SIAM J. MATRIX ANAL. APPL.
Vol. 44, No. 2, pp. 718–752

RANDOMIZED NYSTRÖM PRECONDITIONING*

ZACHARY FRANGELLA†, JOEL A. TROPP‡, AND MADELEINE UDELL†

Abstract. This paper introduces the Nyström preconditioned conjugate gradient (PCG) algorithm for solving a symmetric positive-definite linear system. The algorithm applies the randomized Nyström method to form a low-rank approximation of the matrix, which leads to an efficient preconditioner that can be deployed with the conjugate gradient algorithm. Theoretical analysis shows that the preconditioned system has constant condition number as soon as the rank of the approximation is comparable with the number of effective degrees of freedom in the matrix. The paper also develops adaptive methods that provably achieve similar performance without knowledge of the effective dimension. Numerical tests show that Nyström PCG can rapidly solve large linear systems that arise in data analysis problems, and it surpasses several competing methods from the literature.

Key words. conjugate gradient, cross-validation, kernel method, linear system, Nyström approximation, preconditioner, randomized algorithm, regularized least-squares, ridge regression

MSC codes. 65F08, 68W20, 65F55, 65F22

DOI. 10.1137/21M1466244

1. Motivation.

In their elegant 1997 textbook on numerical linear algebra [44], Trefethen and Bau write,

> In ending this book with the subject of preconditioners, we find ourselves at the philosophical center of the scientific computing of the future... 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 Krylov subspace matrix iterations, this is preconditioning...we can only guess where this idea will take us.

The next century has since arrived, and one of the most fruitful developments in matrix computations has been the emergence of new algorithms that use randomness in an essential way. This paper explores a topic at the nexus of preconditioning and randomized numerical linear algebra. We will show how to use a randomized matrix approximation algorithm to construct a preconditioner for an important class of linear systems that arises throughout data analysis and scientific computing.

1.1. The preconditioner.

Consider the regularized linear system

$$(A + \mu I)x = b \quad \text{where } A \in \mathbb{R}^{n\times n} \text{ is symmetric psd and } \mu \geq 0. \tag{1.1}$$

Here and elsewhere, psd abbreviates the term "positive semidefinite." This type of linear system emerges whenever we solve a regularized least-squares problem. We will design a class of preconditioners for the problem (1.1).

*Received by the editors December 20, 2021; accepted for publication (in revised form) by J. Chung October 28, 2022; published electronically May 30, 2023.
https://doi.org/10.1137/21M1466244
Funding: The work of the first and third authors was supported by the National Science Foundation award IIS-1943131, the ONR Young Investigator Program, and the Alfred P. Sloan Foundation. The work of the second author was supported the ONR BRC award N00014-18-1-2363 and the National Science Foundation FRG award 1952777.

†Department of Management Science and Engineering, Stanford University, Stanford, CA 94305 USA (zfran@stanford.edu, udell@stanford.edu).

‡Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA 91125-5000 USA (jtropp@cms.edu).

Throughout this paper, we assume that we can access the matrix A through matrix-vector products $x \mapsto Ax$, commonly known as *matvecs*. The algorithms that we develop will economize on the number of matvecs, and they may not be appropriate in settings where matvecs are very expensive or there are cheaper ways to interact with the matrix.

For a rank parameter $\ell \in \mathbb{N}$, the randomized Nyström approximation of A takes the form

$$\hat{A}_{\mathrm{nys}} = (A\Omega)(\Omega^T A\Omega)^{\dagger}(A\Omega)^T \quad \text{where } \Omega \in \mathbb{R}^{n\times \ell} \text{ is standard normal.} \tag{1.2}$$

This matrix provides the best psd approximation of A whose range coincides with the range of the sketch $A\Omega$. The randomness in the construction ensures that $\hat{A}_{\mathrm{nys}}$ is a good approximation to the original matrix A with high probability [25, sec. 14].

We can form the Nyström approximation with sketch size ℓ, using ℓ matvecs with A, plus some extra arithmetic. See Algorithm 2.1 for the implementation details.

Given the eigenvalue decomposition $\hat{A}_{\mathrm{nys}} = U\hat{\Lambda}U^T$ of the randomized Nyström approximation, we construct the Nyström preconditioner:

$$P = \frac{1}{\hat{\lambda}_\ell + \mu} U(\hat{\Lambda} + \mu I)U^T + (I - UU^T). \tag{1.3}$$

In a slight abuse of terminology, we refer to ℓ as the rank of the Nyström preconditioner. The key point is that we can solve the linear system $Py = c$ very efficiently, and the action of P^{-1} dramatically reduces the condition number of the regularized matrix $A_\mu = A + \mu I$.

We propose using (1.3) in conjunction with the preconditioned conjugate gradient (PCG) algorithm. Each iteration of PCG involves a single matvec with A, and a single linear solve with P. When the preconditioned matrix $P^{-1}A_\mu$ has a modest condition number, the algorithm converges to a solution of (1.1) very quickly. See Algorithm 5.1 for pseudocode for Nyström PCG.

The idea of using the randomized Nyström approximation to construct the preconditioner in (1.3) was suggested by P.-G. Martinsson in the survey [25, sec. 17], but it has not been implemented or analyzed. An earlier (folklore) preconditioner with similar motivation uses a partial eigendecomposition to form a preconditioner of the form (1.3); for instance, in [14], this idea is called a "deflating preconditioner." However, as computing an exact partial eigendecomposition is prohibitively expensive for large problems, these deflating preconditioners are rarely used. Randomized numerical linear algebra, such as the randomized Nyström approximation used here, provides the key ingredient to make such a preconditioner practical.

1.2. Guarantees. This paper contains the first comprehensive study of the preconditioner (1.3), including theoretical analysis and testing on prototypical problems from data analysis and machine learning. One of the main contributions is a rigorous method for choosing the rank ℓ to guarantee good performance, along with an adaptive rank selection procedure that performs well in practice.

A key quantity in our analysis is the *effective dimension* of the regularized matrix $A + \mu I$. That is,

$$d_{\mathrm{eff}}(\mu) = \operatorname{tr}\left(A(A + \mu I)^{\dagger}\right) = \sum_{j=1}^{n} \frac{\lambda_j(A)}{\lambda_j(A) + \mu}, \tag{1.4}$$

where $(A + \mu I)^{\dagger}$ is the Moore–Penrose pseudoinverse. Our definition differs slightly from the literature [1, 4], which uses $(A + \mu I)^{-1}$; the definition we use allows for the

effective dimension to be defined even when $\mu = 0$, in which case it equals the rank of A. The effective dimension measures the degrees of freedom of the problem after regularization. It may be viewed as a (smoothed) count of the eigenvalues larger than μ. Many real-world matrices exhibit strong spectral decay, so for $\mu > 0$ the effective dimension is typically much smaller than the nominal dimension n. As we will discuss, the effective dimension also plays a role in a number of machine learning papers [1, 2, 4, 7, 22] that consider randomized algorithms for solving regularized linear systems.

Our theory tells us the randomized Nyström preconditioner P is successful when its rank ℓ is proportional to the effective dimension.

THEOREM 1.1 (randomized Nyström preconditioner). *Let $A \in \mathbb{S}_n^+(\mathbb{R})$ be a psd matrix, and write $A_\mu = A + \mu I$, where the regularization parameter $\mu > 0$. Define the effective dimension $d_{\text{eff}}(\mu)$ as in* (1.4)*. Construct the randomized preconditioner P from* (1.2) *and* (1.3) *with rank parameter $\ell = 2\lceil 1.5\, d_{\text{eff}}(\mu)\rceil + 1$. Then the condition number of the preconditioned system satisfies*

$$\mathbb{E}\big[\kappa_2(P^{-1/2} A_\mu P^{-1/2})\big] < 28. \tag{1.5}$$

Theorem 1.1 is a restatement of Theorem 5.1.

Simple probability bounds follow from (1.5) via Markov's inequality. For example,

$$\mathbb{P}\big\{\kappa_2(P^{-1/2} A_\mu P^{-1/2}) \le 56\big\} > 1/2.$$

The main consequence of Theorem 1.1 is a convergence theorem for PCG with the randomized Nyström preconditioner.

COROLLARY 1.2 (Nyström PCG: Convergence). *Construct the preconditioner P as in Theorem* 1.1*, and condition on the event $\{\kappa_2(P^{-1/2} A_\mu P^{-1/2}) \le 56\}$. Solve the regularized linear system* (1.1) *using Nyström PCG, starting with an initial iterate $x_0 = 0$. After t iterations, the relative error δ_t satisfies*

$$\delta_t := \frac{\|x_t - x_\star\|_{A_\mu}}{\|x_\star\|_{A_\mu}} < 2 \cdot (0.77)^t, \quad \text{where } A_\mu x_\star = b.$$

The error norm is defined as $\|u\|_{A_\mu}^2 = u^T A_\mu u$. In particular, $t \ge \lceil 3.9 \log(2/\epsilon)\rceil$ iterations suffice to achieve relative error ϵ.

Although Theorem 1.1 gives an interpretable bound for the rank ℓ of the preconditioner, we cannot instantiate it without knowledge of the effective dimension. To address this shortcoming, we have designed adaptive methods for selecting the rank in practice (subsection 5.4).

Finally, as part of our investigation, we will also develop a detailed understanding of Nyström sketch-and-solve, a popular algorithm in the machine learning literature [1, 4]. Our analysis highlights the deficiencies of Nyström sketch-and-solve relative to Nyström PCG.

1.3. Example: Ridge regression. As a concrete example, we consider the ℓ^2 regularized least-squares problem, also known as ridge regression. This problem takes the form

$$\text{minimize}_{x \in \mathbb{R}^d} \quad \frac{1}{2n}\|Gx - b\|^2 + \frac{\mu}{2}\|x\|^2, \tag{1.6}$$

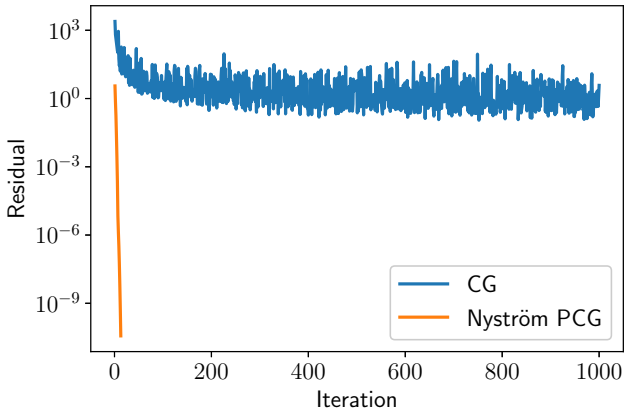


FIG. 1. Ridge regression: CG versus Nyström PCG. *For the shuttle-rf data set, Nyström PCG converges to machine precision in* 13 *iterations, while CG stalls. See subsections* 1.3 *and* 6.2.

where $G \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$ and $\mu > 0$. By calculus, the solution to (1.6) also satisfies the regularized system of linear equations

$$\left(\frac{1}{n}G^T G + \mu I\right) x = \frac{1}{n}G^T b. \tag{1.7}$$

A direct method to solve (1.7) requires $O(nd^2)$ flops, which is prohibitive when n and d are both large. Instead, when n and d are large, iterative algorithms, such as the conjugate gradient method (CG), become the tools of choice. Unfortunately, the ridge regression linear system (1.7) is often very ill-conditioned, and CG converges very slowly.

Nyström PCG can dramatically accelerate the solution of (1.7). As an example, consider the shuttle-rf dataset (subsection 6.2). The matrix G has dimension $43,300 \times 10,000$, while the preconditioner is based on a Nyström approximation with rank $\ell = 800$. Figure 1 shows the progress of the residual as a function of the iteration count. Nyström PCG converges to machine precision in 13 iterations, while CG stalls.

1.4. Comparison to prior randomized preconditioners. Prior proposals for randomized preconditioners [3, 28, 36] accelerate the solution of highly overdetermined or underdetermined least-squares problems using the sketch-and-precondition paradigm [25, sec. 10]. For $n \geq d$, these methods require $\Omega(d^3)$ computation to factor the preconditioner. In contrast, the randomized Nyström preconditioner applies to any symmetric positive-definite linear system and can be significantly faster for regularized problems. See subsection 5.2.2 for more details.

1.5. Roadmap. Section 2 contains an overview of the Nyström approximation and its key properties. Section 3 studies the role of the Nyström approximation in estimating the inverse of the regularized matrix. We analyze the Nyström sketch-and-solve method in Algorithm 4.1, and we give a rigorous performance bound for this algorithm. Section 5 presents a full treatment of Nyström PCG, including theoretical results and guidance on numerical implementation. Computational experiments in section 6 demonstrate the power of Nyström PCG for three different applications involving real data sets.

1.6. Notation. We write $\mathbb{S}_n(\mathbb{R})$ for the linear space of $n \times n$ real symmetric matrices, while $\mathbb{S}_n^+(\mathbb{R})$ denotes the convex cone of real psd matrices. The symbol $\preceq$ denotes the Loewner order on $\mathbb{S}_n(\mathbb{R})$. That is, $A \preceq B$ if and only if the eigenvalues

of $B - A$ are all nonnegative. The function $\mathrm{tr}[\cdot]$ returns the trace of a square matrix. The map $\lambda_j(A)$ returns the jth largest eigenvalue of A; we may omit the matrix if it is clear. As usual, κ_2 denotes the ℓ^2 condition number. We write $\|M\|$ for the spectral norm of a matrix M. For a psd matrix A, we write $\|u\|_A^2 = u^T A u$ for the A-norm. Given $A \in \mathbb{S}_n(\mathbb{R})$ and $1 \le \ell \le n$, the symbol $\lfloor A \rfloor_\ell$ refers to any best rank-ℓ approximation to A relative to the spectral norm. For $A \in \mathbb{S}_n^+(\mathbb{R})$ and $\mu \ge 0$, the regularized matrix is abbreviated $A_\mu = A + \mu I$. For $A \in \mathbb{S}_n^+(\mathbb{R})$ and $\mu \ge 0$ the effective dimension of A_μ is defined as $d_{\mathrm{eff}}(\mu) = \mathrm{tr}(A(A + \mu I)^\dagger)$. For $A \in \mathbb{S}_n^+(\mathbb{R})$, the p-stable rank of A is defined as $\mathrm{sr}_p(A) = \lambda_p^{-1} \sum_{j>p}^n \lambda_j$. For $A \in \mathbb{S}_n^+(\mathbb{R})$, we denote the time taken to compute a matvec with A by T_{mv}.

2. The Nyström approximation.

Let us begin with a review of the Nyström approximation and the randomized Nyström approximation.

2.1. Definition and basic properties.

The Nyström approximation is a natural way to construct a low-rank psd approximation of a psd matrix $A \in \mathbb{S}_n^+(\mathbb{R})$. Let $X \in \mathbb{R}^{n \times \ell}$ be an arbitrary test matrix. The *Nyström approximation* of A with respect to the range of X is defined by

$$A\langle X \rangle = (AX)(X^T A X)^\dagger (AX)^T \in \mathbb{S}_n^+(\mathbb{R}). \tag{2.1}$$

The Nyström approximation is the best psd approximation of A whose range coincides with the range of AX. It has a deep relationship with the Schur complement and with Cholesky factorization [25, sec. 14].

The Nyström approximation enjoys several elementary properties that we record in the following lemma.

LEMMA 2.1. *Let $A\langle X \rangle \in \mathbb{S}_n^+(\mathbb{R})$ be a Nyström approximation of the psd matrix $A \in \mathbb{S}_n^+(\mathbb{R})$. Then the following hold:*

1. *The approximation $A\langle X \rangle$ is psd and has rank at most ℓ.*
2. *The approximation $A\langle X \rangle$ depends only on* $\mathrm{range}(X)$*, that is*
$$\mathrm{range}(A\langle X \rangle) \subset \mathrm{range}(X).$$
3. *In the Loewner order, $A\langle X \rangle \preceq A$.*
4. *In particular, the eigenvalues satisfy $\lambda_j(\hat{A}) \le \lambda_j(A)$ for each $1 \le j \le n$.*

The proof of Lemma 2.1, item 3 is not completely obvious. It is a consequence of the fact that we may express $\hat{A}_{\mathrm{nys}} = A^{1/2} \Pi A^{1/2}$, where Π is an orthogonal projector.

2.2. Randomized Nyström approximation.

How should we choose the test matrix X so that the Nyström approximation $A\langle X \rangle$ provides a good low-rank model for A? Surprisingly, we can obtain a good approximation simply by drawing the test matrix at random. See [45] for theoretical justification of this claim.

Let us outline the construction of the randomized Nyström approximation. Draw a standard normal test matrix $\Omega \in \mathbb{R}^{n \times \ell}$ where ℓ is the sketch size, and compute the sketch $Y = A\Omega$. By Lemma 2.1, the sketch size ℓ is equal to the rank of $\hat{A}_{\mathrm{nys}}$ with probability 1; hence we use these terms interchangeably. The Nyström approximation (2.1) is constructed directly from the test matrix Ω and the sketch Y:

$$\hat{A}_{\mathrm{nys}} = A\langle \Omega \rangle = Y(\Omega^T Y)^\dagger Y^T. \tag{2.2}$$

The formula (2.2) is not numerically sound. We refer the reader to Algorithm 2.1 for a stable and efficient implementation of the randomized Nyström approximation [23, 45, 30]. Conveniently, Algorithm 2.1 returns the truncated eigendecomposition $\hat{A}_{\mathrm{nys}} =$

Algorithm 2.1 Randomized Nyström Approximation [23, 45, 30]

Input: Psd matrix $A \in \mathbb{S}_n^+(\mathbb{R})$, rank ℓ
Output: Nyström approximation in factored form $\hat{A}_{\text{nys}} = U\hat{\Lambda}U^T$

1: $\Omega = \text{randn}(n, \ell)$ ▷ Gaussian test matrix
2: $\Omega = \text{qr}(\Omega, 0)$ ▷ Thin QR decomposition
3: $Y = A\Omega$ ▷ ℓ matvecs with A
4: $\nu = \text{eps}(\text{norm}(Y, \text{'fro'}))$ ▷ Compute shift
5: $Y_\nu = Y + \nu\Omega$ ▷ Shift for stability
6: $C = \text{chol}(\Omega^T Y_\nu)$
7: $B = Y_\nu / C$
8: $[U, \Sigma, \sim] = \text{svd}(B, 0)$ ▷ Thin SVD
9: $\hat{\Lambda} = \max\{0, \Sigma^2 - \nu I\}$ ▷ Remove shift, compute eigs

$U\hat{\Lambda}U^T$, where $U \in \mathbb{R}^{n\times\ell}$ is an orthonormal matrix whose columns are eigenvectors and $\hat{\Lambda} \in \mathbb{R}^{\ell\times\ell}$ is a diagonal matrix listing the eigenvalues, which we often abbreviate as $\hat{\lambda}_1, \ldots, \hat{\lambda}_\ell$.

The randomized Nyström approximation described in this section has a key difference from the Nyström approximations that have traditionally been used in the machine learning literature [1, 4, 9, 15, 48]. In machine learning settings, the Nyström approximation is usually constructed from a sketch Y that samples random columns from the matrix (i.e., the random test matrix Ω has 1-sparse columns). In contrast, Algorithm 2.1 computes a sketch Y via random projection (i.e., the test matrix Ω is standard normal). In most applications, we have strong reasons (subsection 2.2.3) for preferring random projections to column sampling.

2.2.1. Cost of randomized Nyström approximation. Throughout the paper, we write T_{mv} for the time required to compute a matrix-vector product (matvec) with A. Forming the sketch $Y = A\Omega$ with sketch size ℓ requires ℓ matvecs, which costs $T_{\text{mv}}\ell$. The other steps in the algorithm have arithmetic cost $O(n\ell^2)$. Hence, the total computational cost of Algorithm 2.1 is $O(T_{\text{mv}}\ell + \ell^2 n)$ operations. The storage cost is $O(\ell n)$ floating-point numbers.

For Algorithm 2.1, the worst-case performance occurs when A is dense and unstructured. In this case, forming Y costs $O(n^2\ell)$ operations. However, if we have access to the columns of A, then we may reduce the cost of forming Y to $O(n^2 \log \ell)$ by using a structured test matrix Ω, such as a scrambled subsampled randomized Fourier transform (SSRFT) map or a sparse map [25, 45].

2.2.2. A priori guarantees for the randomized Nyström approximation. In this section, we present an a priori error bound for the randomized Nyström approximation. The result improves over previous analyses [15, 16, 45] by sharpening the error terms. This refinement is critical for the analysis of the preconditioner.

PROPOSITION 2.2 (randomized Nyström approximation: Error). *Consider a psd matrix $A \in \mathbb{S}_n^+(\mathbb{R})$ with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$. Choose a sketch size $\ell \geq 4$, and draw a standard normal test matrix $\Omega \in \mathbb{R}^{n\times\ell}$. Then the rank-$\ell$ Nyström approximation $\hat{A}_{\text{nys}}$ computed by Algorithm 2.1 satisfies*

$$\mathbb{E}\|A - \hat{A}_{\text{nys}}\| \leq \min_{2\leq p\leq \ell-2} \left[\left(1 + \frac{2(\ell-p)}{p-1}\right)\lambda_{\ell-p+1} + \frac{2\mathrm{e}^2\ell}{p^2-1}\left(\sum_{j>\ell-p}\lambda_j\right)\right]. \tag{2.3}$$

The proof of Proposition 2.2 may be found in subsection SM1.1.

Proposition 2.2 shows that, in expectation, the randomized Nyström has several appealing features. Many other approximations $\hat{A}_{\mathrm{nys}}$ provide a good rank-ℓ approximation to A. The first term in the bound is comparable with the spectral-norm error $\lambda_{\ell-p+1}$ in the optimal rank-$(\ell-p)$ approximation, $\lfloor A \rfloor_{\ell-p}$. The second term in the bound is comparable with the trace-norm error $\sum_{j>\ell-p} \lambda_j$ in the optimal rank-$(\ell-p)$ approximation.

Proposition 2.2 is better understood via the following simplification.

COROLLARY 2.3. *Instate the assumptions of Proposition* 2.2. *For* $p \geq 2$ *and* $\ell = 2p-1$, *we have the bound*

$$\mathbb{E}\|A - \hat{A}_{\mathrm{nys}}\| \leq \left(3 + \frac{4\mathrm{e}^2}{p}\mathrm{sr}_p(A)\right)\lambda_p.$$

The p*-stable rank,* $\mathrm{sr}_p(A) = \lambda_p^{-1}\sum_{j=p}^{n}\lambda_j$, *reflects decay in the tail eigenvalues.*

Corollary 2.3 shows that the Nyström approximation error is on the order of λ_p when the rank parameter $\ell = 2p-1$. The constant depends on the p-stable rank $\mathrm{sr}_p(A)$, which is small when the tail eigenvalues decay quickly starting at λ_p. This bound is critical for establishing our main results (Theorems 4.2 and 5.1).

2.2.3. Random projection versus column sampling. Most papers in the machine learning literature [1, 4] construct Nyström approximations by sampling columns at random from an adaptive distribution. In contrast, for most applications, we advocate using an oblivious random projection of the matrix to construct a Nyström approximation.

Random projection has several advantages over column sampling. First, column sampling offers no computational advantage when we only have black-box matvec access to the matrix, while random projections are natural in this setting and possess stronger performance guarantees. Second, it can be very expensive to obtain adaptive distributions for column sampling. Indeed, computing approximate ridge leverage scores costs just as much as solving the ridge regression problem directly using random projections [10, Theorem 2]. Third, even with a good sampling distribution, column sampling produces higher variance results than random projection, so it is far less reliable.

On the other hand, we have found that there are a few applications where it is more effective to compute a randomized Nyström preconditioner using column sampling in lieu of random projections. In particular, this seems to be the case for kernel ridge regression (subsection 6.5). Indeed, the entries of the kernel matrix are given by an explicit formula, so we can extract full columns with ease. Sampling ℓ columns may cost only $O(\ell n)$ operations, whereas a single matvec generally costs $O(n^2)$. Furthermore, kernel matrices usually exhibit fast spectral decay, which limits the performance loss that results from using column sampling in lieu of random projection.

3. Approximating the regularized inverse. Let us return to the regularized linear system (1.1). The solution to the problem has the form $x_\star = (A+\mu I)^{-1}b$. Given a good approximation $\hat{A}$ to the matrix A, it is natural to ask whether $\hat{x} = (\hat{A}+\mu I)^{-1}b$ is a good approximation to the desired solution $x_\star$.

There are many reasons why we might prefer to use $\hat{A}$ in place of A. In particular, we may be able to solve linear systems in the matrix $\hat{A}+\mu I$ more efficiently. On the other hand, the utility of this approach depends on how well the inverse $(\hat{A}+\mu I)^{-1}$

approximates the desired inverse $(A+\mu I)^{-1}$. The next result addresses this question for a wide class of approximations that includes the Nyström approximation.

PROPOSITION 3.1 (regularized inverses). *Consider psd matrices A, $\hat{A} \in \mathbb{S}_n^+(\mathbb{R})$, and assume that the difference $E = A - \hat{A}$ is psd. Fix $\mu > 0$. Then*

$$\|(\hat{A}+\mu I)^{-1} - (A+\mu I)^{-1}\| \leq \frac{1}{\mu}\frac{\|E\|}{\|E\|+\mu}. \tag{3.1}$$

Furthermore, the bound (3.1) *is attained when $\hat{A} = \lfloor A \rfloor_\ell$ for $1 \leq \ell \leq n$.*

The proof of Proposition 3.1 may be found in subsection SM1.1.1. It is based on [5, Lemma X.1.4].

Proposition 3.1 has an appealing interpretation. When $\|A - \hat{A}\|$ is small in comparison to the regularization parameter μ, the approximate inverse $(\hat{A}+\mu I)^{-1}$ can serve in place of the inverse $(A+\mu I)^{-1}$.

4. Nyström sketch-and-solve.

The simplest mechanism for using the Nyström approximation is an algorithm called Nyström sketch-and-solve. This section introduces the method, its implementation, and its history. We also provide a general theoretical analysis that sheds light on its performance. In spite of its popularity, the Nyström sketch-and-solve method is rarely worth serious consideration.

4.1. Overview.

Given a rank-ℓ Nyström approximation $\hat{A}_{\text{nys}}$ of the psd matrix A, it is tempting to replace the regularized linear system $(A+\mu I)x = b$ with the proxy $(\hat{A}_{\text{nys}} + \mu I)x = b$. Indeed, we can solve the proxy linear system in $O(\ell n)$ time using the Sherman–Morrison–Woodbury formula [17, eqn. (2.1.4)].

LEMMA 4.1 (approximate regularized inversion). *Consider any rank-ℓ matrix $\hat{A}$ with eigenvalue decomposition $\hat{A} = U\hat{\Lambda}U^T$. Then*

$$(\hat{A}+\mu I)^{-1} = U(\hat{\Lambda}+\mu I)^{-1}U^T + \frac{1}{\mu}(I - UU^T). \tag{4.1}$$

We refer to the approach in this paragraph as the Nyström sketch-and-solve algorithm because it is modeled on the sketch-and-solve paradigm that originated in [39].

See Algorithm 4.1 for a summary of the Nyström sketch-and-solve method. The algorithm produces an approximate solution $\hat{x}$ to the regularized linear system (1.1) in time $O(T_{\text{mv}}\ell + \ell^2 n)$. The arithmetic cost is much faster than a direct method, which costs $O(n^3)$. It can also be faster than running CG for a long time at a cost of $O(T_{\text{mv}})$ per iteration. The method looks attractive if we only consider the runtime, and yet...

Nyström sketch-and-solve only has one parameter, the rank ℓ of the Nyström approximation, which controls the quality of the approximate solution $\hat{x}$. When $\ell \ll n$, the method has an appealing computational profile. As ℓ increases, the approximation

Algorithm 4.1 Nyström sketch-and-solve

Input: Psd matrix $A \in \mathbb{S}_n^+(\mathbb{R})$, right-hand side b, regularization μ, rank ℓ
Output: Approximate solution $\hat{x}$ to (1.1)

1: $[U, \hat{\Lambda}] = \text{RandomizedNyströmApproximation}(A, \ell)$
2: Use (4.1) to compute $\hat{x} = (\hat{A}_{\text{nys}} + \mu I)^{-1} b$

quality increases, but the computational burden becomes heavy. We will show that, alas, an accurate solution to the linear system actually requires $\ell \approx n$, at which point the computational benefits of Nyström sketch-and-solve evaporate completely.

In summary, Nyström sketch-and-solve is almost never the right algorithm to use. We will see that Nyström PCG generally produces much more accurate solutions with a similar computational cost.

4.2. Guarantees and deficiencies. Using Proposition 3.1 together with the a priori guarantee in Proposition 2.2, we quickly obtain a performance guarantee for Algorithm 4.1.

THEOREM 4.2. *Fix $p \geq 2$, and set $\ell = 2p - 1$. For a psd matrix $A \in \mathbb{S}_n^+(\mathbb{R})$, construct a randomized Nyström approximation $\hat{A}_{\mathrm{nys}}$ using Algorithm* 2.1. *Then the approximation error for the inverse satisfies*

$$\mathbb{E}\big\|(A+\mu I)^{-1} - (\hat{A}_{\mathrm{nys}}+\mu I)^{-1}\big\| \leq \left(3 + \frac{4\mathrm{e}^2}{p}\mathrm{sr}_p(A)\right)\frac{\lambda_p}{\mu\cdot(\lambda_p+\mu)}. \tag{4.2}$$

Define $x_\star = (A+\mu I)^{-1}b$, and select $\ell = 2\lceil 1.5\, d_{\mathrm{eff}}(\epsilon\mu)\rceil + 1$. Then the approximate solution $\hat{x}$ computed by Algorithm 4.1 *satisfies*

$$\mathbb{E}\left[\frac{\|\hat{x}-x_\star\|_2}{\|x_\star\|_2}\right] \leq 26\epsilon. \tag{4.3}$$

The proof of Theorem 4.2 may be found in Appendix A.1.

Theorem 4.2 tells us how accurately we can hope to solve linear systems using Nyström sketch-and-solve (Algorithm 4.1). A sketch size $\ell = O(d_{\mathrm{eff}}(\epsilon\mu))$ is needed to guarantee relative error ϵ. When $\epsilon\mu$ is small, we anticipate that $d_{\mathrm{eff}}(\epsilon\mu) \approx n$. In this setting, Nyström sketch-and-solve has no computational value: it is as expensive as a direct method. As a concrete example, let $\mu = 10^{-4}$ and suppose we want six digits of accuracy, i.e., $\epsilon = 10^{-6}$. Then we must hope to find a sketch size ℓ so that $\lambda_\ell \sim 10^{-10}$ to achieve the required accuracy; and $\ell \ll n$ so the method offers a computational advantage. It is rare to find a matrix whose spectrum decays rapidly enough to satisfy both these constraints! Our analysis is sharp in its essential respects, so the pessimistic assessment is irremediable.

4.3. History. Nyström sketch-and-solve has a long history in the machine learning literature. It was introduced in [48] to speed up kernel-based learning, and it plays a role in many subsequent papers on kernel methods. In this context, the Nyström approximation is typically obtained using column sampling [1, 4, 48], which has its limitations (subsection 2.2.3). More recently, Nyström sketch-and-solve has been applied to speed up approximate cross-validation [43].

The analysis of Nyström sketch-and-solve presented above differs from previous analysis. Prior works [1, 4] focus on the kernel setting, and they use properties of column sampling schemes to derive learning guarantees. In contrast, we bound the relative error for a Nyström approximation based on a random projection. Our overall approach extends to column sampling if we replace Proposition 2.2 by an appropriate analogue, such as Gittens's results [15].

5. Nyström preconditioned conjugate Gradient. We now present our main algorithm, Nyström PCG. This algorithm produces high accuracy solutions to a regularized linear system by using the Nyström approximation $\hat{A}_{\mathrm{nys}}$ as a preconditioner. We provide a rigorous estimate for the condition number of the preconditioned system,

and we prove that Nyström PCG leads to fast convergence for regularized linear systems. In contrast, we have shown that Nyström sketch-and-solve cannot be expected to yield accurate solutions.

5.1. The preconditioner. In this section, we introduce the optimal low-rank preconditioner, and we argue that the randomized Nyström preconditioner provides an approximation that is easy to compute.

5.1.1. Motivation. As a warmup, suppose we knew the eigenvalue decomposition of the best rank-ℓ approximation of the matrix: $\lfloor A \rfloor_\ell = V_\ell \Lambda_\ell V_\ell^T$. How should we use this information to construct a good preconditioner for the regularized linear system (1.1)?

Consider the family of symmetric psd matrices that act as the identity on the orthogonal complement of range(V_ℓ). Within this class, we claim that the following matrix is the *optimal preconditioner*:

$$P_\star = \frac{1}{\lambda_{\ell+1} + \mu} V_\ell(\Lambda_\ell + \mu I)V_\ell^T + (I - V_\ell V_\ell^T). \tag{5.1}$$

The optimal preconditioner $P_\star$ requires $O(n\ell)$ storage, and we can solve linear systems in $P_\star$ in $O(n\ell)$ time. Whereas the regularized matrix A_μ has condition number $\kappa_2(A_\mu) = (\lambda_1 + \mu)/(\lambda_n + \mu)$, the preconditioner yields

$$\kappa_2(P_\star^{-1/2} A_\mu P_\star^{-1/2}) = \frac{\lambda_{\ell+1} + \mu}{\lambda_n + \mu}. \tag{5.2}$$

This is the minimum possible condition number attainable by a preconditioner from the class that we have delineated. It represents a significant improvement when $\lambda_{\ell+1} \ll \lambda_1$. The proofs of these claims are straightforward; for details, see subsection SM1.1.2.

5.1.2. Randomized Nyström preconditioner. It is expensive to compute the best rank-ℓ approximation $\lfloor A \rfloor_\ell$ accurately. In contrast, we can compute the rank-ℓ randomized Nyström approximation $\hat{A}_{\text{nys}}$ efficiently (Algorithm 2.1). Furthermore, we have seen that $\hat{A}_{\text{nys}}$ approximates A nearly as well as the optimal rank-ℓ approximation (Corollary 2.3). These facts lead us to study the randomized Nyström preconditioner, proposed in [25, sec. 17] without a complete justification.

Consider the eigenvalue decomposition $\hat{A}_{\text{nys}} = U\hat{\Lambda}U^T$, and write $\hat{\lambda}_\ell$ for its ℓth eigenvalue. The randomized Nyström preconditioner and its inverse take the form

$$\begin{aligned} P &= \frac{1}{\hat{\lambda}_\ell + \mu} U(\hat{\Lambda} + \mu I)U^T + (I - UU^T); \\ P^{-1} &= (\hat{\lambda}_\ell + \mu)U(\hat{\Lambda} + \mu I)^{-1}U^T + (I - UU^T). \end{aligned} \tag{5.3}$$

Like the optimal preconditioner $P_\star$, the randomized Nyström preconditioner (5.3) is cheap to apply and to store. We may hope that it damps the condition number of the preconditioned system $P^{-1/2} A_\mu P^{-1/2}$ nearly as well as the optimal preconditioner $P_\star$. We will support this intuition with a rigorous bound (Proposition 5.3).

5.2. Nyström PCG. We can obviously use the randomized Nyström preconditioner within the framework of PCG. We call this approach Nyström PCG, and we present a basic implementation in Algorithm 5.1. In the case of very ill-conditioned least-squares problems, it is sometimes preferable to use other Krylov methods such as LSQR [32] over CG. We have not found the need to use such methods as we focus

Algorithm 5.1 Nyström PCG

Input: Psd matrix A, right-hand side b, initial guess x_0, regularization parameter μ, sketch size ℓ, solution tolerance η
Output: Approximate solution $\hat{x}$ to regularized system (1.1)

$[U, \hat{\Lambda}] = \text{RandomizedNyströmApproximation}(A, \ell)$
$r_0 = b - (A + \mu I)x_0$
$z_0 = P^{-1}r_0$ ▷ using (5.3)
$p_0 = z_0$
while $\|r\|_2 > \eta$
  $v = (A + \mu I)p_0$
  $\alpha = (r_0^T z_0)/(p_0^T v_0)$ ▷ compute step size
  $x = x_0 + \alpha p_0$ ▷ update solution
  $r = r_0 - \alpha v$ ▷ update residual
  $z = P^{-1}r$ ▷ find search direction via (5.3)
  $\beta = (r^T z)/(r_0^T z_0)$
  $x_0 \leftarrow x$, $r_0 \leftarrow r$, $p_0 \leftarrow z + \beta p_0$, $z_0 \leftarrow z$

on regularized problems and are preconditioning, so that $\kappa_2(P^{-1/2}A_\mu P^{-1/2}) \ll u^{-1}$, where u is machine precision, Nevertheless, the Nyström precondtioner is easily extended to LSQR; one may use $P^{-1/2}$ as a right-preconditioner with P as in (5.3).

More precisely, Algorithm 5.1 uses left-preconditioned CG. This algorithm implicitly works with the unsymmetric matrix $P^{-1}A_\mu$, rather than the symmetric matrix $P^{-1/2}A_\mu P^{-1/2}$. The two variants of PCG yield identical sequences of iterates [38], but the former is more efficient. For ease of analysis, our theoretical results are presented in terms of the symmetrically preconditioned matrix.

5.2.1. Complexity of Nyström PCG. Nyström PCG first constructs the randomized Nyström approximation and then solves the regularized linear system with PCG. We have already discussed the cost of constructing the Nyström approximation (subsection 2.2.1). PCG requires $O(T_{\text{mv}})$ operations per iteration, and it uses a total of $O(n)$ additional storage.

For the regularized linear system (1.1), Theorem 5.1 and Corollary 5.2 demonstrate that it suffices to choose the sketch size $\ell = 2\lceil 1.5 d_{\text{eff}}(\mu)\rceil + 1$. In this case with high probability, the overall runtime needed for Nyström PCG to obtain ϵ-relative error in the A_μ-norm is

$$O\big(d_{\text{eff}}(\mu)^2 n + T_{\text{mv}}(d_{\text{eff}}(\mu) + \log(1/\epsilon))\big) \quad \text{operations.}$$

When the effective dimension $d_{\text{eff}}(\mu)$ is modest, Nyström PCG is very efficient.

In contrast, subsection 4.2 shows that the running time for Nyström sketch-and-solve has the same form—with $d_{\text{eff}}(\epsilon\mu)$ in place of $d_{\text{eff}}(\mu)$. That is, Nyström PCG can produce solutions whose residual norm is close to machine precision, whereas it is impossible to obtain high precision solutions efficiently with Nyström sketch-and-solve.

5.2.2. Comparison to other randomized preconditioning methods. Here we discuss Nyström PCG in the context of prior work on randomized preconditioning [3, 18, 22, 28, 36] based on sketch-and-precondition and related ideas. All these prior methods were developed for least-squares problems. We summarize the complexity of each method for regularized least-squares problems in Table 1.

TABLE 1

Regularized least-squares: Complexity of prior randomized preconditoning methods versus Nyström PCG. The table compares the complexity of Nyström PCG and state-of-the-art randomized preconditioning methods in the overdetermined case $n \geq d$, assuming we can access A only via matrix-vector products. The sketch-and-precondition preconditioner is constructed from a sketch SA, where $S \in \mathbb{R}^{m \times n}$ is a $(1 \pm \gamma)$ Gaussian subspace embedding with sketch size $\Omega(d/\gamma)$ and $\gamma \in (0, 1)$. The time to compute the sketch is $O(T_{mv} d/\gamma)$ and the iteration complexity follows from the argument in [50, *sect.* 2.6]. *For AdaIHS, we use a sketch constructed from a Gaussian subspace embedding with sketch size $O(d_{\mathit{eff}}(\mu)/\rho)$ where $\rho \in (0, 0.18)$. The complexity of AdaIHS follows from* [22, *Theorem* 5]. *Similarly, the construction in* [18] *uses a Gaussian test matrix. Let $\tilde{\kappa}_\ell = (\ell \lambda_\ell + \sum_{j>\ell} \lambda_j)/\mu$. Then the overall runtime of sketched-preconditioned SVRG follows from* [18, *Theorem* 1] *and the runtime of randomized block Krylov method used to construct the preconditioner* [29, 25]. *The complexity of Nyström PCG is derived from Theorem* 5.1 *and Corollary* 5.2.

Method	**Complexity**	**References**
Sketch-and-precondition	$O\left(T_{\mathrm{mv}} d/\gamma + d^3/\gamma + T_{\mathrm{mv}} \frac{\log(1/\epsilon)}{\log(1/\gamma)}\right)$	[3, 28, 36]
AdaIHS	$O\left((T_{\mathrm{mv}} d_{\mathrm{eff}}/\rho + d d_{\mathrm{eff}}^2/\rho^2) \log(d_{\mathrm{eff}}/\rho) + T_{\mathrm{mv}} \frac{\log(1/\epsilon)}{\log(1/\rho)}\right)$	[22]
Sketched preconditioned	$O(T_{\mathrm{mv}} \ell \log(n) + d\ell^2 \log^2(n))$	[18]
SVRG	$+O(T_{\mathrm{mv}} + \tilde{\kappa}_\ell + d^2) \log(1/\epsilon))$	
Nyström PCG	$O(T_{\mathrm{mv}} d_{\mathrm{eff}} + d d_{\mathrm{eff}}^2 + T_{\mathrm{mv}} \log(1/\epsilon))$	This work

The time to construct the sketch-and-precondition preconditioner is always larger than that of the Nyström preconditioner, since $d_{\mathrm{eff}} < d$ and $\gamma < 1$. Indeed, constructing the preconditioner for sketch-and-precondition costs $\Omega(d^3)$, which is the same as a direct method when $d = \Omega(n)$ and is prohibitive for high-dimensional problems. In contrast, Nyström PCG is amenable to problems with large d and runs much faster than sketch-and-precondition whenever $d_{\mathrm{eff}}(\mu) \ll d$. We note the analysis of sketch-and-precondition can likely be improved to require only a sketch size of $O(d_{\mathrm{eff}}(\mu)/\gamma)$. However, this improvement by itself is only of theoretical value as $d_{\mathrm{eff}}(\mu)$ is almost never known beforehand. Thus, without an adaptive scheme or method to estimate $d_{\mathrm{eff}}(\mu)$, the best a priori sketch size one can select with sketch-and-precondition methods is $O(d/\gamma)$. The Nyström preconditioner also enjoys wider applicability then sketch-and-precondition: it applies to square-ish systems, whereas the others only work for strongly overdetermined or underdetermined problems. Nyström PCG also improves slightly on the complexity of AdaIHS: while both scale linearly in d, Nyström PCG removes unnecessary logarithmic factors and the constant $\rho < 0.18$.

Of the methods presented in Table 1, sketched preconditioned SVRG [18] is closest to our approach. The authors of [18] construct a preconditioner from a randomized low-rank approximation to be deployed with the SVRG algorithm [19]. However, while both use a randomized low-rank approximation to construct the preconditioner, the methods differ significantly. In particular, [18] constructs the randomized preconditioner using the randomized block Krylov scheme in [29], which is significantly more expensive than Algorithm 2.1 used for Nyström PCG. Indeed, the randomized block Krylov scheme requires ℓ-matvecs with A $O(\log(n))$ times and $O(\log(n))$ costly orthogonalizations, which are needed for numerical stability [25]. Hence sketched preconditioned SVRG is considerably slower than Nyström PCG; see Table 1. Moreover, the theory in [18] also lacks any connection with the effective dimension and provides no theoretical or practical guidance for selecting the rank ℓ. Last, note SVRG (unlike PCG) is typically used in settings where a full pass through the data, i.e., a matvec, is too expensive. A preconditioner that requires multiple full passes through the data is an odd choice in this setting.

In the context of kernel ridge regression (KRR), the random features method of [2] may be viewed as a randomized preconditioning technique. The authors of [2] prove convergence guarantees for the polynomial kernel with a (large) sketch size $\ell = O(d_{\text{eff}}(\mu)^2)$. In contrast, Nyström PCG can be used for KRR with any kernel and requires only the smaller sketch size $\ell = O(d_{\text{eff}}(\mu))$ to obtain fast convergence.

Finally, in a pure statistical learning setting, where the primary concern is test-set error and not residual tolerance, fast approximate methods for KRR are also available. The current state of the art is the Falkon algorithm from [37], which shares important commonalities with Nyström sketch-and-solve. Instead of working with full kernel, it works with $K_{n\ell} \in \mathbb{R}^{n\times\ell}$, where $K_{n\ell}$ is computed with respect to ℓ-centers randomly sampled from the training set. Let $d^{\star}_{\text{eff}}(\mu)$ denote the effective dimension of the population kernel. Then under appropriate conditions and with $\ell = O(d^{\star}_{\text{eff}}(\mu))$, [37] shows Falkon obtains generalization error comparable to that of exact methods, with runtime $O(nd^{\star}_{\text{eff}}(\mu)\log(n) + d^{\star}_{\text{eff}}(\mu)^3)$. Thus, in principle, Falkon should run much faster than Nyström PCG or random features PCG from [2] and yield nearly identical statistical performance on the test set. We have found Falkon does run faster, but there are gaps in performance relative to Nyström PCG that cannot be improved by increasing the number of centers. That is, to obtain the best statistical performance, there is still a benefit to solving the problem to modest accuracy. See subsection 6.5 for numerical comparison and further discussion. Furthermore, Falkon only applies to vanilla KRR and kernelized logistic regression [26], and not to Gaussian processes, an application where Nyström PCG might prove useful. Moreover, the Gaussian processes literature [12, 47] has found exact inference yields better learning performance than approximate methods.

In summary, Nyström PCG applies to a wider class of problems than prior randomized preconditioners and enjoys stronger theoretical guarantees for regularized problems. Nyström PCG also outperforms other randomized preconditioners numerically (section 6).

5.2.3. Block Nyström PCG. The Nyström preconditioner can also precondition the block CG algorithm [31] to solve regularized linear systems with multiple right-hand sides, as appear in applications to approximate cross validation [43], influence functions [20], and hyperparameter optimization [24]. Blocking provides advantages both in convergence rate and in memory management. The orthogonalization preprocessing proposed in [11] ensures numerical stability for block Nyström PCG without further orthogonalization steps during the iteration.

5.3. Analysis of Nyström PCG. We now turn to the analysis of the randomized Nyström preconditioner P. Theorem 5.1 provides a bound for the rank ℓ of the Nyström preconditioner that reduces the condition number of A_μ to a constant. In this case, we deduce that Nyström PCG converges rapidly (Corollary 5.2).

THEOREM 5.1 (Nyström preconditioning). *Suppose we construct the Nyström preconditioner P in* (5.3) *using Algorithm* 2.1 *with sketch size $\ell = 2\lceil 1.5\, d_{\text{eff}}(\mu)\rceil + 1$. Using P to precondition the regularized matrix A_μ results in the condition number bound*

$$\mathbb{E}\big[\kappa_2(P^{-1/2}A_\mu P^{-1/2})\big] < 28.$$

The proof of Theorem 5.1 may be found in subsection 5.3.3.

Theorem 5.1 has several appealing features. Many other authors have noticed that the effective dimension controls sample size requirements for particular applications

such as discriminant analysis [7], ridge regression [22], and kernel ridge regression [1, 4]. In contrast, our result holds for any regularized linear system.

Our argument makes the role of the effective dimension conceptually simpler, and it leads to explicit, practical parameter recommendations. Indeed, the effective dimension $d_{\text{eff}}(\mu)$ is essentially the same as the sketch size ℓ that makes the approximation error $\|A - \hat{A}_{\text{nys}}\|$ proportional to μ. In previous arguments, such as those in [1, 4, 7], the effective dimension arises because the authors reduce the analysis to approximate matrix multiplication [8], which produces inscrutable constant factors.

We also note that Theorem 5.1 easily extends to the column sampling schemes using Proposition 5.3 and results from [1] to control $\|E\|$. This is particularly attractive for kernel problems, as the Nyström preconditioner may be constructed in $O(n\ell^2)$ operations. For the case of uniform column sampling, the key quantity is the maximal marginal degrees of freedom

$$d_{\text{mof}}(\mu) = n\|\text{diag}(A(A + n\mu I)^{-1})\|_\infty.$$

Clearly, $d_{\text{mof}}(\mu) \geq d_{\text{eff}}(\mu)$, and is generally significantly larger. Combining our results with those from [1], we can conclude a similar result to Theorem 5.1 using a rank of size $\ell = O(d_{\text{mof}}(\mu)\log(n))$. Thus the guarantees for uniform column sampling are considerably worse than those of random projection. In practice we have found the bound on ℓ for uniform column sampling to be very pessimistic; see subsection 6.5 for corroborating numerical evidence.

Theorem 5.1 ensures that Nyström PCG converges quickly.

COROLLARY 5.2 (Nyström PCG: Convergence). *Define P as in Theorem 5.1, and condition on the event $\{\kappa_2(P^{-1/2}A_\mu P^{-1/2}) \leq 56\}$. If we initialize Algorithm 5.1 with initial iterate $x_0 = 0$, then the relative error δ_t in the iterate x_t satisfies*

$$\delta_t = \frac{\|x_t - x_\star\|_{A_\mu}}{\|x_\star\|_{A_\mu}} < 2 \cdot (0.77)^t, \quad \text{where } A_\mu x_\star = b.$$

In particular, after $t = \lceil 3.8\log(2/\epsilon)\rceil$ iterations, we have relative error $\delta_t < \epsilon$.

The proof of Corollary 5.2 is an immediate consequence of the standard convergence result for CG [44, Theorem 38.5, p. 299]. See Appendix A.2. Note that Corollary 5.2 also immediately implies the total number of matvecs required to reach an ϵ-accurate solution in the A-norm.

5.3.1. Analyzing the condition number. The first step in the proof of Theorem 5.1 is a deterministic bound on how the preconditioner (5.3) reduces the condition number of the regularized matrix A_μ. Let us emphasize that this bound is valid for any rank-ℓ Nyström approximation, regardless of the choice of test matrix.

PROPOSITION 5.3 (Nyström preconditioner: Deterministic bound). *Let $\hat{A} = U\hat{\Lambda}U^T$ be any rank-ℓ Nyström approximation, with ℓth largest eigenvalue $\hat{\lambda}_\ell$, and let $E = A - \hat{A}$ be the approximation error. Construct the Nyström preconditioner P as in (5.3). Then the condition number of the preconditioned matrix $P^{-1/2}A_\mu P^{-1/2}$ satisfies*

$$\begin{aligned} \max\left\{\frac{\hat{\lambda}_\ell + \mu}{\lambda_n + \mu}, 1\right\} &\leq \kappa_2(P^{-1/2}A_\mu P^{-1/2}) \\ &\leq \left(\hat{\lambda}_\ell + \mu + \|E\|\right)\min\left\{\frac{1}{\mu}, \frac{\hat{\lambda}_\ell + \lambda_n + 2\mu}{(\hat{\lambda}_\ell + \mu)(\lambda_n + \mu)}\right\}. \end{aligned} \tag{5.4}$$

For the proof of Proposition 5.3 see Appendix A.1.1. It turns out that the only properties of the Nyström approximation we require in Proposition 5.3 are that $\hat{A}$ is psd and $E = A - \hat{A} \succeq 0$. Thus, Proposition 5.3 also applies to any preconditioner of the form (5.3) constructed from a matrix $\hat{A}$ that possesses these two properties.

To interpret the result, recall the expression (5.2) for the condition number induced by the optimal preconditioner. Proposition 5.3 shows that the Nyström preconditioner may reduce the condition number almost as well as the optimal preconditioner. Equation (5.4) shows that the price we pay for using an efficiently computable preconditioner is the condition number of the preconditioned system depending upon our approximation error $\|E\|$. This is natural given that the preconditioner is constructed from $\hat{A}$, a perturbed version of A. Hence we expect P to behave like a perturbed version of $P_\star$, which is precisely the content of Proposition 5.3.

In particular, when $\|E\| = O(\mu)$, the condition number of the preconditioned system is bounded by a constant, independent of the spectrum of A. This follows as $\hat{\lambda}_\ell \leq \lambda_\ell$ and $\|E\|$ dominates λ_ℓ. In this setting, Nyström PCG is guaranteed to converge quickly.

5.3.2. The effective dimension and sketch size selection. How should we choose the sketch size ℓ to guarantee that $\|E\| = O(\mu)$? Corollary 2.3 shows how the error in the rank-ℓ randomized Nyström approximation depends on the spectrum of A through the eigenvalues of A and the tail stable rank. In this section, we present a lemma which demonstrates that the effective dimension $d_{\text{eff}}(\mu)$ controls both quantities. As a consequence of this bound, we will be able to choose the sketch size ℓ proportional to the effective dimension $d_{\text{eff}}(\mu)$.

Recall from (1.4) that the effective dimension of the matrix A is defined as

$$d_{\text{eff}}(\mu) = \operatorname{tr}(A(A+\mu I)^{+}) = \sum_{j=1}^{n} \frac{\lambda_j(A)}{\lambda_j(A)+\mu}. \tag{5.5}$$

As previously mentioned, $d_{\text{eff}}(\mu)$ may be viewed as a smoothed count of the eigenvalues larger than μ. Thus, one may expect that $\lambda_k(A) \lesssim \mu$ for $k \gtrsim d_{\text{eff}}(\mu)$. This intuition is correct, and it forms the content of Lemma 5.4.

LEMMA 5.4 (effective dimension). *Let $A \in \mathbb{S}_n^+(\mathbb{R})$ with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. Let $\mu > 0$ be the regularization parameter, and define the effective dimension as in* (5.5). *The following statements hold.*

1. *Fix $\gamma > 0$. If $j \geq (1+\gamma^{-1}) d_{\text{eff}}(\mu)$, then $\lambda_j \leq \gamma\mu$.*
2. *If $k \geq d_{\text{eff}}(\mu)$, then $k^{-1} \sum_{j>k} \lambda_j \leq (d_{\text{eff}}(\mu)/k) \cdot \mu$.*

The proof of Lemma 5.4 may be found in Appendix A.1.2.

Lemma 5.4, item 1 captures the intuitive fact that there are no more than $2d_{\text{eff}}(\mu)$ eigenvalues larger than μ. Similarly, item 2 states that the effective dimension controls the sum of all the eigenvalues whose index exceeds the effective dimension. It is instructive to think about the meaning of these results when $d_{\text{eff}}(\mu)$ is small.

5.3.3. Proof of Theorem 5.1. We are now prepared to prove Theorem 5.1. The key ingredients in the proof are Proposition 2.2, Proposition 5.3, and Lemma 5.4.

Proof of Theorem 5.1. Fix the sketch size $\ell = 2\lceil 1.5\, d_{\text{eff}}(\mu) \rceil + 1$. Construct the rank-$\ell$ randomized Nyström approximation $\hat{A}_{\text{nys}}$ with eigenvalues $\hat{\lambda}_j$. Write $E = A - \hat{A}_{\text{nys}}$ for the approximation error. Form the preconditioner P via (5.3). We must bound the expected condition number of the preconditioned matrix $P^{-1/2} A_\mu P^{-1/2}$.

First, we apply Proposition 5.3 to obtain a deterministic bound that is valid for any rank-ℓ Nyström preconditioner:

$$\kappa_2(P^{-1/2}A_\mu P^{-1/2}) \le \frac{\hat{\lambda}_\ell + \mu + \|E\|}{\mu} \le 2 + \frac{\|E\|}{\mu}.$$

The second inequality holds because $\hat{\lambda}_\ell \le \lambda_\ell \le \mu$. This is a consequence of Lemma 2.1, item 4 and Lemma 5.4, item 1 with $\gamma = 1$. We rely on the fact that $\ell \ge 2\, d_{\mathrm{eff}}(\mu)$.

Decompose $\ell = 2p - 1$ where $p = \lceil 1.5\, d_{\mathrm{eff}}(\mu) \rceil + 1$. Take the expectation, and invoke Corollary 2.3 to obtain

$$\mathbb{E}\big[\kappa_2(P^{-1/2}A_\mu P^{-1/2})\big] \le 2 + \left(3 + \frac{4\mathrm{e}^2}{p}\mathrm{sr}_p(A)\right)(\lambda_p/\mu).$$

By definition, $\mathrm{sr}_p(A) \cdot \lambda_p = \sum_{j \ge p} \lambda_j$. To complete the bound, apply Lemma 5.4 twice. We use item 1 with $\gamma = 2$ and item 2 with $k = p - 1 = \lceil 1.5\, d_{\mathrm{eff}}(\mu) \rceil$ to reach

$$\mathbb{E}\big[\kappa_2(P^{-1/2}A_\mu P^{-1/2})\big] \le 2 + \frac{3 \cdot 2\mu + 4\mathrm{e}^2 \cdot 2\mu/3}{\mu} < 2 + 26 = 28,$$

which is the desired result. ∎

5.4. Practical parameter selection. In practice, we may not know the regularization parameter μ in advance, and we rarely know the effective dimension $d_{\mathrm{eff}}(\mu)$. As a consequence, we cannot enact the theoretical recommendation for the rank of the Nyström preconditioner: $\ell = 2\lceil 1.5\, d_{\mathrm{eff}}(\mu) \rceil + 1$. Instead, we need an adaptive method for choosing the rank ℓ. Below, we outline three strategies.

5.4.1. Strategy 1: Adaptive rank selection by a posteriori error estimation. The first strategy uses the posterior condition number estimate adaptively in a procedure the repeatedly doubles the sketch size ℓ as required. Recall that Proposition 5.3 controls the condition number of the preconditioned system:

$$\kappa_2(P^{-1/2}A_\mu P^{-1/2}) \le \frac{\hat{\lambda}_\ell + \mu + \|E\|}{\mu}, \quad \text{where } E = A - \hat{A}_{\mathrm{nys}}. \tag{5.6}$$

We get $\hat{\lambda}_\ell$ for free from Algorithm 2.1, and we can compute the error $\|E\|$ inexpensively with the randomized power method [21]; see Algorithm SM4.1 in section SM4. Thus, we can ensure the condition number is small by making $\|E\|$ and $\hat{\lambda}_\ell$ fall below some desired tolerance. The adaptive strategy proceeds to do this as follows. We compute a randomized Nyström approximation with initial sketch size ℓ_0, and we estimate the error $\|E\|$ using randomized powering. If $\|E\|$ is smaller than a prescribed tolerance, then we accept the rank-ℓ_0 approximation. If the tolerance is not met, then we double the sketch size, update the approximation, and estimate $\|E\|$ again. The process repeats until the estimate for $\|E\|$ falls below the tolerance or it breaches a threshold $\ell_{\max}$ for the maximum sketch size. Algorithm C.1 uses the following stopping criteria $\|E\| \le \mathrm{Tol}_{\mathrm{Err}}$ and $\hat{\lambda}_\ell \le \mathrm{Tol}_{\mathrm{Rat}}$ for tolerances $\mathrm{Tol}_{\mathrm{Err}}$ and $\mathrm{Tol}_{\mathrm{Rat}}$. The stopping criterion on $\hat{\lambda}_\ell$ does not seem to be necessary in practice, as it is usually an order of magnitude small than $\|E\|$, but it is needed for Theorem 5.5. Based on numerical experience, we recommend the choices $\mathrm{Tol}_{\mathrm{Err}} = \tau\mu, \mathrm{Tol}_{\mathrm{Rat}} = \tau\mu/10$ for $\tau \in [1, 100]$. For full algorithmic details of adaptive rank selection by estimating $\|E\|$, see Algorithm C.1 in Appendix C.

The following theorem shows that with high probability, Algorithm C.1 terminates with a modest sketch size in at most a logarithmic number of steps, and PCG with the resulting preconditioner converges rapidly.

THEOREM 5.5. *Run Algorithm* C.1 *with initial sketch size* ℓ_0 *and tolerances* $\mathrm{Tol}_{\mathrm{Err}} = \tau\mu, \mathrm{Tol}_{\mathrm{Rat}} = \tau\mu/11$, *where* $\tau \geq 1$, *and let* $\tilde{\ell} = 2\lceil 2d_{\mathrm{eff}}(\frac{\delta\tau\mu}{11})\rceil + 1$. *Then with probability at least* $1-\delta$ *the following hold:*

1. *Algorithm* C.1 *doubles the sketch size at most* $\lceil \log_2(\frac{\tilde{\ell}}{\ell_0})\rceil$ *times.*
2. *The final sketch size* ℓ *satisfies*

$$\ell \leq 4\left\lceil 2d_{\mathrm{eff}}\left(\frac{\delta\tau\mu}{11}\right)\right\rceil + 2.$$

3. *With the preconditioner constructed from Algorithm* C.1, *Nyström PCG converges in at most* $\lceil \frac{\log(2/\epsilon)}{\log(1/\tau_0)}\rceil$ *iterations, where* $\tau_0 = \frac{\sqrt{1+12\tau/11}-1}{\sqrt{1+12\tau/11}+1}$.

Theorem 5.5 immediately implies the following concrete guarantee.

COROLLARY 5.6. *Set* $\tau = 44$ *and* $\delta = 1/4$ *in Algorithm* C.1. *Then with probability at least* $3/4$ *the folloiwng hold:*

1. *Algorithm* C.1 *doubles the sketch size at most* $\lceil \log_2(\frac{\tilde{\ell}}{\ell_0})\rceil$ *times.*
2. *The final sketch size* ℓ *satisfies*

$$\ell \leq 4\lceil 2d_{\mathrm{eff}}(\mu)\rceil + 2.$$

3. *With the preconditioner constructed from Algorithm* C.1, *Nyström PCG converges in at most* $\lceil 3.48\log(2/\epsilon)\rceil$ *iterations.*

5.4.2. Strategy 2: Adaptive rank selection by monitoring $\hat{\lambda}_\ell/\mu$. The second strategy is almost identical to the first, except we monitor the ratio $\hat{\lambda}_\ell/\mu$ instead of $\|E\|/\mu$. Strategy 2 doubles the approximation rank until $\hat{\lambda}_\ell/\mu$ falls below some tolerance (say, 10) or the sample size reaches the threshold $\ell_{\max}$. The approach is justified by the following proposition, which shows that once the rank ℓ is sufficiently large, with high probability, the exact condition number differs from the empirical condition number $(\hat{\lambda}_\ell + \mu)/\mu$ by at most a constant.

PROPOSITION 5.7. *Let* $\tau \geq 0$ *denote the tolerance and* $\delta > 0$ *a given failure probability. Suppose the rank of the randomized Nyström approximation satisfies* $\ell \geq 2\lceil 2d_{\mathrm{eff}}(\tau\mu))\rceil + 1$. *Then*

$$\mathbb{P}\left\{\left(\kappa_2(P^{-1/2}A_\mu P^{-1/2}) - \frac{\hat{\lambda}_\ell + \mu}{\mu}\right)_+ \leq \frac{\tau}{\delta}\right\} \geq 1-\delta, \tag{5.7}$$

where $X_+ = \max\{X, 0\}$.

This strategy has the benefit of saving a bit of computation and is preferable when a moderately small residual is sufficient, e.g., in machine learning problems where training error only loosely predicts test error.

5.4.3. Strategy 3: Choose ℓ as large as the user can afford. The third strategy is to choose the rank ℓ as large as the user can afford. This approach is coarse, and it does not yield any guarantees on the cost of the Nyström PCG method.

Nevertheless, once we have constructed a rank-ℓ Nyström approximation we can combine the posterior estimate of the condition number used in strategy 1 with the

standard convergence theory of PCG to obtain an upper bound for the iteration count of Nyström PCG. This gives us advance warning about how long it may take to solve the regularized linear system. As in strategy 1 we compute the error $\|E\|$ in the condition number bound inexpensively with the randomized power method.

6. Applications and experiments. In this section, we study the performance of Nyström PCG on real-world data from three different applications: ridge regression, kernel ridge regression, and approximate cross-validation. The experiments demonstrate the effectiveness of the preconditioner and our strategies for choosing the rank ℓ compared to other algorithms in the literature: on large datasets, we find that our method outperforms competitors by a factor of 5–10 (Tables 3 and 8).

6.1. Preliminaries. We implemented all experiments in MATLAB R2019a and MATLAB R2021a on a server with 128 Intel Xeon E7-4850 v4 2.10 GHz CPU cores and 1056GB. Except for the very large scale datasets ($n \geq 10^5$), every numerical experiment in this section was repeated twenty times; tables report the mean over the twenty runs, and the standard deviation (in parentheses) when it is nonzero. We highlight the best-performing method in a table in bold.

We select hyperparameters of competing methods by grid search to optimize performance. This procedure tends to be very charitable to the competitors, and it may not be representative of their real-world performance. Indeed, grid search is computationally expensive, and it cannot be used as part of a practical implementation. A detailed overview of the experimental setup for each application may be found in the appropriate section of section SM2, and additional numerical results in section SM3.

6.2. Ridge regression. In this section, we solve the ridge regression problem (1.7) described in subsection 1.3 on some standard machine learning data sets (Table 2) from OpenML [46] and LIBSVM [6]. The effective dimension $d_{\text{eff}}(\mu)$ and the numerical rank of these matrices provide insight into the difficulty of each problem. These are reported in Table 2. We compare Nyström PCG to standard CG and two randomized preconditioning methods, the sketch-and-precondition method of Rokhlin and Tygert (R&T) [36] and the Adaptive Iterative Hessian Sketch (AdaIHS) [22].

6.2.1. Experimental overview. We perform two sets of experiments: computing regularization paths on CIFAR-10 and Guillermo, and random features regression [34, 35] on shuttle, smallNORB, Higgs, YearMSD, and covtype with specified values of μ. The values of μ may be found in subsection SM2.1. We use the Euclidean

TABLE 2

Ridge regression dataset statistics: The table reports the effective dimension and numerical rank (in double precision) of each dataset. For CIFAR-10 and Guillermo, we report $d_{\text{eff}}(\mu)$ using the value of μ on the regularization path that yields the best test error. The numerical rank (NumRank) of a matrix $A \in \mathbb{R}^{n\times n}$ with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_k$ is $\max\{k : \lambda_k \geq \lambda_1\epsilon\}$, the number of eigenvalues larger than machine precision ϵ scaled by the spectral norm of A.

Dataset	n	d	$d_{\text{eff}}(\mu)$	NumRank
CIFAR-10	40,000	3,072	1,258	3,072
Guillermo	16,000	4,297	1,885	2,000
smallNorb-rf	24,300	10,000	1,806	6,812
shuttle-rf	43,300	10,000	439	853
Higgs-rf	800,000	10,000	936	10,000
YearMSD-rf	463,715	15,000	7,262	15,000
covtype-binary	464,810	15,000	14,629	15,000

norm $\|r\|_2$ of the residual as our stopping criteria and declare convergence when $\|r\|_2 \leq 10^{-10}$. For both sets of experiments, we use Nyström PCG with adaptive rank selection (Algorithm C.1 in Appendix C). For experimental details, see subsection SM2.1.

The regularization path experiments solve (1.7) over a regularization path $\mu = 10^j$, where $j = 3, \ldots, -6$. We first solve the problem for the largest μ and then solve for progressively smaller μ by warm starting from the previous solution. We allow every method at most 500 iterations to reach the desired tolerance, for each value of μ.

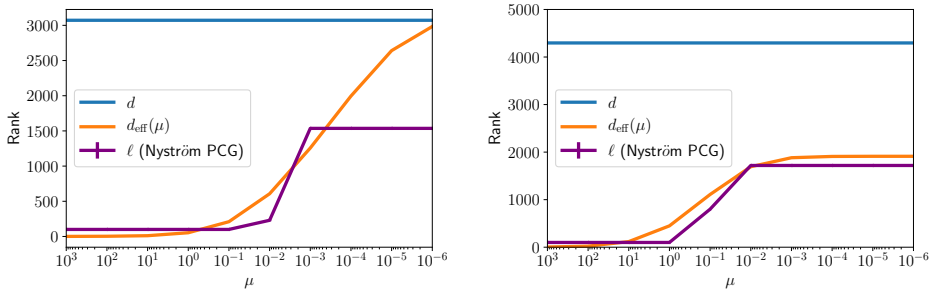
We report the test error achieved on each dataset in Appendix B.1. We also compare to the test-error obtained by a sketch-and-solve approach that approximates the inverse using the Nyström preconditioner, and which is known to admit good learning guarantees under appropriate conditions [1, 4].

6.2.2. Computing the regularization path. Figure 2 shows how the effective dimension $d_{\text{eff}}(\mu)$ varies with the regularization parameter μ on two small datasets. We see that the effective dimension reaches our chosen maximum sketch size, $\ell_{\max} = 0.5d$ for CIFAR-10 and $\ell_{\max} = 0.4d$ for Guillermo, when μ is small enough. For CIFAR-10, Nyström PCG chooses a rank much smaller than the effective dimension for small values of μ, yet the method still performs well (Figure 3).

Figure 3 show the effectiveness of each method for computing the entire regularization path. Nyström PCG is the fastest almost uniformly. The one exception is on CIFAR-10, where R&T performs better for the smallest regularization parameter, for which $d_{\text{eff}}(\mu) \approx d$. That is, the $O(d^3)$ cost of forming the R&T preconditioner is not worthwhile unless $d_{\text{eff}}(\mu) \approx d$ and the regularization is negligible.

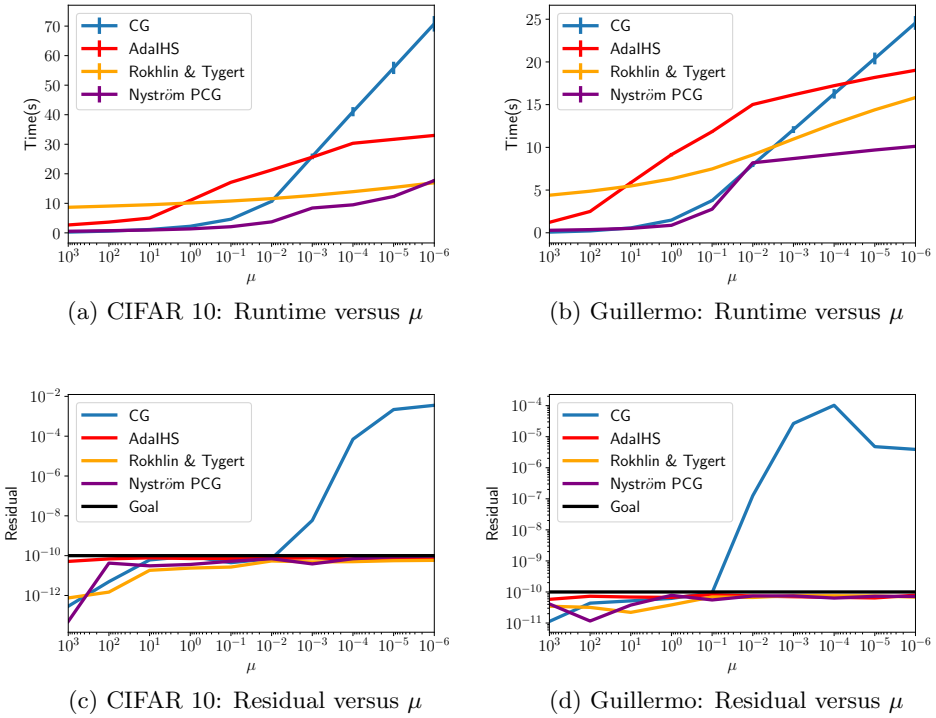
AdaIHS is rather slow. It increases the sketch size parameter several times along the regularization path. Each time, AdaIHS must form a new sketch of the matrix, approximate the Hessian, and compute a Cholesky factorization.

6.2.3. Random features regression. Table 3 compares the performance of Nyström PCG, AdaIHS, and R&T PCG for random features regression. Table 3 shows that Nyström PCG performs best on all datasets for all metrics. The most striking feature is the difference between sketch sizes: AdaIHS and R&T require much larger sketch sizes than Nyström PCG, leading to greater computation time and higher storage costs. Table 3, in conjunction with Table 2, shows the adaptive



(a) CIFAR 10 (b) Guillermo

FIG. 2. Ridge regression: Adaptive sketch size selection. *Nyström PCG with adaptive rank selection (Algorithm* C.1*) selects a preconditioner whose rank is less than or equal to the effective dimension. Error bars for the rank selected by the adaptive algorithm are so small that they are not visible in the graph: The behavior of the adaptive algorithm is essentially deterministic across runs. See subsection* 6.2.2.



(a) CIFAR 10: Runtime versus μ

(b) Guillermo: Runtime versus μ

(c) CIFAR 10: Residual versus μ

(d) Guillermo: Residual versus μ

FIG. 3. Ridge regression: Runtime and residual. *Nyström PCG either is the fastest method, or it is competitive with the fastest method, for all values of the regularization parameter μ. CG is generally the slowest method. All the methods reliably achieve the target residual along the entire regularization path, except for ordinary CG at small values of μ. See subsection* 6.2.2.

scheme in subsection 5.4.1 effectively selects a rank on the order of $d_{\text{eff}}(\mu)$ when the effective dimension is small or moderate.

Nyström PCG also works well for sketch sizes smaller than the effective dimension. For example, on YearMSD-rf, Nyström PCG converges quickly despite a rank three times smaller than $d_{\text{eff}}(\mu)$. For covtype-rf, where $d_{\text{eff}}(\mu) \sim d$, the convergence is no longer as fast, but it still outperforms R&T, owing to the expensive $O(d^3)$ cost of constructing the preconditioner. Thus, even in settings where $d_{\text{eff}}(\mu) \sim d$, Nyström PCG may still be faster than R&T when d is large enough. For a discussion on the statistical performance of Nyström PCG and the test set error obtained on all datasets, see Appendix B.1.

6.3. Approximate cross-validation. In this subsection we use our preconditioner to compute approximate leave-one-out cross-validation (ALOOCV), which requires solving a large linear system with multiple right-hand sides.

6.3.1. Background. Cross-validation is an important machine-learning technique to assess and select models and hyperparameters. Generally, it requires refitting a model on many subsets of the data, so it can take quite a long time. The worst culprit is leave-one-out cross-validation (LOOCV), which requires running an expensive training algorithm n times. Recent work has developed approximate leave-one-out cross-validation (ALOOCV), a faster alternative that replaces model retraining by a linear system solve [13, 33, 49]. In particular, these techniques yield accurate and computationally tractable approximations to LOOCV.

TABLE 3
Ridge regression: Nyström PCG versus AdaIHS and R&T PCG. *Nyström PCG outperforms AdaIHS and R&T PCG in iteration (#iters) and runtime for all datasets. Nyström PCG also requires much less storage (s_{final}). For Nyström PCG, the estimated condition number of the preconditioned system κ_{PCG} is computed using the upper bound in Proposition* 5.3.

Dataset	Method	$\mathbf{s}_{\text{final}}$	$\boldsymbol{\kappa}_{\text{PCG}}$	#iters	Runtime (s)
shuttle-rf	AdaIHS	10,000	-	66.9 (0.933)	66.9 (5.27)
	R&T PCG	20,000	-	60.15	242.6 (12.24)
	NysPCG	800	4.17 (0.161)	**13.1** (**1.47**)	**9.78** (**0.943**)
smallNORB-rf	AdaIHS	12,800	-	38.7 (1.42)	41.0 (2.46)
	R&T PCG	20,000	-	34.5 (1.31)	181.5 (6.53)
	NysPCG	800	18.5 (0.753)	**31.5** (**0.489**)	**6.67** (**0.372**)
YearMSD-rf	AdaIHS	30,000	-	44	1,327.3
	R&T PCG	30,000	-	49	766.5
	NysPCG	2,000	22.7	**22**	**209.7**
Higgs-rf	AdaIHS	6,400	-	55	1,052.7
	R&T PCG	20,000	-	53	607.4
	NysPCG	800	23.8	**28**	**91.26**
covtype-rf	AdaIHS	30,000	-	211	1,633.5
	R&T PCG	30,000	-	**50**	846.4
	NysPCG	2000	2.12e+4	430	**540.05**

TABLE 4
ALOOCV datasets and experimental parameters. *For each dataset we consider two values of μ; we also report the exact effective dimension*

Dataset	n	d	%nz(A)	μ	s_{init}	$\mathbf{d}_{\text{eff}}(\mu)$
Gisette	6,000	5,000	99.1%	1	850	116
				1e-4		948
real-sim	72,308	20,958	0.245%	1e-4	500	891
				1e-4		6,686
rcv1.binary	20,242	47,236	0.157%	1e-4	500	779
				1e-8		3,463
SVHN	73,257	3,072	100%	1	850	10
				1e-4		674

To present the approach, we consider the infinitesimal jackknife (IJ) approximation to LOOCV [13, 42]. The IJ approximation computes

$$\tilde{\theta}_{\text{IJ}}^{n/j} = \hat{\theta} + \frac{1}{n} H^{-1}(\hat{\theta}) \nabla_\theta l(\hat{\theta}, a_j), \tag{6.1}$$

where $H(\hat{\theta}) \in \mathbb{R}^{d \times d}$ is the Hessian of the loss function at the solution $\hat{\theta}$, for each datapoint a_j. The main computational challenge is computing the inverse Hessian vector product $H^{-1}(\hat{\theta}) \nabla_\theta l(\hat{\theta}, a_j)$. When n is very large, we can also subsample the data and average (6.1) over the subsample to estimate ALOOCV. Since ALOOCV solves the same problem with several right-hand sides, blocked PCG methods (here, Nyström blocked PCG) are the tool of choice to efficiently solve for multiple right-hand sides at once. To demonstrate the idea, we perform numerical experiments on ALOOCV for logistic regression. The datasets we use are all from LIBSVM [6]; see Table 4.

6.3.2. Experimental overview. We perform two sets of experiments in this section. The first set of experiments uses Gisette and SVHN to test the efficacy of Nyström sketch-and-solve. These datasets are small enough that we can factor $H(\theta)$ using a direct method. We also compare to block CG and block PCG with the computed Nyström approximation as a preconditioner. To assess the error due to an inexact solve for datapoint a_j, let $x_\star(a_j) = H^{-1}(\theta)\nabla_\theta l(\hat{\theta}, a_j)$. For any putative solution $\hat{x}(a_j)$, we compute the relative error $\|\hat{x}(a_j) - x_\star(a_j)\|_2 / \|x_\star(a_j)\|_2$. We average the relative error over 100 datapoints a_j.

The second set of experiments uses the larger datasets real-sim and rcv1.binary and small values of μ, the most challenging setting for ALOOCV. We restrict our comparison to block Nyström PCG versus the block CG algorithm, as Nyström sketch-and-solve is so inaccurate in this regime. We employ Algorithm C.1 to construct the preconditioner for block Nyström PCG.

6.3.3. Nyström sketch-and-solve. As predicted, Nyström sketch-and-solve works poorly (Table 5). When $\mu = 1$, the approximate solutions are modestly accurate, and the accuracy degrades as μ decreases to 10^{-4}. The experimental results agree with the theoretical analysis presented in Algorithm 4.1, which indicate that sketch-and-solve degrades as μ decreases. In contrast, block CG and block Nyström PCG both provide high-quality solutions for each datapoint for both values of the regularization parameter.

6.4. Large scale ALOOCV experiments. Table 6 summarizes results for block Nyström PCG and block CG on the larger datasets. When $\mu = 10^{-4}$, block Nyström PCG offers little or no benefit over block CG because the data matrices are very sparse (see Table 4) and the rcv1 problem is well-conditioned (see table SM2).

TABLE 5
ALOOCV: Small datasets. *The error for a given value of μ is the maximum relative error on* 100 *randomly sampled datapoints, averaged over* 20 *trials.*

Dataset	μ	Nyström sketch-and-solve	Block CG	Block Nyström PCG
Gisette	1	4.99e–2	2.68e–11	2.58e–12
Gisette	1e–4	1.22e–0	1.19e–11	6.59e–12
SVHN	1	9.12e–5	2.80e–13	1.26e–13
SVHN	1e–4	3.42e–1	2.01e–10	1.41e–11

TABLE 6
ALOOCV: Large datasets. *Block Nyström PCG outperforms block CG for small μ.*

Dataset	μ	Method	#iters	Runtime (s)
rcv1	1e–4	Block CG	12	**11.06** (**0.874**)
	1e–4	Block Nyström PCG	**10**	11.87 (0.767)
rcv1	1e–8	Block CG	52	39.03 (2.97)
	1e–8	Block Nyström PCG	**15**	**24.1** (**1.79**)
realsim	1e–4	Block CG	12	23.04 (2.04)
	1e–4	Block Nyström PCG	**8**	**19.05** (**1.10**)
realsim	1e–8	Block CG	90	163.7 (12.3)
	1e–8	Block Nyström PCG	**32**	**68.9** (**5.30**)

For $\mu = 10^{-8}$, block Nyström PCG reduces the number of iterations substantially, but the speedup is negligible. The data matrix A is sparse, which reduces the benefit of the Nyström method. Block CG also benefits from the presence of multiple right-hand sides just as block Nyström PCG. Indeed, O'Leary proved that the convergence of block CG depends on the ratio $(\lambda_s + \mu)/(\lambda_n + \mu)$, where s is is the number of right-hand sides [31]. Consequently, multiple right-hand sides precondition block CG and accelerate convergence. We expect bigger gains over block CG when A is dense.

6.5. Kernel ridge regression. Our last application is kernel ridge regression (KRR), a supervised learning technique that uses a kernel to model nonlinearity in the data. KRR leads to large dense linear systems that are challenging to solve.

6.5.1. Background. We briefly review KRR [40]. Given a dataset of inputs $x_i \in \mathcal{D}$, their corresponding outputs $b_i \in \mathbb{R}$ for $i = 1, \ldots, n$, and a kernel function $\mathcal{K}(x, y)$, KRR finds a function $f_\star : \mathcal{D} \to \mathbb{R}$ in the associated reproducing kernel Hilbert space $\mathcal{H}$ that best predicts the outputs for the given inputs. The solution $f_\star \in \mathcal{H}$ minimizes the square error subject to a complexity penalty:

$$f_\star = \operatorname*{argmin}_{f \in \mathcal{H}} \quad \frac{1}{2n} \sum_{i=1}^{n} (f(x_i) - b_i)^2 + \frac{\mu}{2} \|f\|_{\mathcal{H}}^2, \tag{6.2}$$

where $\|\cdot\|_{\mathcal{H}}$ denotes the norm on $\mathcal{H}$. Define the kernel matrix $K \in \mathbb{R}^{n \times n}$ with entries $K_{ij} = \mathcal{K}(x_i, x_j)$. The representer theorem [41] states the solution to (6.2) is

$$f_\star(x) = \sum_{i=1}^{n} \alpha_i \mathcal{K}(x, x_i),$$

where $\alpha = (\alpha_1, \ldots, \alpha_n)$ solves the linear system

$$(K + n\mu I)\alpha = b. \tag{6.3}$$

Solving the linear system (6.3) is the computational bottleneck of KRR. Direct factorization methods to solve (6.3) are prohibitive for large n as their costs grow as n^3; for $n > 10^4$ or so, iterative methods are generally preferred. However, K is often extremely ill-conditioned, even with the regularization term $n\mu I$. As a result, CG for Problem (6.3) converges slowly.

6.5.2. Experimental overview. We use Nyström PCG (NysPCG) to solve several KRR problems derived from classification problems on real-world datasets from [6, 46]. For all experiments, we use the Gaussian kernel $\mathcal{K}(x, y) = \exp(-\|x - y\|^2/(2\sigma^2))$. Following [2], we take a machine learning perspective: the objective is to minimize test set error rather than to achieve the smallest possible residual. We compare our method to random features PCG (rfPCG), proposed in [2]. We do not compare to vanilla CG as it is much slower than NysPCG and rfPCG. We also compare to Falkon [37], a state-of-the-art scalable approximation method for kernel ridge regression. For Falkon, we use the author's MATLAB implementation provided here: `https://github.com/LCSL/FALKON_paper`. This implementation is more optimized than the implementations of NysPCG and rfPCG, making use of C++ for several important steps. Thus, the comparison to rfPCG and NysPCG made here is very favorable to Falkon.

Either all datasets come with specified test sets, or we create one from a random 80–20 split. The PCG tolerance, σ, and μ were all chosen to achieve good performance

TABLE 7
Kernel ridge regression datasets and experimental parameters. *The table shows experimental parameters and estimates of the effective dimension and numerical rank (see Table 2) of each kernel in single precision. These estimates are computed using sketching methods as described in* [27].

Dataset	n	d	$\mathbf{n_{classes}}$	μ	σ	Tol	$d_{\text{eff}}(n\mu)$	NumRank
ijcnn1	49,990	49	2	1e-6	0.5	1e-3	5,269	$>$ 12,498
MNIST	60,000	784	10	1e-7	5	1e-4	$>$ 15,000	$>$ 15,000
Sensorless	48,509	48	11	1e-8	0.8	1e-4	1,948	2,331
SensIT	78,823	100	3	1e-8	3	1e-3	8,186	9,216
MiniBooNE	104,052	50	2	1e-7	5	1e-4	522	1,065
EMNIST	105,280	784	47	1e-6	8	1e-3	17,079	$>$ 26,320
Santander	160,000	200	2	1e-6	7	1e-3	$>$ 40,000	$>$ 40,000

on the test sets (see Table 8 below). In particular, the test set error on a given dataset saturates or increases if PCG (either rfPCG or NysPCG) is not stopped after reaching the selected tolerance. Both rfPCG and NysPCG were allowed to run for a maximum of 500 iterations. We report statistics for each dataset and experimental parameters in Table 7.

In addition, Table 7 also reports estimates of the effective dimension and the numerical rank for each kernel matrix. For these KRR systems, computing the exact effective dimension and numerical rank is too expensive, even in single precision. Instead, we use procedures described in [27] to estimate the effective dimension and numerical rank (in single precision) of the kernel matrix, and report only a lower bound on the effective dimension or numerical rank if the estimate exceeds $\lceil 0.25n \rceil$.

We run two sets of experiments. For the datasets with $n < 10^5$, the "oracle" method uses the a posteriori best parameters for rfPCG (the rank of random features approximation used to construct the preconditioner) and NysPCG (the sketch size ℓ), chosen by grid search, which we call Or-rfPCG and Or-NysPCG, respectively. We also compare to the adaptive NysPCG algorithm (Ada-NysPCG) described in subsection 5.4.2. We restrict values for ℓ and the rank of the random features approximation to be less than $10,000$ to ensure the preconditioners are cheap to apply and store. Ada-NysPCG for each dataset was initialized at $\ell = 2,000$, which is smaller than $0.05n$ for all datasets. For the datasets with $n \geq 10^5$, we restrict both ℓ and the rank of the random features approximation to $1,000$, which corresponds to less than $0.01n$. This fixed-rank setting allows us to see how both methods perform in the situation where the size of the preconditioner is restricted owing to memory constraints. We then run both algorithms until they reach the desired tolerance or the maximum number of iterations. Falkon's main hyperparameter is the number of centers, which is typically taken to be a small fraction of the training set. For our experiments, we selected the number of centers via grid search using the grid $\{\lceil 0.01n \rceil, \lceil 0.025n \rceil, \lceil 0.05n \rceil, \lceil 0.075n \rceil, \lceil 0.1n \rceil\}$. The number of iterations used for solving the Falkon linear system is fixed at 20, matching the setting used by the authors in `https://github.com/LCSL/FALKON_paper` for datasets satisfying $n \lesssim 10^6$.

We use column sampling to construct the Nyström preconditioner for all KRR problems. On these problems, random projection takes longer and yields similar performance (with somewhat lower variance).

6.5.3. Experimental results. Table 8 summarizes the results for the KRR experiments. Table 8 shows that both versions of Nyström PCG perform better than

TABLE 8

Kernel ridge regression: Ranks, iteration count, and total runtime. *We denote random features PCG and Nyström PCG by rfPCG and NysPCG, respectively. The prefixes Or and Ada stand for oracle and adaptive. NysPrec is a sketch-and-solve style method that applies the inverse of the Nyström preconditioner to the right-hand side. The total runtime for rfPCG and both variants of NysPCG also includes the time required to compute the kernel matrix. All variants of NysPCG uniformly outperform rfPCG, in both runtime and number of iterations. NysPCG also attains the best test error across all problem instances.*

Dataset	Method	Rank or #centers	#iters	Total Runtime (s)	Test error
	Or-rfPCG	3,000	63.8(2.66)	56.2(2.33)	
icjnn1	Ada-NysPCG	2,000	43.7(1.77)	49.9(1.47)	**1.25**%
	Or-NysPCG	3,000	**31.8**(**0.835**)	51.2(1.60)	
	NysPrec	3,000	-	6.09(0.151)	7.06%
	Falkon	4,999	-	4.60(0.122)	1.39%
	Or-rfPCG	9,000	314.5(2.88)	291.4(6.93)	
MNIST	Ada-NysPCG	6,000 (1,716)	78.5(17.65)	185.8(46.39)	**1.22**%
	Or-NysPCG	4,000	**77.9**(**2.08**)	129.4(2.08)	
	NysPrec	4,000	-	31.36(0.427)	34.57%
	Falkon	6,000	-	7.01(0.288)	1.98%
	Or-rfPCG	5,000	55.4(2.35)	56.5(3.96)	
Sensorless	Ada-NysPCG	2,000	22.0(0.510)	40.0(1.26)	**2.05**%
	Or-NysPCG	2,000	**21.7**(**0.571**)	39.3(1.63)	
	NysPrec	2,000	-	9.21(0.248)	3.91%
	Falkon	3,639	-	3.12(0.214)	2.16%
	Or-rfPCG	7,000	68.0(4.31)	146.0(6.19)	
SensIT	Ada-NysPCG	2,000	**47.8**(**1.72**)	120.9(2.43)	**12.83**%
	Or-NysPCG	2,000	48.7(3.40)	112.4(6.41)	
	NysPrec	2,000	-	14.10(0.321)	22.55%
	Falkon	7,883	-	17.55(0.494)	13.05%
	rfPCG	1,000	92	240.8	
MiniBooNE	NysPCG	1,000	**72**	224.0	**7.93**%
	NysPrec	1,000	-	9.06	8.90%
	Falkon	10,046	-	43.03	7.96%
	rfPCG	1,000	154	753.1	
EMNIST	NysPCG	1,000	**32**	386.3	**15**%
	NysPrec	1,000	-	9.21	26.90%
	Falkon	10,528	-	24.5	17.57%
	rfPCG	1,000	160	1019.7	
Santander	NysPCG	1,000	**31**	374.1	**8.90**%
	NysPrec	1,000	-	13.66	19.24%
	Falkon	16,000	-	43.06	9.26%

random features preconditioning on all the datasets considered. Nyström PCG also uses less storage. In the fixed-rank setting with the larger scale datasets, Nyström PCG performs better than random features PCG. The second column in Table 8 shows that the adaptive strategy proposed in subsection 5.4.2 to select the sketch size ℓ works very well. In contrast, it is difficult to choose the rank for random features preconditioning: the authors of [2] provide no guidance except for the polynomial kernel. Moreover, the success of Nyström PCG is robust to reaching the effective dimension. Indeed, on MNIST, EMNIST, and Santander, Table 7 shows ℓ is much

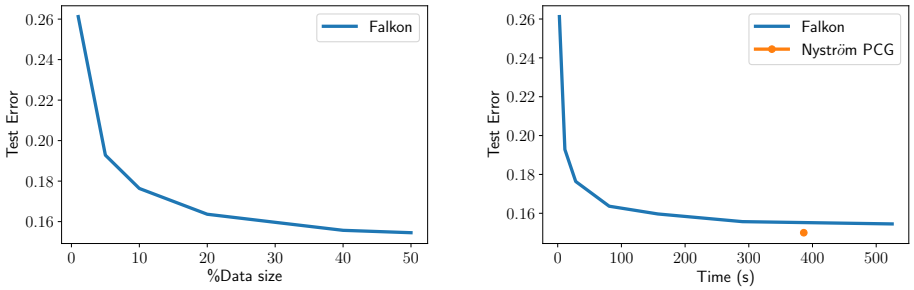


FIG. 4. Falkon saturates: *Falkon's performance saturates as the number of centers increases from* $0.01n$ *to* $0.5n$*, and always underperforms Nyström PCG. Furthermore, once the number of centers reaches* $0.5n$*, Falkon runs slower than Nyström PCG.*

smaller than $d_{\text{eff}}(\mu)$, yet Nyström PCG still converges quickly using the constructed preconditioner. This robustness is important from the viewpoint of practice, for as Table 7 reveals, the effective dimension $d_{\text{eff}}(\mu)$ is often large.

Table 8 shows that good test set error is obtained on all datasets. Significantly, Nyström PCG yields lower test set error than approximate methods such as Falkon and a sketch-and-solve style method that simply applies the inverse of the Nyström preconditioner to the right-hand side (NysPrec). However, Falkon and NysPrec run considerably faster as they work with only a subsample of the kernel matrix. We also see that Falkon generally outperforms NysPrec. The gap between Nyström PCG and Falkon can be quite large, such as with EMNIST where Nyström PCG obtains an error of 15.00% compared to the 17.57% obtained by Falkon. Furthermore, we found that this gap persisted even as we varied the number of centers from $0.01n$ to $0.5n$, at which point Falkon becomes more expensive than Nyström PCG; see Figure 4. Our observation that exact methods outperform approximate methods is consistent with findings in [2], which noted a similar performance gap between random features PCG and the basic random features method of [34]. Thus, even in the statistical learning setting, solving the problem more accurately using the full data does yield improved performance.

7. Conclusion. We have shown that Nyström PCG delivers a strong benefit over standard CG both in the theory and in practice, thanks to the ease of parameter selection, on a range of interesting large-scale computational problems including ridge regression, kernel ridge regression, and ALOOCV. In our experience, Nyström PCG outperforms all generic methods for solving large-scale dense linear systems with spectral decay. It is our hope that this paper motivates further research on randomized preconditioning for solving large-scale linear systems and offers a useful speedup to practitioners.

Appendix A. Proofs of main results. This appendix contains full proofs of the main results that are substantially novel (Theorem 4.2, Proposition 5.3, and Corollary 5.2). The supplement contains proofs of results that are similar to existing work but do not appear explicitly in the literature.

A.1. Proof Theorem 4.2. This result contains the analysis of the Nyström sketch-and-solve method. We begin with (4.2), which provides an error bound that compares the regularized inverse of a psd matrix A with the regularized inverse of the randomized Nyström approximation $\hat{A}_{\text{nys}}$. Since $0 \preceq \hat{A}_{\text{nys}} \preceq A$, we can apply Proposition 3.1 to obtain a deterministic bound for the discrepancy:

$$\|(\hat{A}_{\text{nys}} + \mu I)^{-1} - (A + \mu I)^{-1}\| \leq \frac{1}{\mu} \frac{\|E\|}{\|E\| + \mu}, \quad \text{where } E = A - \hat{A}_{\text{nys}}.$$

The function $f(t) = t/(t + \mu)$ is concave, so we can take expectations and invoke Jensen's inequality to obtain

$$\mathbb{E}\|(\hat{A}_{\text{nys}} + \mu I)^{-1} - (A + \mu I)^{-1}\| \leq \frac{1}{\mu} \frac{\mathbb{E}\|E\|}{\mathbb{E}\|E\| + \mu}.$$

Inserting the bound (2.3) on $\mathbb{E}\|E\|$ from Corollary 2.3 gives

$$\mathbb{E}\|(\hat{A}_{\text{nys}} + \mu I)^{-1} - (A + \mu I)^{-1}\| \leq \frac{1}{\mu} \cdot \frac{(3 + 4\mathrm{e}^2 \mathrm{sr}_p(A)/p)\lambda_p}{\mu + (3 + 4\mathrm{e}^2 \mathrm{sr}_p(A)/p)\lambda_p}.$$

To conclude, observe that the denominator of the second fraction exceeds $\mu + \lambda_p$.

Now, let us establish (4.3), the error bound for Nyström sketch-and-solve. Introduce the solution $\hat{x}$ to the Nyström sketch-and-solve problem and the solution $x_\star$ to the regularized linear system:

$$(\hat{A}_{\text{nys}} + \mu I)\hat{x} = b \quad \text{and} \quad (A + \mu I)x_\star = b.$$

We may decompose the regularized matrix as $A + \mu I = \hat{A}_{\text{nys}} + \mu I + E$. Subtract the two equations in the last display to obtain

$$(\hat{A}_{\text{nys}} + \mu I)(\hat{x} - x_\star) - Ex_\star = 0.$$

Rearranging to isolate the error in the solution, we have

$$\hat{x} - x_\star = (\hat{A}_{\text{nys}} + \mu I)^{-1} E x_\star.$$

Take the norm, apply the operator norm inequality, and use the elementary bound $\|(\hat{A}_{\text{nys}} + \mu I)^{-1}\| \leq \mu^{-1}$. We obtain

$$\frac{\|\hat{x} - x_\star\|_2}{\|x_\star\|_2} \leq \frac{\|E\|}{\mu}.$$

Finally, take the expectation and repeat the argument used to control $\mathbb{E}\|E\|/\mu$ in the proof of Theorem 5.1.

A.1.1. Proof of Proposition 5.3. Let $\hat{A} = U\hat{\Lambda}U^T$ be a rank-ℓ Nyström approximation constructed from an arbitrary test matrix, whose ℓth eigenvalue is $\hat{\lambda}_\ell$. Proposition 5.3 provides a deterministic bound on the condition number of the regularized matrix A_μ after preconditioning with

$$P = \frac{1}{\hat{\lambda}_\ell + \mu} U(\hat{\Lambda} + \mu I)U^T + (I - UU^T).$$

We remind the reader that this argument is completely deterministic.

First, note that the preconditioned matrix $P^{-1/2}A_\mu P^{-1/2}$ is psd, so

$$\kappa_2(P^{-1/2}A_\mu P^{-1/2}) = \frac{\lambda_1(P^{-1/2}A_\mu P^{-1/2})}{\lambda_n(P^{-1/2}A_\mu P^{-1/2})}.$$

Let us begin with the upper bound on the condition number. We have the decomposition

$$P^{-1/2}A_\mu P^{-1/2} = P^{-1/2}(\hat{A} + \mu I)P^{-1/2} + P^{-1/2}EP^{-1/2}, \tag{A.1}$$

owing to the relation $A_\mu = \hat{A} + \mu I + E$. Recall that the error matrix E is psd, so the matrix $P^{-1/2}EP^{-1/2}$ is also psd.

First, we bound the maximum eigenvalue. Weyl's inequalities imply that

$$\lambda_1(P^{-1/2}A_\mu P^{-1/2}) \le \lambda_1(P^{-1/2}(\hat{A}+\mu I)P^{-1/2}) + \lambda_1(P^{-1/2}EP^{-1/2}).$$

To determine $\lambda_1(P^{-1/2}(\hat{A}+\mu I)P^{-1/2})+$, we write $\hat{A}+\mu I = U(\hat{\Lambda}+\mu I)U^T + \mu U_\perp U_\perp^T$, where $U_\perp$ is an orthonormal basis for the eigenvectors orthogonal to U. From this and the definition of P^{-1}, we obtain

$$P^{-1/2}(\hat{A}+\mu I)P^{-1/2} = (\hat{\lambda}_\ell + \mu)UU^T + \mu U_\perp U_\perp^T.$$

The preceding display immediately yields $\lambda_1(P^{-1/2}(\hat{A}+\mu I)P^{-1/2}) = \hat{\lambda}_\ell + \mu$. We now turn to bounding $\lambda_1(P^{-1/2}EP^{-1/2})$. When $\ell < n$, we have $\lambda_1(P^{-1}) = 1$. Therefore,

$$\lambda_1(P^{-1/2}EP^{-1/2}) = \lambda_1(P^{-1}E) \le \lambda_1(P^{-1})\lambda_1(E) = \lambda_1(E) = \|E\|.$$

In summary,

$$\lambda_1(P^{-1/2}A_\mu P^{-1/2}) \le \hat{\lambda}_\ell + \mu + \|E\|. \tag{A.2}$$

For the minimum eigenvalue, we first assume that $\mu > 0$. Apply Weyl's inequality to (A.1) to obtain to obtain

$$\begin{aligned} \lambda_n(P^{-1/2}A_\mu P^{-1/2}) &\ge \lambda_n(P^{-1/2}(\hat{A}+\mu I)P^{-1/2}) + \lambda_n(P^{-1/2}EP^{-1/2}) \\ &\ge \lambda_n(P^{-1/2}(\hat{A}+\mu I)P^{-1/2}) = \mu. \end{aligned} \tag{A.3}$$

Combining (A.2) and (A.3), we reach

$$\kappa_2(P^{-1/2}A_\mu P^{-1/2}) \le \frac{\hat{\lambda}_\ell + \mu + \|E\|}{\mu}.$$

This gives a bound for the maximum in case $\mu > 0$.

If we only have $\mu \ge 0$, then a different argument is required for the smallest eigenvalue. Assume that A is positive definite, in which case $\hat{\lambda}_\ell > 0$. As $P^{-1/2}A_\mu P^{-1/2}$ is symmetric positive definite we have

$$\lambda_n(P^{-1/2}A_\mu P^{-1/2}) = \frac{1}{\lambda_1(P^{1/2}A_\mu^{-1}P^{1/2})}.$$

Conjugating by $A_\mu^{1/2}P^{-1/2}$ and using similarity, we obtain the equality

$$\lambda_1(P^{1/2}A_\mu^{-1}P^{1/2}) = \lambda_1(A_\mu^{-1/2}PA_\mu^{-1/2}).$$

Hence it suffices to produce an upper bound for $\lambda_1(A_\mu^{-1/2}PA_\mu^{-1/2})$. To that end, we expand

$$\begin{aligned} \lambda_1(A_\mu^{-1/2}PA_\mu^{-1/2}) &= \lambda_1\left(A_\mu^{-1/2}\left(\frac{1}{\hat{\lambda}_\ell+\mu}(\hat{A}+\mu UU^T) + (I-UU)^T\right)A_\mu^{-1/2}\right) \\ &\le \frac{1}{\hat{\lambda}_\ell+\mu}\lambda_1\left(A_\mu^{-1/2}(\hat{A}+\mu UU^T)A_\mu^{-1/2}\right) \\ &\quad + \lambda_1\left(A_\mu^{-1/2}\left(I-UU^T\right)A_\mu^{-1/2}\right). \end{aligned}$$

The second inequality is Weyl's. Since $\hat{A} \preceq A$, we have $\hat{A} + \mu UU^T \preceq A_\mu$. The last display simplifies to

$$\lambda_1(A_\mu^{-1/2} P A_\mu^{-1/2}) \leq \frac{1}{\hat{\lambda}_\ell + \mu} + \frac{1}{\lambda_n + \mu}.$$

Putting the pieces together with (A.2), we obtain

$$\kappa_2(P^{-1/2} A_\mu P^{-1/2}) \leq (\hat{\lambda}_\ell + \mu + \|E\|)\left(\frac{1}{\hat{\lambda}_\ell + \mu} + \frac{1}{\lambda_n + \mu}\right).$$

Thus,

$$\kappa_2(P^{-1/2} A_\mu P^{-1/2}) \leq \left(\hat{\lambda}_\ell + \mu + \|E\|\right) \min\left\{\frac{1}{\mu},\ \frac{\hat{\lambda}_\ell + \lambda_n + 2\mu}{(\hat{\lambda}_\ell + \mu)(\lambda_n + \mu)}\right\}.$$

This formula is valid when A is positive definite or when $\mu > 0$.

We now prove the lower bound on $\kappa_2(P^{-1/2} A_\mu P^{-1/2})$. Returning to (A.1) and invoking Weyl's inequalities yields

$$\lambda_1(P^{-1/2} A_\mu P^{-1/2}) \geq \lambda_1(P^{-1/2}(\hat{A} + \mu I) P^{-1/2}) + \lambda_n(P^{-1/2} E P^{-1/2}) \geq \hat{\lambda}_\ell + \mu.$$

For the smallest eigenvalue we observe that

$$\lambda_n(P^{-1/2} A_\mu P^{-1/2}) = \lambda_n(A_\mu P^{-1}) \leq \lambda_n(A_\mu)\lambda_1(P^{-1}) = \lambda_n + \mu,$$

where the last inequality in the preceding display follows from the identity

$$\lambda_j(AB) \leq \lambda_j(A)\lambda_1(B),$$

which holds for symmetric positive definite matrices A and B. Combining the last two displays, we obtain

$$\frac{\hat{\lambda}_\ell + \mu}{\lambda_n + \mu} \leq \kappa_2(P^{-1/2} A_\mu P^{-1/2}).$$

Condition numbers always exceed one, so

$$\max\left\{\frac{\hat{\lambda}_\ell + \mu}{\lambda_n + \mu}, 1\right\} \leq \kappa_2(P^{-1/2} A_\mu P^{-1/2}).$$

This point concludes the argument.

A.1.2. Proof of Lemma 5.4. Lemma 5.4 establishes the central facts about the effective dimension. First, we prove item 1. Fix a parameter $\gamma \geq 1$, and set $j_\star = \max\{1 \leq j \leq n : \lambda_j > \gamma\mu\}$. We can bound the effective dimension below by the following mechanism:

$$d_{\text{eff}}(\mu) = \sum_{j=1}^{n} \frac{\lambda_j}{\lambda_j + \mu} \geq \sum_{j=1}^{j_\star} \frac{\lambda_j}{\lambda_j + \mu} \geq j_\star \cdot \frac{\lambda_{j_\star}}{\lambda_{j_\star} + \mu}.$$

We have used the fact that $t \mapsto t/(1+t)$ is increasing for $t \geq 0$, Solving for $j_\star$, we determine that

$$j_\star \leq (1 + \mu/\lambda_{j_\star}) d_{\text{eff}}(\mu) < (1 + \gamma^{-1}) d_{\text{eff}}(\mu).$$

The last inequality depends on the definition of $j_\star$. This is the required result.

Item 2 follows from a short calculation:

$$\begin{aligned}\frac{1}{k}\sum_{j>k}\lambda_j &= \frac{\lambda_k+\mu}{k}\sum_{j>k}\frac{\lambda_j}{\lambda_k+\mu} \le \frac{\lambda_k+\mu}{k}\sum_{j>k}\frac{\lambda_j}{\lambda_j+\mu} \\ &= \frac{\lambda_k+\mu}{k}\left(d_{\mathrm{eff}}(\mu) - \sum_{j=1}^{k}\frac{\lambda_j}{\lambda_j+\mu}\right) \le \frac{\lambda_k+\mu}{k}\left(d_{\mathrm{eff}}(\mu) - \frac{k\lambda_k}{\lambda_k+\mu}\right) \\ &= \frac{\mu\, d_{\mathrm{eff}}(\mu)}{k} + \lambda_k\left(\frac{d_{\mathrm{eff}}(\mu)}{k} - 1\right) \le \frac{\mu\, d_{\mathrm{eff}}(\mu)}{k}.\end{aligned}$$

The last inequality depends on the assumption that $k \ge d_{\mathrm{eff}}(\mu)$.

A.2. Proof of Corollary 5.2. This result gives a bound for the relative error δ_t in the iterates of PCG. Recall the standard convergence bound for CG [44, Theorem 38.5]:

$$\delta_t \le 2\left(\frac{\sqrt{\kappa_2(P^{-1/2}A_\mu P^{-1/2})}-1}{\sqrt{\kappa_2(P^{-1/2}A_\mu P^{-1/2})}+1}\right)^t.$$

We conditioned on the event that $\{\kappa(P^{-1/2}A_\mu P^{-1/2}) \le 56\}$. On this event, the relative error must satisfy

$$\delta_t < 2\left(\frac{\sqrt{56}-1}{\sqrt{56}+1}\right)^t \le 2\cdot(0.77)^t.$$

Solving for t, we see that $\delta_t < \epsilon$ when $t \ge \lceil 3.8\log(1/\epsilon)\rceil$. This concludes the proof.

A.3. Proof of Theorem 5.5. Theorem 5.5 establishes that with high probability Algorithm C.1 terminates in a logarithmic number of steps, the sketch size remains $O(d_{\mathrm{eff}}(\delta\tau\mu))$, and PCG with the preconditioner constructed from the output converges quickly.

Proof. We first recall with the tolerances chosen in Theorem 5.5 that Algorithm C.1 terminates when the event

$$\mathcal{E} = \{\|E\| \le \tau\mu\} \cap \left\{\hat{\lambda}_\ell \le \frac{\tau\mu}{11}\right\}$$

holds. Observe that conditioned on $\mathcal{E}$, Proposition 5.3 yields

$$\kappa_2(P^{-1/2}A_\mu P^{-1/2}) \le \frac{\hat{\lambda}_\ell + \mu + \|E\|}{\mu} \le 1 + \left(1+\frac{1}{11}\right)\tau = 1 + \frac{12}{11}\tau.$$

Statement 3 now follows from the above display and the standard convergence theorem for CG.

Now, if Algorithm C.1 terminates with $N \le \lceil \log_2(\tilde{\ell}/\ell_0)\rceil - 1$ steps of sketch size doubling, then $\mathcal{E}$ holds with probability 1. Statement 3 then follows by our initial observation, while statements 1 and 2 hold trivially. Hence statements 1–3 all hold if the algorithm terminates in $N \le \lceil \log_2(\tilde{\ell}/\ell_0)\rceil - 1$ steps.

Thus to conclude the proof, it suffices to show that if $N \geq \lceil \log_2(\tilde{\ell}/\ell_0) \rceil$, then $\mathcal{E}$ holds with probability at least $1-\delta$, which implies that statements 1–3 hold with probability at least $1-\delta$, as above.

We now show that $\mathcal{E}$ holds with probability at least $1-\delta$ when $N = \lceil \log_2(\tilde{\ell}/\ell_0) \rceil$. To see this note that when $N = \lceil \log_2(\tilde{\ell}/\ell_0) \rceil$, we have $\ell \geq \tilde{\ell}$. Consequently, we may invoke Proposition 2.2 with $p = \lceil 2d_{\text{eff}}(\frac{\delta\tau\mu}{11}) \rceil + 1$ and Lemma 5.4 to show

$$\begin{aligned}
\mathbb{E}[\|E\|] &\overset{(1)}{\leq} 3\lambda_p + \frac{2\mathrm{e}^2}{p}\left(\sum_{j=p}^{n} \lambda_j\right) \\
&\overset{(2)}{\leq} 3\frac{\delta\tau\mu}{11} + 2\mathrm{e}^2 \frac{d_{\text{eff}}(\delta\tau\mu/11)}{p}\frac{\delta\tau\mu}{11} \\
&\overset{(3)}{\leq} \frac{3}{11}\delta\tau\mu + \frac{2\mathrm{e}^2}{2}\frac{\delta\tau\mu}{11} = \left(\frac{3+\mathrm{e}^2}{11}\right)\delta\tau\mu \leq \delta\tau\mu,
\end{aligned}$$

where step (1) uses Proposition 2.2, step (2) uses items 1 and 2 of Lemma 5.4 with $\gamma = 1$, and step (3) follows from $p \geq 2d_{\text{eff}}(\delta\tau\mu)$. Thus,

$$\mathbb{E}[\|E\|] \leq \delta\tau\mu.$$

By Markov's inequality,

$$\mathbb{P}\{\|E\| > \tau\mu\} \leq \delta.$$

Hence $\{\|E\| \leq \tau\mu\}$ holds with probability at least $1-\delta$. Furthermore, by Lemma 5.4 we have $\{\hat{\lambda}_\ell \leq \delta\tau\mu/11\}$ with probability 1 as $\hat{\lambda}_\ell \leq \lambda_\ell \leq \lambda_p$. Thus when $N = \lceil \log_2(\tilde{\ell}/\ell_0) \rceil$, $\mathcal{E}$ holds with probability at least $1-\delta$, and this immediately implies statements 1 and 3. Statement 2 follows as

$$\ell = 2^N \ell_0 \leq 2^{\log_2(\tilde{\ell}/\ell_0)+1}\ell_0 = 2\tilde{\ell} = 4\left\lceil 2d_{\text{eff}}\left(\frac{\delta\tau\mu}{11}\right)\right\rceil + 2,$$

where in the first inequality we used $\lceil x \rceil \leq x+1$; this completes the proof. ∎

A.4. Proof of Proposition 5.7. Proposition 5.7 shows once $\ell = \Omega(d_{\text{eff}}(\tau\mu))$, then with high probability $\kappa_2(P^{-1/2}A_\mu P^{-1/2})$ differs from $(\hat{\lambda}_\ell + \mu)$ by at most a constant.

Proof. Proposition 5.3 implies that

$$\left(\kappa_2(P^{-1/2}A_\mu P^{-1/2}) - \frac{\hat{\lambda}_\ell + \mu}{\mu}\right)_+ \leq \frac{\|E\|}{\mu}.$$

Combining the previous display with Markov's inequality yields

$$\mathbb{P}\left\{\left(\kappa_2(P^{-1/2}A_\mu P^{-1/2}) - \frac{\hat{\lambda}_\ell + \mu}{\mu}\right)_+ > \frac{\tau}{\delta}\right\} \leq \frac{\delta}{\tau}\frac{\mathbb{E}[\|E\|]}{\mu}.$$

TABLE 9

Ridge regression: Test set error. *We report the relative error for regression tasks and the misclassification rate for classification tasks. Nyström PCG outperforms the Nyström preconditioner on nearly all the datasets.*

Dataset	Method	Test set error
CIFAR-10	Nyström Preconditioner	9.51%
	Nyström PCG	**8.67**%
Guillermo	Nyström Preconditioner	**32.5**%
	Nyström PCG	32.6%
shuttle-rf	Nyström Preconditioner	**0.20**%
	Nyström PCG	0.22%
smallnorb-rf	Nyström Preconditioner	57.92%
	Nyström PCG	**16.14**%
YearMSD-rf	Nyström Preconditioner	5.48e–3
	Nyström PCG	**4.55e–3**
Higgs-rf	Nyström Preconditioner	3.49%
	Nyström PCG	**0.05**%
Covtype-rf	Nyström Preconditioner	20.76%
	Nyström PCG	**9.39**%

Now, our choice of ℓ combined with Proposition 2.2 and Lemma 5.4 implies that $\mathbb{E}[\|E\|] \leq \tau\mu$. Hence we have

$$\mathbb{P}\left\{\left(\kappa_2(P^{-1/2}A_\mu P^{-1/2}) - \frac{\hat{\lambda}_\ell + \mu}{\mu}\right)_+ > \frac{\tau}{\delta}\right\} \leq \delta,$$

which implies the desired claim. ∎

Appendix B. Additional numerical results.

B.1. Ridge regression experiments. Here, we report the test error obtained on datasets considered in subsection 6.2 and the implications of these results.

Table 9 compares the test error obtained using the Nyström PCG solution with that of a sketch-and-solve approach we call *Nyström preconditioner*, which uses P^{-1} (the inverse of the Nyström preconditioner) to approximate $(1/nA^TA + \mu I)^{-1}$.

Nyström PCG outperforms, especially on the larger datasets $n \geq 10^5$, and even for datasets where the effective dimension is small, such as Higgs-rf. Hence even in the statistical learning setting, where one only cares about test error, solving the ridge regression problem accurately improves statistical performance.

Appendix C. Adaptive rank selection via a posteriori error estimation.

The pseudocode for adaptive rank selection by a priori error estimation is given in Algorithm C.1. The code is structured to reuse use the previously computed Ω and Y, resulting in significant computational savings. The error $\|E\|$ is estimated from q iterations of the randomized power method on the error matrix $A - U\hat{\Lambda}U^T$.

Algorithm C.1 Adaptive Randomized Nyström Approximation

Input: symmetric psd matrix A, initial rank ℓ_0, maximum rank $\ell_{\max}$, number of power iterations for estimating E, error tolerance $\mathrm{Tol}_{\mathrm{Err}}$, ratio tolerance $\mathrm{Tol}_{\mathrm{Rat}}$

$Y = [\ ], \Omega = [\ ]$, $E = \mathrm{Inf}$, and $\hat{\lambda}_\ell = \mathrm{Inf}$
$m = \ell_0$
while $E > \mathrm{Tol}_{\mathrm{Err}}$ and $\hat{\lambda}_\ell / \mu > \mathrm{Tol}_{\mathrm{Rat}}$ **do**
  Generate Gaussian test matrix $\Omega_0 \in \mathbb{R}^{n \times m}$
  $[\Omega_0, \sim] = \mathrm{qr}(\Omega_0, 0)$
  $Y_0 = A\Omega_0$
  $\Omega = [\Omega\ \ \Omega_0]$ and $Y = [Y\ \ Y_0]$
  $\nu = \sqrt{n}\,\mathrm{eps}(\mathrm{norm}(Y, 2))$
  $Y_\nu = Y + \nu\Omega$,
  $C = \mathrm{chol}(\Omega^T Y_\nu)$
  $B = Y_\nu / C$
  Compute $[U, \Sigma, \sim] = \mathrm{svd}(B, 0)$
  $\hat{\Lambda} = \max\{0, \Sigma^2 - \nu I\}$ ▷ remove shift
  $E = \mathrm{RandomizedPowerErrEst}(A, U, \hat{\Lambda}, q)$ ▷ estimate error
  $m \leftarrow \ell_0$, $\ell_0 \leftarrow 2\ell_0$ ▷ double rank if tolerances are not met
  If $\ell_0 > \ell_{\max}$ **then**
    $\ell_0 = \ell_0 - m$ ▷ when $\ell_0 > \ell_{\max}$, reset to $\ell_0 = \ell_{\max}$
    $m = \ell_{\max} - \ell_0$
    Generate Gaussian test matrix $\Omega_0 \in \mathbb{R}^{n \times m}$
    $[\Omega_0, \sim] = \mathrm{qr}(\Omega_0, 0)$
    $Y_0 = A\Omega_0$
    $\Omega = [\Omega\ \ \Omega_0]$ and $Y = [Y\ \ Y_0]$
    $\nu = \sqrt{n}\,\mathrm{eps}(\mathrm{norm}(Y, 2))$ ▷ compute final approximation and break
    $Y_\nu = Y + \nu\Omega$,
    $C = \mathrm{chol}(\Omega^T Y_\nu)$
    $B = Y_\nu / C$
    Compute $[U, \Sigma, \sim] = \mathrm{svd}(B, 0)$
    $\hat{\Lambda} = \max\{0, \Sigma^2 - \nu I\}$
    break

Output: Nyström approximation $(U, \hat{\Lambda})$

Acknowledgments. The authors would like to thank the associate editor Julianne Chung and the two anonymous reviewers for helping to greatly improve the quality of the manuscript with their suggestions and comments. We also thank Alex Townsend and David Bindel for helpful discussions and suggestions.

REFERENCES

[1] A. Alaoui and M. W. Mahoney, *Fast randomized kernel ridge regression with statistical guarantees*, in NIPS, Vol. 28, 2015, pp. 775–783.
[2] H. Avron, K. L. Clarkson, and D. P. Woodruff, *Faster kernel ridge regression using sketching and preconditioning*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 1116–1138, https://doi.org/10.1137/16M1105396.
[3] H. Avron, P. Maymounkov, and S. Toledo, *Blendenpik: Supercharging LAPACK's least-squares solver*, SIAM J. Sci. Comput., 32 (2010), pp. 1217–1236, https://doi.org/10.1137/090767911.

[4] F. Bach, *Sharp analysis of low-rank kernel matrix approximations*, in COLT, 2013, pp. 185–209.
[5] R. Bhatia, *Matrix Analysis*, Grad. Texts in Math. 169, Springer, Berlin, 1997.
[6] C.-C. Chang and C.-J. Lin, *LIBSVM: A library for support vector machines*, ACM Trans. Intelligent Syst. Technol., 2 (2011), 27.
[7] A. Chowdhury, J. Yang, and P. Drineas, *Randomized iterative algorithms for Fisher discriminant analysis*, in Uncertainty in Artificial Intelligence, PMLR, 2020, pp. 239–249.
[8] M. B. Cohen, J. Nelson, and D. P. Woodruff, *Optimal approximate matrix product in terms of stable rank*, in International Colloquium on Automata, Languages, and Programming, 2016, 11.
[9] M. Derezinski, R. Khanna, and M. W. Mahoney, *Improved guarantees and a multiple-descent curve for column subset selection and the Nyström method*, in NeurIPS, Vol. 33, 2020, pp. 4953–4964.
[10] P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff, *Fast approximation of matrix coherence and statistical leverage*, J. Mach. Learn. Res., 13 (2012), pp. 3475–3506.
[11] Y. Feng, D. Owen, and D. Perić, *A block conjugate gradient method applied to linear systems with multiple right-hand sides*, Comput. Methods Appl. Mech. Engrg., 127 (1995), pp. 203–215.
[12] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson, *GPyTorch: Blackbox matrix-matrix Gaussian process inference with GPU acceleration*, in NeurIPS 31, 2018.
[13] R. Giordano, W. T. Stephenson, R. Liu, M. I. Jordan, and T. Broderick, *A Swiss Army infinitesimal jackknife*, in AISTATS, PMLR, 2019, pp. 1139–1147.
[14] L. Giraud and S. Gratton, *On the sensitivity of some spectral preconditioners*, SIAM J. Matrix Anal. Appl., 27 (2006), pp. 1089–1105, https://doi.org/10.1137/040617546.
[15] A. Gittens, *The Spectral Norm Error of the Naive Nyström Extension*, preprint, arXiv:1110.5305, 2011.
[16] A. Gittens and M. W. Mahoney, *Revisiting the Nyström method for improved large-scale machine learning*, J. Mach. Learn. Res., 17 (2016), pp. 3977–4041.
[17] G. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 2013.
[18] A. Gonen, F. Orabona, and S. Shalev-Shwartz, *Solving ridge regression using sketched preconditioned SVRG*, in ICML, PMLR, 2016, pp. 1397–1405.
[19] R. Johnson and T. Zhang, *Accelerating stochastic gradient descent using predictive variance reduction*, in NIPS 26, 2013, pp. 19377–19387.
[20] P. W. Koh and P. Liang, *Understanding black-box predictions via influence functions*, in ICML, PMLR, 2017, pp. 1885–1894.
[21] J. Kuczyński and H. Woźniakowski, *Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1094–1122, https://doi.org/10.1137/0613066.
[22] J. Lacotte and M. Pilanci, *Effective dimension adaptive sketching methods for faster regularized least-squares optimization*, in NeurIPS, Vol. 33, 2020.
[23] H. Li, G. C. Linderman, A. Szlam, K. P. Stanton, Y. Kluger, and M. Tygert, *Algorithm 971: An implementation of a randomized algorithm for principal component analysis*, ACM Trans. Math. Softw., 43 (2017), pp. 1–14.
[24] J. Lorraine, P. Vicol, and D. Duvenaud, *Optimizing millions of hyperparameters by implicit differentiation*, in AISTATS, PMLR, 2020, pp. 1540–1552.
[25] P.-G. Martinsson and J. A. Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numer., 29 (2020), pp. 403–572.
[26] G. Meanti, L. Carratino, L. Rosasco, and A. Rudi, *Kernel methods through the roof: Handling billions of points efficiently*, in NeurIPS 33, 2020, pp. 14410–14422.
[27] M. Meier and Y. Nakatsukasa, *Fast Randomized Numerical Rank Estimation*, preprint, arXiv:2105.07388, 2021.
[28] X. Meng, M. A. Saunders, and M. W. Mahoney, *LSRN: A parallel iterative solver for strongly over- or underdetermined systems*, SIAM J. Sci. Comput., 36 (2014), pp. C95–C118, https://doi.org/10.1137/120866580.
[29] C. Musco and C. Musco, *Randomized block Krylov methods for stronger and faster approximate singular value decomposition*, in NIPS 28, 2015.
[30] Y. Nakatsukasa, *Fast and Stable Randomized Low-Rank Matrix Approximation*, preprint, arXiv:2009.11392, 2020.
[31] D. P. O'Leary, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl., 29 (1980), pp. 293–322.

[32] C. C. Paige and M. A. Saunders, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Softw., 8 (1982), pp. 43–71.
[33] K. R. Rad and A. Maleki, *A scalable estimate of the out-of-sample prediction error via approximate leave-one-out cross-validation*, J. R. Stat. Soc. Ser. B Stat. Methodol., 82 (2020), pp. 965–996.
[34] A. Rahimi and B. Recht, *Random features for large-scale kernel machines*, in NIPS, Vol. 20, 2007, pp. 1177–1184.
[35] A. Rahimi and B. Recht, *Uniform approximation of functions with random bases*, in Allerton Conference on Communication, Control, and Computing, IEEE, 2008, pp. 555–561.
[36] V. Rokhlin and M. Tygert, *A fast randomized algorithm for overdetermined linear least-squares regression*, Proc. Natl. Acad. Sci. USA, 105 (2008), pp. 13212–13217.
[37] A. Rudi, L. Carratino, and L. Rosasco, *Falkon: An optimal large scale kernel method*, in NIPS 30, 2017.
[38] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, 2003, https://doi.org/10.1137/1.9780898718003.
[39] T. Sarlos, *Improved approximation algorithms for large matrices via random projections*, in IEEE Symposium on Foundations of Computer Science (FOCS), IEEE, 2006, pp. 143–152.
[40] B. Schölkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT Press, Cambridge, MA, 2002.
[41] I. Steinwart and A. Christmann, *Support Vector Machines*, Springer, New York, 2008.
[42] W. T. Stephenson and T. Broderick, *Approximate cross-validation in high dimensions with guarantees*, in AISTATS, PMLR, 2020, pp. 2424–2434.
[43] W. T. Stephenson, M. Udell, and T. Broderick, *Approximate cross-validation with low-rank data in high dimensions*, in NeurIPS, Vol. 33, 2020, pp. 9830–9840.
[44] L. N. Trefethen and D. Bau III, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.
[45] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Fixed-rank approximation of a positive-semidefinite matrix from streaming data*, in NIPS, Vol. 30, 2017, pp. 1225–1234.
[46] J. Vanschoren, J. N. Van Rijn, B. Bischl, and L. Torgo, *OpenML: Networked science in machine learning*, ACM SIGKDD Explorations Newsletter, 15 (2014), pp. 49–60.
[47] K. Wang, G. Pleiss, J. Gardner, S. Tyree, K. Q. Weinberger, and A. G. Wilson, *Exact Gaussian processes on a million data points*, in NeurIPS 32, 2019.
[48] C. K. Williams and M. Seeger, *Using the Nyström method to speed up kernel machines*, in NIPS, Vol. 13, 2001, pp. 682–688.
[49] A. Wilson, M. Kasy, and L. Mackey, *Approximate cross-validation: Guarantees for model assessment and selection*, in AISTATS, PMLR, 2020, pp. 4530–4540.
[50] D. P. Woodruff, *Sketching as a tool for numerical linear algebra*, Found. Trends Theoret. Comput. Sci., 10 (2014), pp. 1–157.