(\boldsymbol{I}, \boldsymbol{M}) = \mathrm{ALG}(\boldsymbol{P}, \boldsymbol{M})$ are equal. But $\boldsymbol{P} \ne \boldsymbol{I}$ since $\boldsymbol{P}$ was an orthogonal projection onto a subspace of dimension strictly less than $d$, and hence

$$0 < (1-\epsilon)\|\boldsymbol{I} - \boldsymbol{P}\|_{\mathsf{F}} \le \mathrm{ALG}(\boldsymbol{P}, \boldsymbol{M}) = \mathrm{ALG}(\boldsymbol{I}, \boldsymbol{M}) \le (1+\epsilon)\|\boldsymbol{I} - \boldsymbol{I}\|_{\mathsf{F}} = 0$$

by our approximation guarantee. This contradiction ensures that we must take $k \ge d$. Since $\boldsymbol{M} = \boldsymbol{I}$, matrix-vector multiply access to $\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}$ is equivalent via a bijection to matrix-vector multiply access to $\boldsymbol{M}^{-1}$ and $\boldsymbol{A}$, hence our strengthened statement of the result.                                                                           □

Of course, using $k = d$ queries suffices to achieve no error at all, and so the above lower bound is tight:

$$(2.1) \qquad \|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}} = \left(\sum_{i=1}^{d} \|(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\boldsymbol{e}_i\|_2^2\right)^{1/2}$$

---

**Algorithm 1:** Stab($\boldsymbol{A}, \boldsymbol{M}, k$): Estimates the stability of the preconditioner $\boldsymbol{M}$ in $\sim 3dk + kT_m + 2k\,\text{NNZ}(\boldsymbol{A})$ floating point operations when $\mathbb{F} = \mathbb{R}$ or $\sim 6dk + kT_m + 8k\,\text{NNZ}(\boldsymbol{A})$ flops when $\mathbb{F} = \mathbb{C}$, where $T_m$ is the number of flops needed to compute $\boldsymbol{M}^{-1}\boldsymbol{b}$ for an arbitrary $\boldsymbol{b} \in \mathbb{F}^d$.

---

**Data:** A matrix $\boldsymbol{A} \in \mathbb{F}^{d \times d}$, preconditioner $\boldsymbol{M} \in \mathbb{F}^{d \times d}$, and an accuracy parameter $k \in \{1, 2, \ldots\}$.
**Result:** A estimate of the preconditioner stability $\|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}}$.
Form a matrix $\boldsymbol{Q} = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_k]$ with independent columns $\boldsymbol{q}_i \sim \mathcal{N}_{\mathbb{F}}(\boldsymbol{0}, \frac{1}{k}\boldsymbol{I}_d)$.
Construct the sketch $\boldsymbol{S} = (\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\boldsymbol{Q}$ via its columns $\boldsymbol{q}_i - \boldsymbol{M}^{-1}(\boldsymbol{A}\boldsymbol{q}_i)$.
Return $\|\boldsymbol{S}\|_{\mathsf{F}}$.

---

where $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_d$ is any orthonormal basis for $\mathbb{F}^d$. Also, note that the condition that $\boldsymbol{A}$ and $\boldsymbol{M}$ be positive semi-definite gives a stronger result than if they were allowed to be arbitrary matrices.

In order to put Theorem 2.1 into better context, recall that the dominant cost of an iteration of preconditioned conjugate gradients [16, Alg. 11.5.1] is (a) computing $\boldsymbol{A}\boldsymbol{y}$ for a vector $\boldsymbol{y}$, and (b) computing $\boldsymbol{M}^{-1}\boldsymbol{z}$ for a vector $\boldsymbol{z}$. Thus Theorem 2.1 says roughly that in the time it takes to even approximate $\|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}}$ deterministically, one can solve a system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ exactly (in exact arithmetic) by running the conjugate gradients algorithm for $d$ iterations. Since our whole goal of computing $\|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}}$ is to forecast how well $\boldsymbol{M}$ would do as a preconditioner for solving the system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, this means that any deterministic algorithm for this task is, in general, impractical for this task.

**2.2. Computing Preconditioner Stability via Randomization.** Now we will demonstrate that, unlike the deterministic case, randomization makes it entirely practical to compute preconditioner stability. To see why this is intuitive, let $\boldsymbol{q}$ be a standard (real or complex) Gaussian vector. Then

$$(2.2) \qquad \|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}}^2 = \text{tr}\big((\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})^*(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\big)$$

$$(2.3) \qquad = \text{tr}\big((\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})^*(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\mathbb{E}\boldsymbol{q}\boldsymbol{q}^*\big)$$

$$(2.4) \qquad = \mathbb{E}\,\text{tr}\big(\boldsymbol{q}^*(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})^*(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\boldsymbol{q}\big)$$

$$(2.5) \qquad = \mathbb{E}\|(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\boldsymbol{q}\|_2^2$$

by the linearity of expectation, the cyclic property of the trace, and the fact that $\mathbb{E}\boldsymbol{q}\boldsymbol{q}^* = \boldsymbol{I}$. Thus, if $\boldsymbol{q}_i$ are independent standard normal vectors for all $i = 1, 2, \ldots, k$, the Monte-Carlo squared stability estimate

$$(2.6) \qquad S^2 = \frac{1}{k}\sum_{i=1}^{k}\|(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\boldsymbol{q}_i\|_2^2 \to \|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}}^2$$

almost surely as $k \to \infty$ by the strong law of large numbers. We can rewrite the above estimator as $S = \|(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\boldsymbol{Q}\|_{\mathsf{F}}$ where $\boldsymbol{Q}$ is a matrix with independent and identically distributed elements $\boldsymbol{Q}_{ii} \sim \mathcal{N}(0, \frac{1}{k})$. This stability estimation algorithm for $S = \sqrt{S^2} \approx \|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}}$ is given as Algorithm 1. In Algorithm 1, we adjust whether $\boldsymbol{q}$ is real or complex depending on $\mathbb{F}$ for analysis reasons that will be apparent later in Section 2.4.

---

**Algorithm 2:** Returns an approximately optimal preconditioner in the Tyr-
tyshkinov sense among $n$ candidates with strictly fewer floating point oper-
ations than running $k$ iterations of preconditioned conjugate gradients [16,
Alg. 11.5.1] with each of the $n$ preconditioners.

---

> **Data:** A matrix $\boldsymbol{A} \in \mathbb{F}^{d \times d}$, $n$ candidate preconditioners $\boldsymbol{M}_1$,
> $\boldsymbol{M}_2, \ldots, \boldsymbol{M}_n \in \mathbb{F}^{d \times d}$, and an accuracy parameter $k \in \{1, 2, \ldots\}$.
> **Result:** A preconditioner $\boldsymbol{M}_i$ which approximately minimizes the stability
> criterion $\|\boldsymbol{I} - \boldsymbol{M}_j^{-1} \boldsymbol{A}\|_{\mathsf{F}}$ over $j = 1, \ldots, n$.
> Compute a stability estimate $S_j = \mathrm{Stab}(\boldsymbol{A}, \boldsymbol{M}_j, k)$ for each $j = 1, 2, \ldots, n$.
> Return an arbitrary $\boldsymbol{M}_i$ with $S_i = \min\limits_{1 \leq j \leq n} S_j$.

---

It is important to note that the mathematical foundations of the above algorithm
are not novel. It is equivalent in exact arithmetic to applying the trace estimators in
[25] to the matrix $(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})^*(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})$ and then taking the square root. It is
also a simplification of the Schatten-2 norm estimator in [34, Thm. 69] (relayed from
[22]) applied to $\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}$. The reason we include Algorithm 1 is not because of its
mathematical novelty but because of its observational novelty: sketching algorithms
using the matrix-vector multiply access model are a perfect fit for interrogating the
matrices $\boldsymbol{M}^{-1}$ and $\boldsymbol{A}$ in the context of iterative methods for solving linear systems,
since this kind of access to $\boldsymbol{M}^{-1}$ and $\boldsymbol{A}$ are precisely what make that kind of access
practical.

Of course, the presentation thus far does not help us choose how large the accuracy
parameter $k$ should be. To that end, Theorem 2.2 in Section 2.4 presents a tight
upper bound on the necessary $k$ to achieve a given error bound with high probability,
although in practice just setting $k = 10$ appears sufficient in many situations of
interest – see Section 3.

**2.3. Randomized Algorithm for Selecting the 'Best' Preconditioner.**
Once we have a practical way to compute preconditioner stability, a trivial algorithm
for picking the preconditioner among $n$ candidates $\boldsymbol{M}_1, \boldsymbol{M}_2, \ldots, \boldsymbol{M}_n$ becomes natu-
ral. Namely, we can compute estimates $S_i \approx \|\boldsymbol{I} - \boldsymbol{M}_i^{-1}\boldsymbol{A}\|_{\mathsf{F}}$ for $i = 1, 2, \ldots, n$ and
then just return the preconditioner $\boldsymbol{M}_i$ for which $S_i$ is minimized. This is presented
as Algorithm 2. As we mentioned in the previous section, theoretical advice on how
to pick $k$ will be given in Section 2.4. An improvement to this algorithm in the case
there is a clear winner, relying on those analytical bounds, is included in Section 2.4.2.

We note that the sketching matrix $\boldsymbol{Q}$ can be re-used when computing the stability
estimates $S_j$ in Algorithm 1. This is done in all our computational experiments in
Section 3, and reduces the number of normal variates one needs to simulate from $ndk$
to $dk$. Reuse does not affect our theoretical upper bound presented as Theorem 2.3.

**2.3.1. Parallelization.** A convenient aspect of sketching based algorithms like
Algorithm 2 is that they can be parallelized extremely easily. For instance, sup-
pose we are trying to pick the minimal stability preconditioner among $n$ candidates
$\boldsymbol{M}_1, \boldsymbol{M}_2, \ldots, \boldsymbol{M}_n$, and have $n$ processors $P_j$, $j = 1, 2, \ldots, n$, which have access to $\boldsymbol{A}$
and $\boldsymbol{M}_j$, respectively. Then we can compute each stability estimate $S_j$ in parallel at
processor $P_j$. Ignoring communication costs (which are a genuine concern in prac-
tice,) this would bring the runtime of the algorithm down to computing $k$ steps of
preconditioned conjugate gradients with $\boldsymbol{A}$ and the most computation-intensive (in

terms of matrix-vector multiply access to $\boldsymbol{M}^{-1}$) preconditioner $\boldsymbol{M}_j$.

Taken to the extreme, one could similarly parallelize Algorithm 2 over $nk$ processors $P_{ij}$, $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, k$, assuming each $P_{ij}$ had access to $\boldsymbol{A}$ and the candidate preconditioner $\boldsymbol{M}_j$. Each processor $P_{ij}$ would need to compute and return $s_{ij} = \|(\boldsymbol{I} - \boldsymbol{M}_j^{-1}\boldsymbol{A})\boldsymbol{q}_i\|_2^2$ where $\boldsymbol{q}_i$ is an independently sampled standard normal vector. Then in parallel for all $i = 1, 2, \ldots, n$ processor $P_{i1}$ could compute $S_i^2 = \frac{1}{k}\sum_{j=1}^{k} s_{ij}$, at which point we could use processor $P_{11}$ to compute $i$ such that $S_i^2$ is minimal and return the corresponding $\boldsymbol{M}_i$. Ignoring communication costs again, this algorithm take fewer floating point operations than running *one* iteration of preconditioned conjugate gradients with $\boldsymbol{A}$ and the most computation-intensive preconditioner $\boldsymbol{M}_j$, plus $k$ flops that were used to turn the $s_{ij}$ into the estimate $S_i$.

**2.4. Approximation Guarantees and Runtime Bounds.** This section details runtime bounds and approximation guarantees for Algorithms 1 and 2, as well as using those bounds to derive a theoretical improvement to Algorithm 2.

To start, we will give the following theorem, which says that to estimate preconditioner stability up to a $1 \pm \epsilon$ multiplicative factor with failure probability at most $\delta$, one may take $k = \mathcal{O}(\frac{1}{\epsilon^2} \log \frac{1}{\delta})$ in Algorithm 1. This is (in contrast to the deterministic case) completely independent of the underlying dimension. Theorem 2.2 result sharpens an analogous bound for trace estimators given in [25, Thm. 3], largely following the proof structure of Theorem 1 from that work. We bring the leading constant from a 8 to 4 as $\epsilon \to 0$ and allow for complex matrices. This Theorem is included as a result in its own right primarily since we can even show that the leading constant is tight; see Section 2.4.1.

THEOREM 2.2. *Let $\boldsymbol{M}$ and $\boldsymbol{A}$ be arbitrary matrices in $\mathbb{F}^{d \times d}$ where $\boldsymbol{M}$ is invertible. If $\epsilon$ and $\delta$ are positive and less than one, taking $k \geq \frac{12}{\epsilon^2(3-2\epsilon)} \log \frac{2}{\delta}$ ensures that in exact arithmetic the estimate $\mathrm{Stab}(\boldsymbol{A}, \boldsymbol{M}, k) \in \mathbb{R}$ of Algorithm 1 satisfies*

$$\sqrt{1-\epsilon}\, \|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}} \leq \mathrm{Stab}(\boldsymbol{A}, \boldsymbol{M}, k) \leq \sqrt{1+\epsilon}\, \|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}}.$$

*with probability at least $1 - \delta$. In particular, if $\epsilon \leq 1/2$, then the simpler condition $k \geq \frac{6}{\epsilon^2} \log \frac{1}{\delta}$ ensures this same approximation guarantee.*

*Proof.* Unitarily diagonalize $(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})^*(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})$ as $\boldsymbol{U}^*\Lambda\boldsymbol{U}$ for some unitary $\boldsymbol{U}$ and non-negative diagonal matrix $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$. By the rotation invariance of the Gaussian and the fact that $\boldsymbol{U}$ is orthogonal when $\mathbb{F} = \mathbb{R}$, $\|(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\boldsymbol{q}_i\|_2$ is equal in distribution to $\|\Lambda^{1/2}\boldsymbol{q}_i\|_2$ in both the real and complex cases. This implies $\mathrm{Stab}(\boldsymbol{A}, \boldsymbol{M}, k)$ is equal in distribution to $\frac{1}{k}\sum_{i=1}^{k} \boldsymbol{q}_i^*\Lambda\boldsymbol{q}_i = \frac{1}{k}\sum_{j=1}^{d} \lambda_j \sum_{i=1}^{k} |\boldsymbol{q}_{ij}|^2$ and we can focus on the latter, simpler quantity. By Markov's inequality, for any $t > 0$,

$$(2.7) \qquad \mathbb{P}\big(\mathrm{Stab}(\boldsymbol{A}, \boldsymbol{M}, k)^2 > (1+\epsilon)\|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}}^2\big)$$

$$(2.8) \qquad = \mathbb{P}\left(\sum_{j=1}^{d} \lambda_j \sum_{i=1}^{k} |\boldsymbol{q}_{ij}|^2 \geq k(1+\epsilon)\sum_{j=1}^{d} \lambda_j\right)$$

$$(2.9) \qquad \leq e^{-(1+\epsilon)kt}\mathbb{E}\exp\left(\sum_{j=1}^{d} \frac{\lambda_j}{\sum_{j=1}^{d} \lambda_j} \sum_{i=1}^{k} t|\boldsymbol{q}_{ij}|^2\right).$$

Jensen's inequality reduces

$$(2.10) \qquad \mathbb{E}\exp\left(\sum_{j=1}^{d} \frac{\lambda_j}{\sum_{j=1}^{d} \lambda_j} \sum_{i=1}^{k} t|\boldsymbol{q}_{ij}|^2\right) \leq \prod_{i=1}^{k} \mathbb{E}e^{t|\boldsymbol{q}_{i1}|^2} \leq (1-2t)^{-\frac{k}{2}}$$

so long as $t < 1/2$. Note that the last inequality covers both real and complex $\mathbb{F}$, and results from the $\mathbb{F} = \mathbb{C}$ moment generating function $\mathbb{E}e^{t|\boldsymbol{q}_{i1}|^2} = (1-t)^{-1} \leq (1-2t)^{-1/2}$ being bounded above uniformly by the real case for this range of $t$. Taking $t = \frac{1}{2}\frac{\epsilon}{1+\epsilon} \in (0, 1/2)$ in Equation 2.9 gives

$$(2.11) \quad \mathbb{P}\big(\mathrm{Stab}(\boldsymbol{A}, \boldsymbol{M}, k)^2 > (1+\epsilon)\|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}}^2\big) \leq \big((1+\epsilon)e^{-\epsilon}\big)^{m/2} < e^{-\frac{m}{2}\left(\frac{\epsilon^2}{2} - \frac{\epsilon^3}{3}\right)}$$

via the scalar inequality $(1 + \epsilon)e^{-\epsilon} < e^{\frac{\epsilon^3}{3} - \frac{\epsilon^2}{2}}$ for all $\epsilon > 0$. The same argument for the lower tail, instead taking $t = \frac{1}{2}\frac{\epsilon}{1-\epsilon}$, gives

$$(2.12) \qquad \mathbb{P}\big(\mathrm{Stab}(\boldsymbol{A}, \boldsymbol{M}, k)^2 < (1-\epsilon)\|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}}^2\big) < e^{-\frac{m}{2}\left(\frac{\epsilon^2}{2} - \frac{\epsilon^3}{3}\right)}$$

as well. A union bound provides the desired result. $\qquad\square$

Using Theorem 2.2 we are able to prove an approximation guarantee for Algorithm 2 via a union bound. In particular, to achieve an $\epsilon$-multiplicative approximation to the best of $n$ candidate preconditioners with probability at least $1 - \delta$ we can take $k = \mathcal{O}(\frac{1}{\epsilon^2} \log \frac{n}{\delta})$, again independent of the underlying dimension. This dependence on $n$ is quite weak, especially since in realistic applications we would only expect to have at most, say, fifty candidate preconditioners.

THEOREM 2.3. *Let $\boldsymbol{A} \in \mathbb{F}^{d \times d}$ be an arbitrary matrix, and $\boldsymbol{M}_1, \boldsymbol{M}_2, \ldots, \boldsymbol{M}_n \in \mathbb{F}^{d \times d}$ be invertible candidate preconditioners for $\boldsymbol{A}$. If $\epsilon$ and $\delta$ are positive and less than one, taking $k \geq \frac{12}{\epsilon^2(3-2\epsilon)} \log \frac{2n}{\delta}$ ensures that the preconditioner $\boldsymbol{M}_i$ returned by Algorithm 2 satisfies*

$$\|\boldsymbol{I} - \boldsymbol{M}_i^{-1}\boldsymbol{A}\|_{\mathsf{F}} \leq \sqrt{\frac{1+\epsilon}{1-\epsilon}} \min_{1 \leq j \leq n} \|\boldsymbol{I} - \boldsymbol{M}_j^{-1}\boldsymbol{A}\|_{\mathsf{F}}$$

*with probability at least $1 - \delta$. In particular, if $\epsilon < 1/2$ the simpler condition $k \geq \frac{11}{\epsilon^2} \log \frac{2n}{\delta}$ ensures*

$$\|\boldsymbol{I} - \boldsymbol{M}_i^{-1}\boldsymbol{A}\|_{\mathsf{F}} \leq (1 + \epsilon) \min_{1 \leq j \leq n} \|\boldsymbol{I} - \boldsymbol{M}_j^{-1}\boldsymbol{A}\|_{\mathsf{F}}$$

*with probability at least $1 - \delta$.*

*Proof.* Start by fixing any $j \in \{1, 2, \ldots, n\}$. If we take $k \geq \frac{12}{\epsilon^2(3-2\epsilon)} \log \frac{2n}{\delta}$, Theorem 2.2 ensures that

$$(2.13) \qquad \sqrt{1-\epsilon}\,\|\boldsymbol{I} - \boldsymbol{M}_j^{-1}\boldsymbol{A}\|_{\mathsf{F}} \leq \mathrm{Stab}(\boldsymbol{A}, \boldsymbol{M}_j, k) \leq \sqrt{1+\epsilon}\,\|\boldsymbol{I} - \boldsymbol{M}_j^{-1}\boldsymbol{A}\|_{\mathsf{F}},$$

except with probability at most $\frac{\delta}{n}$. In particular, if we unfix $j$ the probability at least one of the $\mathrm{Stab}(\boldsymbol{A}, \boldsymbol{M}_j, k)$ does not satisfy Equation 2.13 is at most $\sum_{j=1}^{n} \frac{\delta}{n} = \delta$ by a union bound. (Note that we did not need independence of the estimates $\mathrm{Stab}(\boldsymbol{A}, \boldsymbol{M}_j, k)$ here; this is why reusing the sketching matrix $\boldsymbol{Q}$ is valid.) Thus with probability at least $1 - \delta$ all estimates $\mathrm{Stab}(\boldsymbol{A}, \boldsymbol{M}_j, k)$ satisfy Equation 2.13 simultaneously.

Write $\boldsymbol{M}_i$ for the candidate preconditioner returned by Algorithm 2, and write $\boldsymbol{M}_\star$ for a candidate preconditioner which satisfies

$$(2.14) \qquad\qquad \|\boldsymbol{I} - \boldsymbol{M}_\star^{-1}\boldsymbol{A}\|_{\mathsf{F}} = \min_{1 \leq j \leq n} \|\boldsymbol{I} - \boldsymbol{M}_j^{-1}\boldsymbol{A}\|_{\mathsf{F}}.$$

Then since the *estimate* of the stability of $\boldsymbol{M}_i$ was at most that of $\boldsymbol{M}_\star$ by minimality, the simultaneous bounds of Equation 2.13 give

$$\sqrt{1-\epsilon}\|\boldsymbol{I} - \boldsymbol{M}_i^{-1}\boldsymbol{A}\|_{\mathsf{F}} \leq \mathrm{Stab}(\boldsymbol{A}, \boldsymbol{M}_i, k) \leq \mathrm{Stab}(\boldsymbol{A}, \boldsymbol{M}_\star, k) \leq \sqrt{1+\epsilon}\|\boldsymbol{I} - \boldsymbol{M}_\star^{-1}\boldsymbol{A}\|_{\mathsf{F}}$$

except with probability at most $\delta$. Rearranging the inequality gives the desired result after substituting Equation 2.14.

The final simplified bound results from the scalar inequality $\sqrt{\frac{1+\epsilon}{1-\epsilon}} \leq 1 + \frac{4}{3}\epsilon$ when $0 \leq \epsilon < 2/5$ and simple algebraic manipulation. $\square$

**2.4.1. The Constant in Theorem 2.2 is Tight.** Most of the theory presented in this paper relies on Theorem 2.2 to create more sophisticated bounds. Since Algorithm 1 is at its core a repurposing of a trace estimator using only matrix vector products, the work [33] applies and ensures that no randomized, adaptive algorithm for estimating the stability $\|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}}^2 = (\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})^*(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})$ could possibly use asymptotically fewer matrix-vector multiplies so long as the algorithm only has access to $\|(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\boldsymbol{q}\|_2$ for query vectors $\boldsymbol{q}$. In this sense, Algorithm 1 is optimal.

The theoretically-inclined practitioner, however, also cares about knowing the optimality of our *analysis* in Theorem 2.2. The following Theorem says that our analysis in Theorem 2.2 is asymptotically tight even up to the leading effective constant $12/(3-2\epsilon)$ which tends to 4 for small $\epsilon$. In the proof, $W(x) = \log x - \log\log x + o(1) = \Theta(\log x)$ as $x \to \infty$ is the Lambert-$W$ function [19].

THEOREM 2.4. *Fix some $0 < \delta < 1/10$. For any underlying dimension $d$, there exists a positive semi-definite matrix $\boldsymbol{A} \in \mathbb{R}^{d \times d}$, positive definite matrix $\boldsymbol{M} \in \mathbb{R}^{d \times d}$, and some $\epsilon_0 > 0$ so that for any $0 < \epsilon < \epsilon_0$, taking $k = \lfloor \frac{4}{\epsilon^2}\log\frac{1}{\sqrt{8\pi\delta}} - \frac{2}{\epsilon^2}\log\log\frac{1}{\sqrt{8\pi\delta}} \rfloor$ guarantees the stability estimate $\mathrm{Stab}(\boldsymbol{A}, \boldsymbol{M}, k)$ returned by Algorithm 1 fails to satisfy the equation*

$$\sqrt{1-\epsilon}\|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}} \leq \mathrm{Stab}(\boldsymbol{A}, \boldsymbol{M}, k) \leq \sqrt{1+\epsilon}\|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}}$$

*with probability at least $\delta$.*

*Proof.* If $Z \sim \mathcal{N}(0,1)$ is a standard normal random variable then

$$(2.15) \qquad \mathbb{P}(|Z| \geq t) = 2\mathbb{P}(Z > t) > \sqrt{\frac{2}{\pi}}\frac{t}{t^2+1}e^{-t^2/2} \geq \frac{1}{\sqrt{2\pi}}\frac{1}{t}e^{-t^2/2}$$

by [17] for all $t \geq 1$. Setting the right hand side of Inequality 2.15 to $2\delta$ and solving gives

$$(2.16) \qquad \mathbb{P}\big(|Z| \geq \sqrt{W(8^{-1}\pi^{-1}\delta^{-2})}\big) > 2\delta$$

whenever $\sqrt{W(8^{-1}\pi^{-1}\delta^{-2})} \geq 1$, which is satisfied when $0 < \delta \leq 1/10$.

Now let $\boldsymbol{A} = \boldsymbol{I} - \boldsymbol{e}_1\boldsymbol{e}_1^*$ and $\boldsymbol{M} = \boldsymbol{I}$, where $\boldsymbol{e}_1$ is the first standard basis vector. We can observe that

$$(2.17) \qquad \|(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\boldsymbol{q}\|_2^2 = \|\boldsymbol{e}_1(\boldsymbol{e}_1^*\boldsymbol{q})\|_2^2 = \boldsymbol{q}_1^2 \sim \chi^2$$

if $\boldsymbol{q}$ is a standard Gaussian vector. In particular, the standard deviation of $\|(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\boldsymbol{q}\|_2^2$ is $\sigma = \sqrt{2}$. Thus since $\mathrm{Stab}(\boldsymbol{A}, \boldsymbol{M}, k)^2$ is a sample average of independent

copies of these random variables, fixing $k = \lfloor \frac{2}{\epsilon^2} W(\frac{1}{8\pi\delta^2}) \rfloor$ ensures

$$(2.18) \qquad \mathbb{P}(|\operatorname{Stab}(\boldsymbol{A}, \boldsymbol{M}, k)^2 - 1| > \epsilon)$$

$$(2.19) \qquad = \mathbb{P}(\tfrac{\sqrt{k}}{\sigma}|\operatorname{Stab}(\boldsymbol{A}, \boldsymbol{M}, k)^2 - 1| \geq \tfrac{\sqrt{k}\epsilon}{\sigma})$$

$$(2.20) \qquad \geq \mathbb{P}\big(\tfrac{\sqrt{k}}{\sigma}|\operatorname{Stab}(\boldsymbol{A}, \boldsymbol{M}, k)^2 - 1| \geq \sqrt{W(8^{-1}\pi^{-1}\delta^{-2})}\big)$$

$$(2.21) \qquad \to \mathbb{P}\big(|Z| \geq \sqrt{W(8^{-1}\pi^{-1}\delta^{-2})}\big) > 2\delta$$

by the Central Limit Theorem [31, Thm. 1.3.2] and Equation 2.16 as $k \to \infty$. This implies the existence of an $\epsilon_0$ so that $0 < \epsilon < \epsilon_0$ ensures the relation

$$(2.22) \qquad \sqrt{1-\epsilon}\|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}} \leq \operatorname{Stab}(\boldsymbol{A}, \boldsymbol{M}, k) \leq \sqrt{1+\epsilon}\|\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A}\|_{\mathsf{F}}$$

fails with probability *at least* $\delta$ under our relation defining $k$. The simpler condition on $k$ given in the statement of this result follows from the bound $W(x) \geq \log(x) - \log\log(x)$ for all $x \geq e$ from [19, Thm. 2]. □

We point out that the above proof gives another confirmation of the conjecture in [5] regarding the true asymptotics of the Gaussian trace estimator. The work in [33] confirmed this conjecture to be true, but did so as a corollary of a general result regarding lower bounds for trace estimation algorithms. Our result above is much more direct.

**2.4.2. An Improvement in the Presence of a Clear Winner.** Algorithm 2 is simple to implement and works well for selecting preconditioners of minimal stability, as we shall see in Section 3. Nevertheless, if we are selecting between preconditioners where some are clearly worse than the optimal preconditioner in terms of stability, our method seems excessive. Intuitively, we should be able to tell that terrible preconditioners will not be optimal with very rudimentary information. Algorithm 3 presents such a revision to Algorithm 2, iteratively refining the stability estimates we have and filtering out any preconditioners as soon as we can be confident they will not be optimal. Note that the algorithm crucially relies on the bounds from Section 2.4.

We can prove that Algorithm 3 is actually an improvement over Algorithm 2 by making an anti-concentration assumption about the input stabilities.

THEOREM 2.5. *Let $\boldsymbol{A} \in \mathbb{F}^{d \times d}$ be an arbitrary matrix, $\boldsymbol{M}_1, \boldsymbol{M}_2, \ldots, \boldsymbol{M}_n \in \mathbb{F}^{d \times d}$ be invertible candidate preconditioners for $\boldsymbol{A}$, $0 < \epsilon < 1/2$, and $0 < \delta < 1$. Denoting $i^\star \in \arg\min_{1 \leq j \leq n} \|\boldsymbol{I} - \boldsymbol{M}_j^{-1}\boldsymbol{A}\|_{\mathsf{F}}$, we will write*

$$F(t) = \frac{1}{n}\left|\left\{ j : j \in \{1, 2, \ldots, n\} \text{ and } \frac{\|\boldsymbol{I} - \boldsymbol{M}_j^{-1}\boldsymbol{A}\|_{\mathsf{F}}}{\|\boldsymbol{I} - \boldsymbol{M}_{i^\star}^{-1}\boldsymbol{A}\|_{\mathsf{F}}} \leq 1 + t \right\}\right|$$

*for the (shifted) cumulative distribution function of the input relative stabilities. If $F(t) \leq ct$ uniformly over $t \in [\epsilon/2, 2]$ for some positive constant $c$, then Algorithm 3 returns a preconditioner $\boldsymbol{M}_i$ satisfying*

$$\|\boldsymbol{I} - \boldsymbol{M}_i^{-1}\boldsymbol{A}\|_{\mathsf{F}} \leq \sqrt{\frac{1+\epsilon}{1-\epsilon}} \min_{1 \leq j \leq n} \|\boldsymbol{I} - \boldsymbol{M}_j^{-1}\boldsymbol{A}\|_{\mathsf{F}}.$$

*with probability at least $1-\delta$ using strictly fewer floating point operations than running $24n(1 + \frac{2c}{\epsilon})\log\frac{2n}{\delta} + 24n(1 + \frac{2c}{\epsilon})\log\log_2\frac{2}{\epsilon}$ iterations of the preconditioned conjugate gradients algorithm in $\boldsymbol{A}$ with the most expensive preconditioner $\boldsymbol{M}_j$ in terms of the number of floating point operations required to compute $\boldsymbol{M}_j^{-1}\boldsymbol{y}$ for input vectors $\boldsymbol{y}$.*

---

**Algorithm 3:** An improvement to Algorithm 2 when there is a relatively clear winner among the candidate preconditioners.

---

**Data:** A matrix $\boldsymbol{A} \in \mathbb{F}^{d \times d}$, $n$ candidate preconditioners $\boldsymbol{M}_1$, $\boldsymbol{M}_2, \ldots, \boldsymbol{M}_n \in \mathbb{F}^{d \times d}$, an accuracy parameter $0 < \epsilon < \frac{1}{2}$ and an acceptable failure probability $0 < \delta < 1$.

**Result:** A preconditioner $\boldsymbol{M}_i$ for which the stability criterion $\|\boldsymbol{I} - \boldsymbol{M}_i^{-1}\boldsymbol{A}\|_{\mathsf{F}}$ is an $\epsilon$-multiplicative approximation to the minimum possible among the candidate preconditioners, except with probability at most $\delta$.

$\epsilon_{\mathsf{cur}} \leftarrow 1$
$P \leftarrow \{1, 2, \ldots, n\}$
$T \leftarrow \lceil \log_2 \frac{1}{\epsilon} \rceil$
**for** $t = 1, 2, \ldots, T$ **do**
$\quad$ $\epsilon_{\mathsf{cur}} \leftarrow \epsilon_{\mathsf{cur}}/2$
$\quad$ $k \leftarrow \frac{6}{\epsilon_{\mathsf{cur}}^2} \log \frac{2T|P|}{\delta}$
$\quad$ $S_i \leftarrow \mathrm{Stab}(\boldsymbol{A}, \boldsymbol{M}_i, k)$ for all $i \in P$
$\quad$ $i^\star = \arg\min_{i \in P} S_i$
$\quad$ $P \leftarrow \left\{ i \in P : S_i \leq S_{i^\star} \sqrt{\frac{1+\epsilon_{\mathsf{cur}}}{1-\epsilon_{\mathsf{cur}}}} \right\}$
**end**
Return $\boldsymbol{M}_{i^\star}$

---

*Proof.* The same Bonferroni-correction argument from the proof of Theorem 2.3 ensures that

$$(2.23) \qquad \sqrt{1-\epsilon_{\mathsf{cur}}}\,\|\boldsymbol{I} - \boldsymbol{M}_i^{-1}\boldsymbol{A}\|_{\mathsf{F}} \leq S_i \leq \sqrt{1+\epsilon_{\mathsf{cur}}}\,\|\boldsymbol{I} - \boldsymbol{M}_i^{-1}\boldsymbol{A}\|_{\mathsf{F}},$$

simultaneously for all $i \in P$ over the course of the algorithm, except with probability at most $\delta$. The rest of the proof will only rely on property 2.23, so everything we say will hold with this same probability.

If $i_t^\star$ is the $i^\star$ set in step $t$ of the algorithm and $i^\star \in \arg\min_{1 \leq i \leq n} \|\boldsymbol{I} - \boldsymbol{M}_i^{-1}\boldsymbol{A}\|_{\mathsf{F}}$ is in $P$ before the filtering at the end of step $t$, Equation 2.23 implies

$$S_{i^\star} \leq \sqrt{1+\epsilon_{\mathsf{cur}}}\|\boldsymbol{I} - \boldsymbol{M}_{i^\star}^{-1}\boldsymbol{A}\|_{\mathsf{F}} \leq \sqrt{\frac{1+\epsilon_{\mathsf{cur}}}{1-\epsilon_{\mathsf{cur}}}}\sqrt{1-\epsilon_{\mathsf{cur}}}\|\boldsymbol{I} - \boldsymbol{M}_{i_t^\star}^{-1}\boldsymbol{A}\|_{\mathsf{F}} \leq \sqrt{\frac{1+\epsilon_{\mathsf{cur}}}{1-\epsilon_{\mathsf{cur}}}}S_{i_t^\star}.$$

Thus, since $i^\star \in P$ initially, we know by induction that $i^\star \in P$ throughout the process of the entire algorithm. Now consider $P$ in the final step $t = T$ of Algorithm 3. Since $i^\star \in P$,

$$(2.24) \qquad \sqrt{1-\epsilon_{\mathsf{cur}}}\|\boldsymbol{I} - \boldsymbol{M}_{i_T^\star}^{-1}\boldsymbol{A}\|_{\mathsf{F}} \leq S_{i_T^\star} \leq S_{i^\star} \leq \sqrt{1+\epsilon_{\mathsf{cur}}}\|\boldsymbol{I} - \boldsymbol{M}_{i^\star}^{-1}\boldsymbol{A}\|_{\mathsf{F}}.$$

Rearranging and realizing that $\epsilon_{\mathsf{cur}} = 2^{-\lceil \log_2 \frac{1}{\epsilon} \rceil} \leq 2^{-\log_2 \frac{1}{\epsilon}} = \epsilon$ at $t = T$ gives our desired approximation guarantee.

Now we will exhibit the runtime bound by bounding $|P|$ at each step of Algorithm 3. We claim that $|P| \leq 4cn2^{-t}$ for all $t = 1, 2, \ldots, T$. To see this, note that if a candidate preconditioner $\boldsymbol{M}_j$ is retained after filtering in any step $t$ of the algorithm,

$$(2.25) \qquad \|\boldsymbol{I} - \boldsymbol{M}_j^{-1}\boldsymbol{A}\|_{\mathsf{F}} \leq \frac{S_j}{\sqrt{1-\epsilon_{\mathsf{cur}}}} \leq \frac{\sqrt{1+\epsilon_{\mathsf{cur}}}}{1-\epsilon_{\mathsf{cur}}}S_{i_t^\star} \leq \frac{1+\epsilon_{\mathsf{cur}}}{1-\epsilon_{\mathsf{cur}}}\|\boldsymbol{I} - \boldsymbol{M}_{i^\star}^{-1}\boldsymbol{A}\|_{\mathsf{F}}.$$

Thus the number of elements in $P$ just after step $t$ in the algorithm is at most $nF(4\epsilon_{\mathsf{cur}}) \leq 4cn2^{-t}$ since $\frac{1+x}{1-x} \leq 1 + 4x$ for $0 \leq x \leq 1/2$. Our runtime bound follows from the sum

$$(2.26) \qquad \sum_{t=1}^{T-1} |P_t| \frac{6}{(2^{-t})^2} \log \frac{2T|P_t|}{\delta} \leq \sum_{t=1}^{T-1} 4cn2^{-t} \frac{6}{(2^{-t})^2} \log \frac{2nT}{\delta} \leq 24cn2^T \log \frac{2nT}{\delta}$$

where $P_t$ is the set $P$ during iteration $t$ of the algorithm. This gives the number of matrix-vector multiplies of the form $(\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\boldsymbol{q}$ used by the algorithm *after* the first step. To see the final form, add on the $24n \log \frac{2nT}{\delta}$ multiplies done during the first iteration $t = 1$ and plug in $T = \lceil \log_2 \frac{1}{\epsilon} \rceil \leq \log_2 \frac{1}{\epsilon} + 1 = \log_2 \frac{2}{\epsilon}$. $\qquad\square$

The anti-concentration condition in Theorem 2.5 intuitively asserts that the stabilities of the preconditioners do not cluster around the minimal stability. This is satisfied, for example, if at most some number $m$ of the candidate preconditioners have stability within a multiplicative factor 3 of the optimal stability. The resulting constant $c = \frac{2m}{n\epsilon}$ gives an asymptotic runtime bound for Algorithm 3 of $\mathcal{O}(n \log \frac{n}{\delta} + n \log \log \frac{1}{\epsilon} + \frac{m}{\epsilon^2} \log \frac{n}{\delta} + \frac{m}{\epsilon^2} \log \log \frac{1}{\epsilon})$, decoupling the linear dependence in $n$ with the polynomial accuracy dependence on $1/\epsilon^2$. Such an improvement is serious when $n$ is moderately large; while this example is contrived many other distributions on input data satisfy the assumptions of Theorem 2.5 with the same constant $c$.

Of course, one would hope that Algorithm 3 does not perform poorly when the input data assumptions made in Theorem 2.5 are not satisfied. For example, this would happen when all preconditioners have extremely similar performance, to the point that even our target accuracy $\epsilon$ cannot distinguish their stabilities. Luckily, a constant $c = \frac{2}{\epsilon}$ always works in Theorem 2.5, so Algorithm 3 never suffers more than a multiplicative $\mathcal{O}(\log \log \frac{1}{\epsilon})$ increase over Algorithm 2 in number of floating point operations needed to select a preconditioner.

**3. Experiments.** The present section will jointly test the hypothesis that preconditioner stability is a good forecast for preconditioner quality along with the performance of our algorithms. This is done by evaluating how well Algorithm 2 can select a candidate preconditioner which minimizes the number of conjugate gradients iterations required to achieve some fixed approximation quality in various situations. In Section 3.1, we detail one experiment of this type for generic real-world systems from the SuiteSparse Matrix Collection [13]. Section 3.2 details our other numerical experiment, where we use Algorithm 2 to contribute improvements to the growing literature on preconditioned solvers for kernel regression problems.

**3.1. Experiments with Sparse Systems.** First we attempt a generic experiment on a collection of real-world sparse linear systems and simple preconditioners, testing how well the preconditioner chosen a-priori by Algorithm 2 compares to the minimal-iterations preconditioner. For the target system $\boldsymbol{Ax} = \boldsymbol{b}$, we fix a sampled $\boldsymbol{b} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ for the entire experiment. The positive definite matrices $\boldsymbol{A}$ are taken from the SuiteSparse/University of Florida Sparse Matrix Collection [13]. We include all matrices from the `Boeing` and `GHS_psdef` groups which have between 100,000 and 2,250,000 non-zero entries and are strictly positive definite.

We provide nine candidate preconditioners for Algorithm 2 to select between, using block diagonal preconditioners in order to avoid existence issues of other common preconditioning methods [7]:

- The first candidate preconditioner is the trivial preconditioner $\boldsymbol{I}$, which is equivalent to using no preconditioner at all.

TABLE 1

*This table reports the number of iterations taken by the conjugate gradients algorithm to report an approximate solution $\tilde{x}$ to the linear system $Ax = b$ for specified test matrices $A$, a constant sampled standard normally distributed $b \sim \mathcal{N}(0, I)$, and various candidate preconditioners.*

| Matrix | Conjugate Gradients Iterations With Various Preconditioners | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $I$ | $D_1$ | $D_{10}$ | $D_{25}$ | $D_{50}$ | $D_{75}$ | $D_{100}$ | $R_{75}$ | $R_{100}$ |
| apache1 | 3,538 | 3,513 | 3,286 | 3,283 | 3,270 | 3,265 | 3,269 | 3,710 | 3,693 |
| crystm01 | 122 | 54 | 39 | 34 | 30 | 27 | 27 | 24 | 21 |
| crystm02 | 138 | 54 | 38 | 35 | 34 | 29 | 30 | 24 | 24 |
| crystm03 | 143 | 54 | 38 | 34 | 33 | 30 | 29 | 25 | 24 |
| cvxbqp1 | 16,424 | 11,337 | 11,338 | 11,332 | 11,331 | 11,330 | 11,328 | 10,148 | 10,353 |
| gridgena | 3,658 | 3,542 | 2,659 | 2,572 | 2,504 | 2,504 | 2,479 | 2,892 | 2,863 |
| jnlbrng1 | 139 | 131 | 126 | 126 | 125 | 125 | 125 | 130 | 130 |
| minsurfo | 94 | 88 | 64 | 63 | 63 | 62 | 62 | 88 | 88 |
| msc10848 | — | 5,659 | 3,791 | 3,028 | 2,793 | 2,656 | 2,628 | 2,192 | 2,092 |
| obstclae | 66 | 65 | 49 | 48 | 47 | 47 | 47 | 65 | 65 |
| oilpan | 48,291 | 28,065 | 12,804 | 8,167 | 5,476 | 4,992 | 4,127 | 4,757 | 4,433 |
| torsion1 | 66 | 65 | 49 | 48 | 47 | 47 | 47 | 65 | 65 |
| wathen100 | 327 | 45 | 44 | 44 | 44 | 44 | 44 | 42 | 42 |
| wathen120 | 378 | 45 | 45 | 45 | 44 | 44 | 44 | 43 | 43 |

- The preconditioner $D_\ell$ denotes a block-diagonal pinching/truncation of the matrix $A$ with block size $\ell$.
- The preconditioner $R_\ell$ is the same block-diagonal pinching, but performed after a Reverse Cuthill-McKee ordering of the matrix [12].

To ensure uniqueness and clarity, blocking is performed by taking the matrix $A \in \mathbb{F}^{d \times d}$ and constructing a block diagonal matrix $M$ with blocks of the form $A(m\ell : \min\{d, (m+1)\ell\}, m\ell : \min\{d, (m+1)\ell\})$ for $m = 0, 1, 2, \ldots$. Since $A$ is positive definite, the resulting preconditioners $M$ are also positive definite [9, Ex. 2.2.1.(viii)].

In Table 1 we present the number of iterations the preconditioned conjugate gradients algorithm took for each test matrix and preconditioner pair. The algorithm was run until the approximate solution $\tilde{x}$ satisfied $\|A\tilde{x} - b\|_2 \leq 10^{-9}\|b\|_2$. The number of iterations was limited to 50,000. Entries in Table 1 achieving this iteration limit are overwritten with '—'. The conjugate gradients algorithm applied to the matrices bcsstk36, bcsstk38, msc23052, and vanbody did not converge with any candidate preconditioner, so they are omitted in Table 1.

Even though larger block sizes $\ell$ ought to create better approximations of the original matrix, there are situations when smaller block sizes result in fewer conjugate gradients iterations. Similarly, there are some situations when the original ordering of the data is preferable over the Reverse Cuthill-McKee ordering, and vice-versa. As a result, it is unclear a-priori which preconditioner one should choose to solve the linear system, and this is why someone might wish to use Algorithm 2 to automate that choice.

We test this use of Algorithm 2 under two parameter settings $k = 10$ and $k = 50$. Algorithm 2 is run for 1,000 independent trials for each matrix-preconditioner-$k$ pairing. After the fact, we compare the number of iterations of preconditioned conjugate gradients would be necessary when using the recommendation of Algorithm

*This table summarizes the performance of Algorithm 2 for each matrix in Table 1, reporting statistics of the empirical number of iterations given by the algorithm compared to picking the worst-possible preconditioner (in terms of number of CG iterations) or choosing arbitrarily at random. Since the conjugate gradients algorithm did not converge for the matrix* `msc10848` *with no preconditioner, the 'Worst-Case' and 'Random' columns are* lower *bounds for their true values in that row only.*

| Matrix | Worst-Case | Random | Algorithm 2 Approximation Ratio | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | | | $k = 10$ | | | $k = 50$ | | |
| | | | Min | Mean | Max | Min | Mean | Max |
| apache1 | 1.14 | 1.05 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| crystm01 | 5.81 | 2.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| crystm02 | 5.75 | 1.88 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| crystm03 | 5.96 | 1.90 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| cvxbqp1 | 1.62 | 1.15 | 1.12 | 1.12 | 1.12 | 1.12 | 1.12 | 1.12 |
| gridgena | 1.48 | 1.15 | 1.00 | 1.00 | 1.01 | 1.00 | 1.00 | 1.00 |
| jnlbrng1 | 1.11 | 1.03 | 1.04 | 1.04 | 1.04 | 1.04 | 1.04 | 1.04 |
| minsurfo | 1.52 | 1.20 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| msc10848 | 23.90 | 3.97 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| obstclae | 1.40 | 1.18 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| oilpan | 11.70 | 3.26 | 1.00 | 1.09 | 1.15 | 1.07 | 1.08 | 1.15 |
| torsion1 | 1.40 | 1.18 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| wathen100 | 7.79 | 1.79 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| wathen120 | 8.79 | 1.89 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |

2 relative to the minimal number of iterations possible if we knew in advance how many iterations each preconditioner would use.

**3.1.1. Results.** The results of our generic real-world-use experiment are presented in Table 2. Every cell is an approximation ratio, i.e. the number of iterations an algorithm for selecting preconditioners took divided by the minimal number of iterations possible using our set of candidate preconditioners. As such, an entry of 1.00 is optimal and represents the minimal-number-of-iterations preconditioner being correctly selected. The column 'Worst-Case' reports the approximation ratio if one deterministically selected the maximal-number-of-iterations preconditioner in each setting. The column 'Random' reports the expected approximation ratio if one were to select a candidate preconditioner from Table 1 uniformly at random. The columns corresponding to Algorithm 2 gives statistics of the empirical distribution of approximation ratios seen over the 1,000 independent trial runs of the method.

For 10 of the 14 test matrices reported, setting $k = 10$ always picks the optimal preconditioner for the problem across every one of the 1,000 trials. If we take $k = 50$, this happens for 11 of the 14 test matrices. Moreover, even when the accuracy parameter $k = 10$, the returned preconditioner never needs more than 15% iterations over than the optimal choice.

One might wonder if taking $k$ to be even larger would result in approximation ratios concentrating more uniformly at the ideal 1.00 mark. This will not happen in general, and is where the good-proxy hypothesis is put to the test. For the `oilpan`

matrix, increasing $k$ from 10 to 50 *raises* the best-seen approximation ratio given by Algorithm 2 from 1.00 to 1.07. Increasing $k$ causes the preconditioner returned by Algorithm 2 to concentrate further around the true minimal-stability preconditioner (see Theorem 2.3,) and so this implies that the preconditioner stability criterion itself is not perfect and will not in general forecast the exact preconditioner resulting in the minimal number of conjugate gradients iterations.

**3.2. Experiments with Kernel Regression Preconditioners.** This section will show that Algorithm 2 can turn two simple preconditioners for the standard kernel regression problem into a robust, state-of-the-art preconditioning method. As a corollary of this investigation, we exhibit how Algorithm 2 performs well in situations when the 'minimal accuracy' criterion for selecting preconditioners fails, something left unanswered in the previous experiment.

**3.2.1. A Quick Review.** Kernel regression is a common statistical technique for nonlinear regression. In this setting, we have a dataset $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_d, y_d)\}$ consisting of $\boldsymbol{x}_i \mapsto y_i$ mappings from Euclidean space $\mathbb{R}^d$ to the real line $\mathbb{R}$. We wish to find coefficients $\boldsymbol{\alpha} \in \mathbb{R}^d$ so that the functional mapping

$$(3.1) \qquad \boldsymbol{x} \mapsto f(\boldsymbol{x}) = \sum_{i=1}^{d} \boldsymbol{\alpha}_i k(\boldsymbol{x}, \boldsymbol{x}_i)$$

faithfully represents the empirical mapping in the sense that $f(\boldsymbol{x}_i) \approx y_i$. In general, $k(\boldsymbol{x}, \boldsymbol{y})$ is just required to be a positive definite kernel, but in our experiment, we will only use the squared exponential kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\frac{\|\boldsymbol{x}-\boldsymbol{y}\|_2^2}{2\ell^2})$, parametrized by the length-scale $\ell$ which controls the derivative of the model $f(\boldsymbol{x})$. The coefficients $\boldsymbol{\alpha}$ are found by solving the system

$$(3.2) \qquad \boldsymbol{\alpha} = (\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{y}$$

where the positive definite Gram matrix $\boldsymbol{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, the output vector $\boldsymbol{y} = (y_1, y_2, \ldots, y_d)$, and the noise standarad deviation $\sigma_n > 0$ is used for regularization so that the model $f(\boldsymbol{x})$ fits well on out-of-sample data. In almost all kernel regression problems, $\boldsymbol{K}$ and hence $\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}$ are dense. See [32] for more background on this model and associated inference procedure.

**3.2.2. Related Work.** This experiment will test a preconditioning procedure for solving the linear system $(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I})\boldsymbol{\alpha} = \boldsymbol{y}$ via conjugate gradients. There has been a recent interest in this general iterative framework for kernel regression, which largely focuses on creating quality preconditioners for the Gram matrices $\boldsymbol{K}$ [3, 11, 18, 27, 26]. This is necessary since for reasonable parameter settings even $\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}$ is often ill-conditioned.

Cutajar et al. [11] performed some initial leg-work in this area, proposing eight candidate preconditioners. These preconditioners include a block-diagonal approximation of $\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}$, adding a larger regularizer $\sigma_n^2$ and solving recursively, a Nyström approximation of the Gram matrix using $\sqrt{n}$ data points as inducing points chosen uniformly at random, a coupling of the Nyström approximation with a block-diagonal approximation, or replacing $\boldsymbol{K}$ with an optimal low-rank factorization which can be computed via a randomized SVD [18] or the Lanczos method [16, Sec. 10.1]. Both [11] and the work [3] of Avron et al. use the Fourier features method of Rahimi and Recht [24] to create a preconditioner which replaces $\boldsymbol{K}$ with a sketched version $\tilde{\boldsymbol{K}}$.

The latter paper [3] also proposes using the TENSORSKETCH method of [23] for creating a sketched preconditioner when using the polynomial kernel $k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^* \boldsymbol{y})^q$, though the necessary sketching dimension in their upper bounds is exponential in $q$. The work of Rudi et al. [27] also uses the Nyström-based preconditioner like [11], combining it with other computational tricks. This Nyström-based approach was refined with approximate leverage score sampling in more recent work by Rudi et al. [26].

An empirical issue within many of the above methods is illustrated perfectly in Figure 1 of [11]. For every preconditioner presented therein, there exist parameter settings for which using no preconditioner results in fewer iterations than using the preconditioner when solving for $\boldsymbol{\alpha}$ via conjugate gradients. As such, it is unclear how one would choose a preconditioner in practice which makes these solvers work well in practice.

**3.2.3. Two Simple Geometrically Driven Preconditioners.** Here we detail the two candidate preconditioners which we will use in our experiments. They both utilize a geometrically-motivated reordering of the data to empirically achieve superior performance to the preconditioners of [11] for many settings of the kernel parameters $\sigma_n$ and $\ell$.

The first preconditioner is a simple block diagonal pinching of a reordering of the data. The kernel regression model under the squared exponential kernel effectively asserts that points nearby in $\ell^2$ ought to have similar outputs $y$. If the input data is highly clustered in $\ell^2$, our model then ought to largely ignore points from different clusters when considering a point in some cluster. The first preconditioning algorithm turns this 'ought to' statement directly into an approximation of the Gram matrix $\boldsymbol{K}$. We first cluster the data $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_d\}$ in $\ell^2$ via the k-means or k-means++ [1] algorithm with $c = \lceil \sqrt{d} \rceil$ clusters, constructing a permutation matrix $\boldsymbol{P}$ that places points in the same cluster next to each other on the Gram matrix. At this point, we precondition the re-ordered system $(\boldsymbol{PKP}^* + \sigma_n^2 \boldsymbol{I})\boldsymbol{P\alpha} = \boldsymbol{Py}$ by creating a block-diagonal pinching of the re-ordered matrix $\boldsymbol{PKP}^*$ where each block corresponds to the points within a cluster. The resulting preconditioner is that pinching $\hat{\boldsymbol{K}}$ plus the true noise term $\sigma_n^2 \boldsymbol{I}$.

The second preconditioner is a slightly more complex version of the first. After computing the permuted matrix $\boldsymbol{PKP}^*$, we compute a truncated rank-$r$ approximation $\boldsymbol{U\Lambda U}^*$ of $\boldsymbol{PKP}^*$ where $\Lambda \in \mathbb{R}^{r \times r}$ is diagonal and $\boldsymbol{U} \in \mathbb{R}^{d \times r}$ has orthonormal columns. At this point we compute the same block diagonal pinching $\tilde{\boldsymbol{E}}$ of the error in approximation $\boldsymbol{E} = \boldsymbol{PKP}^* - \boldsymbol{U\Lambda U}^*$. The resulting preconditioner is then $\boldsymbol{U\Lambda U}^* + \tilde{\boldsymbol{E}} + \sigma_n^2 \boldsymbol{I}$.

To show that these are feasible to use as preconditioners, it suffices to consider the latter since it reduces to the former when $r = 0$. If $r$ is a constant, we can solve systems in this preconditioner using the Woodbury identity [16, Sec. 2.1.4] in $\mathcal{O}(n^2)$ floating point operations under the assumption that the cluster sizes are all within a constant factor of each other. This is after computing a one-time Cholesky decomposition of the block-diagonal pinching in $\mathcal{O}(n^2)$ floating point operations. Similarly, computing the low-rank factorization takes $\mathcal{O}(n^2)$ floating point operations using either the Implicitly-Restarted Lanczos method [28] or a Randomized SVD [18], though for higher ranks $r$ the latter method is preferable. Since matrix-vector multiplies with $\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}$ take $\mathcal{O}(n^2)$ floating point operations, these preconditioners do not raise the per-iteration complexity of the conjugate gradients method.

Moreover, if we fix the resulting sparsity pattern of the preconditioner, the for-

mer preconditioner exactly minimizes the accuracy $\|\boldsymbol{M} - \boldsymbol{PKP}^* + \sigma_n^2\boldsymbol{I}\|_{\mathsf{F}}$ over all matrices with the same sparsity pattern. Since the identity matrix $\boldsymbol{I}$ also has this sparsity pattern, we would always choose the preconditioner over the identity matrix if using the accuracy criterion, and a similar logic shows the same holds for the latter preconditioner.

The work of Cutajar et al. [11] already suggests that block-diagonal approximations to kernel matrices can be faithful for small length-scales $\ell$, and that low-rank approximations can be faithful for large length-scales $\ell$. Following this helpful research, the main insight in our proposed preconditioners is that permutating the data in a way that points close together in Euclidean space are close on the matrix ensures a block-diagonal approximation is even more faithful. The permutation we use relies on the Euclidean geometry of our kernel, and other permutations would need to be used for kernels other than the squared exponential kernel.
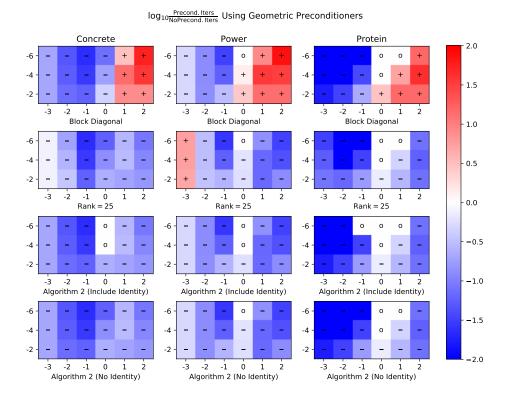
**3.2.4. Experimental Design.** We consider three datasets along with various parameter settings, aiming to see how many iterations the conjugate gradients algorithm takes with each of our proposed preconditioners, as well as using Algorithm 2, in comparison to using no preconditioner at all. This experiment is identical to one from [11] except with different preconditioners, and ideally our preconditioned solvers significantly reduce the number of iterations used by the standard conjugate gradients algorithm.

The datasets have names `Concrete`, `Power`, and `Protein`, and are identically the same as the data in [11] pulled from the UCI Machine Learning Repository [2]. The `Concrete` dataset consists of $d = 1{,}029$ data points in $\mathbb{R}^8$. The `Power` dataset consists of $d = 9{,}567$ data points in $\mathbb{R}^4$. The `Protein` dataset consists of $d = 45{,}729$ data points in $\mathbb{R}^9$.

For each of these datasets, and each pair of parameters chosen from $\ell \in \{10^{-3}, 10^{-2}, \ldots, 10^2\}$ and $\sigma_n^2 \in \{10^{-2}, 10^{-4}, 10^{-6}\}$, we construct a kernel system $(\boldsymbol{K} + \sigma_n^2\boldsymbol{I})\boldsymbol{\alpha} = \boldsymbol{y}$. This system is solved using conjugate gradients with no preconditioner, the geometric preconditioner with no low-rank approximation, and the geometric preconditioner with a rank $r = 25$ low-rank approximation. We also solve the system using the preconditioner chosen by one run of Algorithm 2 among no preconditioner, the purely block-diagonal geometric preconditioner, and the rank $r = 25$ low-rank approximation-based geometric preconditioner, using an accuracy parameter $k = 10$. We also attempt using Algorithm 2 with the same $k = 10$ if we restrict the choice to the two geometric preconditioners, ruling out the use of no preconditioner. In solving these systems, we record the number of conjugate gradients iterations needed to achieve a residual norm less than $10^{-5}\sqrt{d}$ as in [11]; a relative tolerance of $10^{-15}\|\boldsymbol{y}\|_2$ is also specified, though this is vacuous in comparison to the absolute tolerance. The solver is stopped after 10,000 iterations if the residual has not converged to within tolerance by then. The low-rank approximations are computed via `ARPACK` [21] with a tolerance parameter of $10^{-5}$.

**3.2.5. Results.** Figure 1 illustrates the relative improvement different preconditioning schemes have over using no preconditioner for each dataset and parameter combination. Each cell gives the logarithm of the ratio of the preconditioned conjugate gradients iterations to the non-preconditioned conjugate gradients iterations, i.e. the order of magnitude of the improvement granted by using the preconditioner. Accordingly, negative values (blue or '–') represent improvement through using the preconditioner, while positive values (red or '+') correspond to the preconditioned system requiring *more* iterations than using no preconditioner at all. Five of the

FIG. 1. *This figure presents the relative improvement of using our proposed preconditioners, or the one automatically chosen by Algorithm 2, with respect to using no preconditioner at all. Each individual matrix corresponds to a specific preconditioner and dataset pair. Each row gives the value of $\log \sigma_n^2$ used in the experiment, whereas each column corresponds to $\log \ell$. The absence of red cells in the result matrices corresponding to 'Our Method' indicates significant improvement over the results in [11].*



cells for the `Protein` dataset with the purely block diagonal geometric preconditioner have relative improvements of more than two orders of magnitude. Another three preconditioners using a low-rank approximation with the `Protein` dataset have this property. In spite of this, we restrict the visual range of the plot from −2 to 2 to allow Figure 1 to be compared easily to the identical presentation in Figure 1 of [11]. No cell values exceed 2.

First we comment purely on the performance of the two geometrically motivated preconditioners. The main take-away is that the geometric permutation based on the `k-means` algorithm appears to truly help in creating a faithful preconditioner. As evidence, we can point to the fact that the simple geometric block-diagonal preconditioner gives, for five different parameter settings with the `Protein` dataset, a relative improvement better than every single preconditioner-parameters-dataset pair in [11]. Phrased differently, at these parameter settings the number of iterations drops from 189, 111, 2,345, 618, and 10,000 (did not converge) to 1, 1, 3, 3, and 94 iterations, respectively. Moreover, the geometric preconditioner using a low-rank approximation for the `Concrete` dataset always outperforms using no preconditioner, something no preconditioner proposed in [11] can do. These improvements are genuine and stark, and again achieved by an extremely simple method just by relying on geometry.

Of course, one can rightfully point out that the block diagonal pinching is not robust as a preconditioner, just like many methods from [11]. This is true; the block diagonal approximation works well for small length scales $\ell$, as in these circumstances dependencies $\boldsymbol{K}_{ij}$ between far away data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are shrunk, resulting in a genuine clustering of the underlying data where the intuition we used in justifying the preconditioner carries through. For large $\ell$, the block diagonal preconditioner performs poorly because the matrix $\boldsymbol{K}$ looks more uniform and doesn't have a genuine clustered structure. Luckily, the more sophisticated preconditioners with added rank-25 terms perform well in precisely this regime, as the low-rank term can capture uniform structure in the Gram matrix $\boldsymbol{K}$. While this complicated preconditioner is not perfect, it is more robust to parameter changes than the analogous SVD-based preconditioner from [11]. Between our two candidate preconditioners, at least one provides a performance boost over non-preconditioned conjugate gradients for every dataset and parameter setting chosen. Such a claim cannot be said about any pair of preconditioners in [11].

Since we have two quality preconditioners, each performing admirably in opposing parameter regimes, we might hope to get the best of both worlds by forecasting via Algorithm 2 which one will perform better than using no preconditioner and solving the system with that resulting preconditioner. This approach does quite well, as we can see in Figure 1. While Algorithm 2 does not always pick the best preconditioner in terms of minimizing the number of conjugate gradients iterations, it never selects a preconditioner which performs worse than using no preconditioner. That said, a preconditioner resulting in an exactly minimal number of iterations is chosen over 80% of the time if the 'use no preconditioner' option is included, and over 40% of the time the preconditioner ranking induced by our stability estimates exactly corresponds to the ranking induced by the true iteration count. If we exclude the 'use no preconditioner' option, which corresponds to an a-priori understanding that at least one of the geometric preconditioners works well, the former statistic jumps from 80% to an impressive 98.1%. This 'all blue' plot which represents a robust preconditioner regime can not be found using the techniques of [11]. Moreover, the algorithm was able to return the advice 'use no preconditioner' in the face of uncertainty instead of suggesting the use of a poor preconditioner. This fact alone is highly desirable for the practitioner.

To confirm the importance of this paper, it is necessary to show that our method performs well when the computationally simple accuracy method does not. As mentioned when detailing the construction of these preconditioners, the accuracy criterion would never choose the 'use no preconditioner' option over one of the geometric preconditioner. If we were just looking at the purely block-diagonal geometric preconditioner versus the 'use no preconditioner option', the accuracy criterion would result in a poor preconditioner (higher number of iterations than possible) exactly a third of the time with the `Concrete` dataset. Of these times that the accuracy method fails, the estimated stability criterion succeeds exactly half of the time. For the `Power` dataset, the accuracy method fails 44.4% of the time, but our estimated stability criterion succeeds in a quarter of these cases. While this behavior is not universal, it indicates that our method can be a crucial help when standard tools fail.

Finally, it is important to point out that in this setting, Algorithm 2 performed computation commensurate with taking 30 steps of conjugate gradients. Since in over half of the parameter-dataset pairs the non-preconditioned conjugate gradients algorithm took more than five times this number of iterations, and our method can in most situations reduce that full-solution cost significantly, this initial cost is acceptable.

**4. Conclusions.** We have created a method to make the conjugate gradients algorithm friendlier to the practitioner. In particular, we took a quality forecast of preconditioner quality previously thought of as unusable in this arena, preconditioner stability, and presented a randomized algorithm which can quickly compute this quantity and use it to select a quality preconditioner. Our methods are motivated and justified heavily by theory and backed up by empirical evidence which suggests its applicability in the real world. In particular, Algorithm 2 allowed us to create the first practical preconditioning system for kernel regression which is reported to never run fewer iterations than using no preconditioner at all in standard experiments; this is accomplished with surprisingly little leg-work.

Our work raises some important theoretical questions which would be ripe for future work. Most notably, it would be helpful to determine the fundamental limits of using preconditioner stability as a proxy for preconditioner quality with regard to different iterative methods. For example, it would be interesting to know if there are arbitrarily long sequences of preconditioners for some fixed positive definite matrix for which the order induced by the preconditioner-stability criterion is the opposite of the order induced by the condition number.

Finally, we note that our framing in Section 2.2 allows us to interpret the problem of finding a good preconditioner as a multi-armed bandit problem. Algorithms and lower bounds from this research area – see [20] for example – would then be informative to our problem of interest. As before, this investigation is left for future work.

## REFERENCES

[1] D. ARTHUR AND S. VASSILVITSKII, *k-means++: The advantages of careful seeding*, in Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[2] A. ASUNCION AND D. NEWMAN, *Uci machine learning repository*, 2007.

[3] H. AVRON, K. L. CLARKSON, AND D. P. WOODRUFF, *Faster kernel ridge regression using sketching and preconditioning*, SIAM Journal on Matrix Analysis and Applications, 38 (2017), pp. 1116–1138.

[4] H. AVRON, A. DRUINSKY, AND S. TOLEDO, *Spectral condition-number estimation of large sparse matrices*, Numerical Linear Algebra with Applications, 26 (2019), p. e2235.

[5] H. AVRON AND S. TOLEDO, *Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix*, Journal of the ACM (JACM), 58 (2011), p. 8.

[6] O. AXELSSON AND V. EIJKHOUT, *Vectorizable preconditioners for elliptic difference equations in three space dimensions*, in Advances in Parallel Computing, vol. 1, Elsevier, 1990, pp. 299–321.

[7] M. BENZI, *Preconditioning techniques for large linear systems: a survey*, Journal of computational Physics, 182 (2002), pp. 418–477.

[8] M. BENZI, D. B. SZYLD, AND A. VAN DUIN, *Orderings for incomplete factorization preconditioning of nonsymmetric problems*, SIAM Journal on Scientific Computing, 20 (1999), pp. 1652–1670.

[9] R. BHATIA, *Positive definite matrices*, vol. 16, Princeton university press, 2009.

[10] E. CHOW AND Y. SAAD, *Experimental study of ilu preconditioners for indefinite matrices*, Journal of Computational and Applied Mathematics, 86 (1997), pp. 387–414.

[11] K. CUTAJAR, M. OSBORNE, J. CUNNINGHAM, AND M. FILIPPONE, *Preconditioning kernel matrices*, in International Conference on Machine Learning, 2016, pp. 2529–2538.

[12] E. CUTHILL AND J. MCKEE, *Reducing the bandwidth of sparse symmetric matrices*, in Proceedings of the 1969 24th national conference, ACM, 1969, pp. 157–172.

[13] T. A. DAVIS AND Y. HU, *The University of Florida sparse matrix collection*, ACM Transactions on Mathematical Software (TOMS), 38 (2011), p. 1.

[14] P. DEIFT AND T. TROGDON, *The conjugate gradient algorithm on well-conditioned wishart matrices is almost deteriministic*, arXiv preprint arXiv:1901.09007, (2019).

[15] I. S. DUFF AND G. A. MEURANT, *The effect of ordering on preconditioned conjugate gradients*, BIT Numerical Mathematics, 29 (1989), pp. 635–657.

[16] G. H. GOLUB AND C. F. VAN LOAN, *Matrix computations*, vol. 3, JHU Press, 2012.

[17] R. D. GORDON, *Values of mills' ratio of area to bounding ordinate and of the normal probability integral for large values of the argument*, The Annals of Mathematical Statistics, 12 (1941), pp. 364–366.

[18] N. HALKO, P.-G. MARTINSSON, AND J. A. TROPP, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM review, 53 (2011), pp. 217–288.

[19] A. HOORFAR AND M. HASSANI, *Approximation of the Lambert-W function and hyperpower function*, Research report collection, 10 (2007).

[20] K. JAMIESON, M. MALLOY, R. NOWAK, AND S. BUBECK, *On finding the largest mean among many*, arXiv preprint arXiv:1306.3917, (2013).

[21] R. B. LEHOUCQ, D. C. SORENSEN, AND C. YANG, *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*, vol. 6, Siam, 1998.

[22] Y. LI, H. L. NGUYEN, AND D. P. WOODRUFF, *On sketching matrix norms and the top singular vector*, in Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2014, pp. 1562–1581.

[23] R. PAGH, *Compressed matrix multiplication*, ACM Transactions on Computation Theory (TOCT), 5 (2013), p. 9.

[24] A. RAHIMI AND B. RECHT, *Random features for large-scale kernel machines*, in Advances in neural information processing systems, 2008, pp. 1177–1184.

[25] F. ROOSTA-KHORASANI AND U. ASCHER, *Improved bounds on sample size for implicit matrix trace estimators*, Foundations of Computational Mathematics, 15 (2015), pp. 1187–1212.

[26] A. RUDI, D. CALANDRIELLO, L. CARRATINO, AND L. ROSASCO, *On fast leverage score sampling and optimal learning*, in Advances in Neural Information Processing Systems, 2018, pp. 5672–5682.

[27] A. RUDI, L. CARRATINO, AND L. ROSASCO, *Falkon: An optimal large scale kernel method*, in Advances in Neural Information Processing Systems, 2017, pp. 3888–3898.

[28] D. C. SORENSEN, *Implicitly restarted arnoldi/lanczos methods for large scale eigenvalue calculations*, in Parallel Numerical Algorithms, Springer, 1997, pp. 119–165.

[29] L. N. TREFETHEN AND D. BAU III, *Numerical linear algebra*, vol. 50, Siam, 1997.

[30] E. E. TYRTYSHNIKOV, *Optimal and superoptimal circulant preconditioners*, SIAM Journal on Matrix Analysis and Applications, 13 (1992), pp. 459–473.

[31] R. VERSHYNIN, *High-dimensional probability: An introduction with applications in data science*, vol. 47, Cambridge University Press, 2018.

[32] C. K. WILLIAMS AND C. E. RASMUSSEN, *Gaussian processes for machine learning*, vol. 2, MIT Press Cambridge, MA, 2006.

[33] K. WIMMER, Y. WU, AND P. ZHANG, *Optimal query complexity for estimating the trace of a matrix*, in International Colloquium on Automata, Languages, and Programming, Springer, 2014, pp. 1051–1062.

[34] D. P. WOODRUFF, *Sketching as a tool for numerical linear algebra*, Foundations and Trends® in Theoretical Computer Science, 10 (2014), pp. 1–157.