FAST & ACCURATE RANDOMIZED ALGORITHMS FOR LINEAR SYSTEMS AND EIGENVALUE PROBLEMS*

YUJI NAKATSUKASA† AND JOEL A. TROPP‡

Abstract. This paper develops a new class of algorithms for general linear systems and eigenvalue problems. These algorithms apply fast randomized sketching to accelerate subspace projection methods, such as GMRES and Rayleigh–Ritz. This approach offers great flexibility in designing the basis for the approximation subspace, which can improve scalability in many computational environments. The resulting algorithms outperform the classic methods with minimal loss of accuracy. For model problems, numerical experiments show large advantages over MATLAB's optimized routines, including a 100× speedup over `gmres` and a 10× speedup over `eigs`.

Key words. Eigenvalue problem, linear system, numerical linear algebra, Petrov–Galerkin method, projection method, randomized algorithm, Rayleigh–Ritz, sketching, subspace embedding

AMS subject classifications. 65F10, 65F15, 65F25

1. Introduction.

Arguably, the most exciting recent development in numerical linear algebra (NLA) is the advent of new randomized algorithms that are fast, scalable, robust, and reliable. For example, many practitioners have adopted the "randomized SVD" and its relatives [20, 26] to compute truncated singular value decompositions of large matrices. Randomized preconditioning [36, 3] allows us to solve highly overdetermined least-squares problems faster than any classical algorithm.

In spite of these successes, we have made less progress on other core challenges from NLA, especially problems involving nonsymmetric square matrices. This paper exposes a new class of algorithms for solving general linear systems and eigenvalue problems. Our framework enhances subspace projection methods [37, 38], such as GMRES and the Rayleigh–Ritz process, with the modern technique of randomized sketching [41, 52, 26]. This approach allows us to accelerate the existing methods by incorporating approximation subspaces that are easier to construct. The resulting algorithms are faster than their classical counterparts, without much loss of accuracy. In retrospect, the marriage of these ideas appears foreordained.

1.1. Sketching a least-squares problem.

The *sketch-and-solve* paradigm [41, 52, 26] is a basic tool for randomized matrix computations. The idea is to decrease the dimension of a large problem by projecting it onto a random subspace and to solve the smaller problem instead. The solution of this "sketched problem" sometimes serves in place of the solution to the original computational problem.

For a typical example, consider the $n \times d$ overdetermined least-squares problem

$$\text{minimize}_{\boldsymbol{y} \in \mathbb{C}^d} \quad \|\boldsymbol{M}\boldsymbol{y} - \boldsymbol{f}\|_2, \tag{1.1}$$

where $\boldsymbol{M} \in \mathbb{C}^{n \times d}$ is a tall matrix with $n \gg d$. Draw a random sketching matrix $\boldsymbol{S} \in \mathbb{C}^{s \times n}$ with sketch dimension $s = 2d$, say. Then solve the $s \times d$ sketched problem

$$\text{minimize}_{\boldsymbol{y} \in \mathbb{C}^d} \quad \|\boldsymbol{S}(\boldsymbol{M}\boldsymbol{y} - \boldsymbol{f})\|_2. \tag{1.2}$$

For a carefully designed sketching matrix $\boldsymbol{S}$, the whole sketch-and-solve process may be significantly faster than solving (1.1) directly. See section 2 for details.

*Submitted to the editors DATE.

Funding: JAT acknowledges ONR BRC N00014-1-18-2363 and NSF FRG 1952777.

†Mathematical Institute, University of Oxford, Oxford, UK (nakatsukasa@maths.oxford.ac.uk).

‡Computing and Mathematical Sciences, Caltech, Pasadena, CA, USA (jtropp@cms.caltech.edu).

We can compare the residual norms of the solution $\hat{\boldsymbol{y}}$ to the sketched problem (1.2) and the solution $\boldsymbol{y}_\star$ to the original problem (1.1). The sketching method ensures that

$$\|\boldsymbol{M}\boldsymbol{y}_\star - \boldsymbol{f}\|_2 \quad \leq \quad \|\boldsymbol{M}\hat{\boldsymbol{y}} - \boldsymbol{f}\|_2 \quad \leq \quad \text{Const} \cdot \|\boldsymbol{M}\boldsymbol{y}_\star - \boldsymbol{f}\|_2. \tag{1.3}$$

Provided that the original problem has a tiny residual, the solution to the sketched problem also yields a tiny residual!

1.2. Solving linear systems by sketched GMRES. Now, suppose that we wish to solve the (nonsymmetric) linear system

$$\text{Find } \boldsymbol{x} \in \mathbb{C}^n : \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{f} \quad \text{where } \boldsymbol{A} \in \mathbb{C}^{n\times n} \text{ and } \boldsymbol{f} \in \mathbb{C}^n. \tag{1.4}$$

All algorithms in this paper access the matrix via products: $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}$. Our approach builds on a classic template, called a *subspace projection method* [37], which casts the linear system as a variational problem. We can treat this formulation by sketching. Let us summarize the ideas; a full exposition appears in sections 3 and 4.

1.2.1. Sketched GMRES. For the moment, suppose that we have acquired a tall matrix $\boldsymbol{B} \in \mathbb{C}^{n\times d}$, called a *basis*, with the property that range($\boldsymbol{B}$) contains a good approximate solution to the linear system (1.4). That is, $\boldsymbol{A}\boldsymbol{B}\boldsymbol{y} \approx \boldsymbol{f}$ for some $\boldsymbol{y} \in \mathbb{C}^d$. In addition, assume we have the reduced matrix $\boldsymbol{A}\boldsymbol{B} \in \mathbb{C}^{n\times d}$ at hand.

At its heart, the GMRES algorithm [40, 39, 37] is a subspace projection method that replaces the linear system (1.4) with the overdetermined least-squares problem

$$\text{minimize}_{\boldsymbol{y}\in\mathbb{C}^d} \quad \|\boldsymbol{A}\boldsymbol{B}\boldsymbol{y} - \boldsymbol{f}\|_2. \tag{1.5}$$

The solution $\boldsymbol{y}_\star$ to (1.5) yields an approximate solution $\boldsymbol{x}_{\boldsymbol{B}} = \boldsymbol{B}\boldsymbol{y}_\star$ to the linear system (1.4). The residual norm $\|\boldsymbol{A}\boldsymbol{x}_{\boldsymbol{B}} - \boldsymbol{f}\|_2$ reflects how well the basis $\boldsymbol{B}$ captures a solution to the linear system.

The least-squares formulation (1.5) is a natural candidate for sketching. Draw a sketching matrix $\boldsymbol{S} \in \mathbb{C}^{s\times n}$ with $s = 2d$, say, and sketch the problem:

$$\text{minimize}_{\boldsymbol{y}\in\mathbb{C}^d} \quad \|\boldsymbol{S}(\boldsymbol{A}\boldsymbol{B}\boldsymbol{y} - \boldsymbol{f})\|_2. \tag{1.6}$$

The solution $\hat{\boldsymbol{y}}$ of the sketched problem (1.6) induces an approximate solution $\hat{\boldsymbol{x}} = \boldsymbol{B}\hat{\boldsymbol{y}}$ to the linear system (1.4). According to (1.3), the residual norm $\|\boldsymbol{A}\hat{\boldsymbol{x}} - \boldsymbol{f}\|_2$ is within a constant factor of the original residual norm $\|\boldsymbol{A}\boldsymbol{x}_{\boldsymbol{B}} - \boldsymbol{f}\|_2$. In summary, the sketched formulation (1.6) is effective if and only if the subspace range($\boldsymbol{B}$) contains a good solution of the linear system.

We refer to (1.6) as the *sketched GMRES* problem (sGMRES). For an unstructured basis $\boldsymbol{B}$, the sGMRES approach is faster than solving the original least-squares problem (1.5), both in theory and in practice. With careful implementation, sGMRES is reliable and robust, even when the conditioning of the reduced matrix $\boldsymbol{A}\boldsymbol{B}$ is poor. Indeed, it suffices that $\kappa_2(\boldsymbol{A}\boldsymbol{B}) < u^{-1}$ where u is the unit roundoff.[1] As a consequence, we have an enormous amount of flexibility in choosing the basis $\boldsymbol{B}$.

1.2.2. Krylov subspaces. To make sGMRES work well, we must construct a subspace that captures an approximate solution to the linear system. To that end, consider a Krylov subspace of the form

$$\mathsf{K}_p(\boldsymbol{A}, \boldsymbol{f}) := \text{span}\{\boldsymbol{f}, \boldsymbol{A}\boldsymbol{f}, \boldsymbol{A}^2\boldsymbol{f}, \dots, \boldsymbol{A}^{p-1}\boldsymbol{f}\}. \tag{1.7}$$

[1] The unit roundoff $u \approx 10^{-16}$ in standard IEEE double-precision arithmetic.

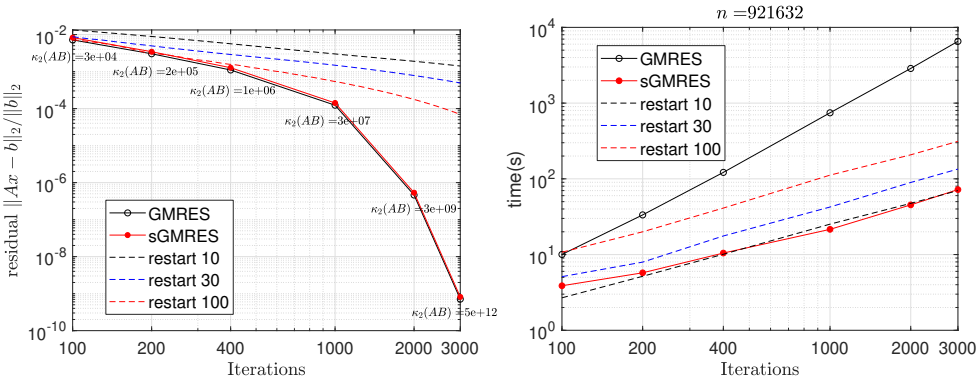

FIG. 1. ***GMRES versus sGMRES: Nonsymmetric linear system.*** *These panels compare the performance of MATLAB* `gmres` *(with and without restarting) against the sGMRES algorithm (where the basis* $\boldsymbol{B}$ *is computed by partial orthogonalization against* $k = 4$ *previous vectors). The sparse linear system* $\boldsymbol{Ax} = \boldsymbol{f}$ *has dimension* $n = 921,632$. ***Left:*** *Relative residual and the conditioning* $\kappa_2(\boldsymbol{AB})$ *of the reduced matrix.* ***Right:*** *Total runtime including basis generation.*

The Krylov subspace often contains an excellent approximate solution to the linear system, even when the depth $p \ll n$. See [37, Chaps. 6 and 7].

For computations, we need an explicit basis for the Krylov subspace. Although it is straightforward to form the monomial basis visible in (1.7), the condition number may grow exponentially [7, 17], rendering the basis useless for numerical purposes.

Instead, we pursue other fast basis constructions that offer better conditioning. For example, we may assemble a basis $\boldsymbol{B} = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_d] \in \mathbb{C}^{n\times d}$ using the Arnoldi process with *k-partial orthogonalization*. Define $\boldsymbol{b}_0 = \boldsymbol{0}$ and $\boldsymbol{b}_1 = \boldsymbol{f}/\|\boldsymbol{f}\|_2$. For each $j = 2, \dots, d$, we iteratively form

$$\boldsymbol{b}_j = \boldsymbol{w}_j/\|\boldsymbol{w}_j\|_2 \quad \text{where} \quad \boldsymbol{w}_j = (\mathbf{I} - \boldsymbol{b}_{j-1}\boldsymbol{b}_{j-1}^* - \cdots - \boldsymbol{b}_{j-k}\boldsymbol{b}_{j-k}^*)(\boldsymbol{A}\boldsymbol{b}_{j-1}). \tag{1.8}$$

It often suffices to take $k = 2$ or $k = 4$. Note that we obtain the reduced matrix $\boldsymbol{AB}$ as a by-product of this computation.

1.2.3. Comparison with GMRES. The standard version of GMRES [40] applies the expensive Arnoldi process (with full orthogonalization) to build an orthonormal basis for the Krylov subspace, and it exploits the structure of this basis to solve the least-squares problem (1.5) efficiently.

In contrast, we propose to use a quick-and-dirty construction, such as Arnoldi with k-partial orthogonalization, to obtain a basis for the Krylov subspace. Then we solve the sGMRES least-squares problem (1.6) to produce an approximate solution of the linear system. The sGMRES approach has lower arithmetic costs than classic GMRES, while attaining similar accuracy:

GMRES: $O(nd^2)$ operations vs. sGMRES: $O(d^3 + nd\log d)$ operations.

This expression assumes that k is a fixed constant.

As evidence of the benefits of sGMRES, Figure 1 depicts a **100× speedup** for a sparse nonsymmetric linear system with dimension $n = 921,632$. In this case, sGMRES is comparable in speed to restarted GMRES with restarting frequency 10, whose convergence is significantly impaired. Section 8 provides more details on the experimental setup, as well as further illustrations. For example, when applied to a

sparse positive-definite linear system, sGMRES can produce residual norms about 5× smaller than the conjugate gradient (CG) method after the same running time. Thus, it can be argued that sGMRES combines the speed of CG with the generality and robustness of GMRES.

1.3. Solving eigenvalue problems by sketched Rayleigh–Ritz. Similar ideas apply to spectral computations. We pose the nonsymmetric eigenvalue problem

$$\text{Find nonzero } \boldsymbol{x} \in \mathbb{C}^n \text{ and } \lambda \in \mathbb{C} : \quad \boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x} \quad \text{where } \boldsymbol{A} \in \mathbb{C}^{n \times n}. \tag{1.9}$$

As before, we access the matrix via products: $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}$. Typically, we seek a family of eigenvectors associated with a particular class of eigenvalues (e.g., largest real part, closest to zero). Let us outline a sketched subspace projection method for the eigenvalue problem. Full details appear in sections 6 and 7.

1.3.1. Sketched Rayleigh–Ritz. As in subsection 1.2.1, suppose that we are equipped with a basis $\boldsymbol{B} \in \mathbb{C}^{n \times d}$ and the reduced matrix $\boldsymbol{A}\boldsymbol{B} \in \mathbb{C}^{n \times d}$. The range of the basis should contain approximate eigenpairs $(\boldsymbol{x}, \lambda)$ for which $\boldsymbol{A}\boldsymbol{x} \approx \lambda \boldsymbol{x}$. In this setting, the most commonly employed strategy is the Rayleigh–Ritz (RR) method.

We begin with the classic variational formulation [33, Thm. 11.4.2] of RR:

$$\text{minimize}_{\boldsymbol{M} \in \mathbb{C}^{d \times d}} \quad \|\boldsymbol{A}\boldsymbol{B} - \boldsymbol{B}\boldsymbol{M}\|_{\mathrm{F}}. \tag{1.10}$$

The solution is $\boldsymbol{M} = \boldsymbol{B}^{\dagger}\boldsymbol{A}\boldsymbol{B}$. At this point, RR frames the $d \times d$ eigenvalue problem $\boldsymbol{M}\boldsymbol{y} = \theta \boldsymbol{y}$. Each solution yields an approximate eigenpair $(\boldsymbol{B}\boldsymbol{y}, \theta)$ of the matrix $\boldsymbol{A}$.

Evidently, the least-squares problem (1.10) is ripe for sketching. Draw a sketching matrix $\boldsymbol{S} \in \mathbb{C}^{s \times n}$ with $s = 4d$, say, and pass to the sketched RR problem:

$$\text{minimize}_{\boldsymbol{M} \in \mathbb{C}^{d \times d}} \quad \|\boldsymbol{S}(\boldsymbol{A}\boldsymbol{B} - \boldsymbol{B}\boldsymbol{M})\|_{\mathrm{F}}. \tag{1.11}$$

It is cheap to form a solution $\hat{\boldsymbol{M}} = (\boldsymbol{S}\boldsymbol{B})^{\dagger}(\boldsymbol{S}\boldsymbol{A}\boldsymbol{B})$ to the sketched problem (1.11). As before, we can frame an ordinary eigenvalue problem

$$\hat{\boldsymbol{M}}\boldsymbol{y} = (\boldsymbol{S}\boldsymbol{B})^{\dagger}(\boldsymbol{S}\boldsymbol{A}\boldsymbol{B})\boldsymbol{y} = \theta \boldsymbol{y}. \tag{1.12}$$

For each solution $(\hat{\boldsymbol{y}}, \hat{\theta})$, we obtain an approximate eigenpair $(\boldsymbol{B}\hat{\boldsymbol{y}}, \hat{\theta})$ of the original matrix $\boldsymbol{A}$. We will show—both theoretically and empirically—that the computed eigenpairs are competitive with the eigenpairs obtained from RR.

We refer to (1.11) as the sketched Rayleigh–Ritz (sRR) formulation. Although it demands a careful implementation, sRR is faster than the original least-squares method (1.10) for an unstructured basis $\boldsymbol{B}$. Moreover, sRR is robust, even when the basis $\boldsymbol{B}$ has poor conditioning. Indeed, it suffices that $\kappa_2(\boldsymbol{B}) < u^{-1}$.

1.3.2. Comparison with Arnoldi + Rayleigh–Ritz. We can also deploy the Krylov subspace (1.7) for eigenvalue computations [33, 38]. In this case, we typically use a *random* starting vector $\boldsymbol{\omega} \in \mathbb{C}^n$ to generate the subspace $\mathsf{K}_p(\boldsymbol{A}; \boldsymbol{\omega})$.

To solve a large nonsymmetric eigenvalue problem, one standard algorithm [38, Sec. 6.2] applies the Arnoldi process (with full orthogonalization) to form an orthonormal basis for the Krylov subspace, and it uses the structure of the basis to solve the RR eigenvalue problem efficiently.

Instead, we propose to combine a fast construction of a Krylov subspace basis, such as k-partial orthogonalization (1.8), with the sRR eigenvalue problem (1.12). Asymptotically, this algorithm uses less arithmetic than the classic approach.

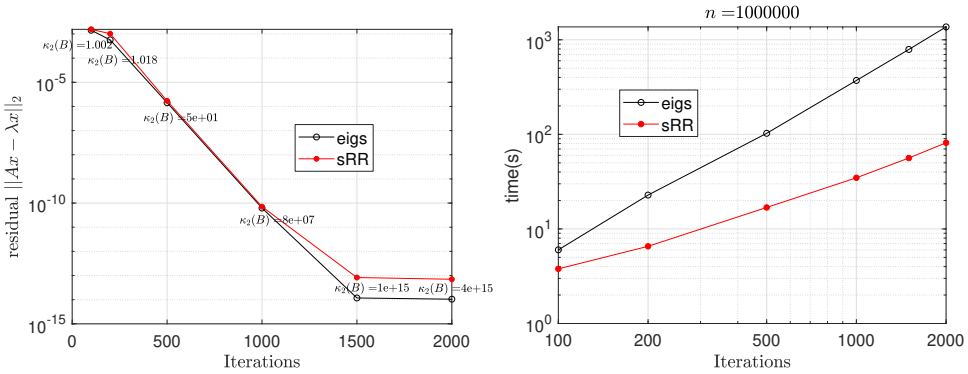

FIG. 2. ***RR versus sRR: Nonsymmetric eigenvalue problem.*** *These panels compare the performance of MATLAB* `eigs` *against the sRR algorithm (where the basis* $\boldsymbol{B}$ *is computed by partial orthogonalization against* $k = 10$ *previous vectors). The sparse eigenvalue problem* $\boldsymbol{Ax} = \lambda \boldsymbol{x}$ *has dimension* $n = 10^6$*, and it arises from a trust-region subproblem in optimization.* ***Left:*** *Relative residual for the right-most eigenpair and the conditioning* $\kappa_2(\boldsymbol{B})$ *of the basis.* ***Right:*** *Total runtime including basis generation.*

As evidence, Figure 2 highlights an eigenvalue problem from optimization where sRR runs over **10× faster** than the MATLAB `eigs` command. Even so, both methods compute the desired eigenpair to the same accuracy. Section 8 describes the experimental setup and provides further illustrations. For example, when applied to a sparse symmetric eigenvalue problem, sRR can outperform standard implementations of the Lanczos method in both speed and reliability.

1.3.3. Block Krylov subspaces. For eigenvalue problems, there is also a compelling opportunity to explore alternative subspace constructions. For example, consider the *block* Krylov subspace

$$\mathsf{K}_p(\boldsymbol{A}, \boldsymbol{\Omega}) := \operatorname{span}\{\boldsymbol{\Omega}, \boldsymbol{A\Omega}, \boldsymbol{A}^2\boldsymbol{\Omega}, \dots, \boldsymbol{A}^{p-1}\boldsymbol{\Omega}\} \quad \text{where } \boldsymbol{\Omega} \in \mathbb{C}^{n\times b}. \tag{1.13}$$

We commonly generate the Krylov subspace from a random matrix $\boldsymbol{\Omega}$. The standard prescription recommends a small block size b and a large depth p, but recent research [26, Sec. 11] has shown the value of a large block size b and a small depth p.

We must take care in constructing the block Krylov subspace. Partial orthogonalization is only competitive when the block size b is a small constant. For larger b, the Chebyshev recurrence offers an elegant way to form a basis $\boldsymbol{B} = [\boldsymbol{B}_1, \dots, \boldsymbol{B}_p] \in \mathbb{C}^{n\times bp}$:

$$\boldsymbol{B}_1 = \boldsymbol{\Omega}; \qquad \boldsymbol{B}_2 = \boldsymbol{A\Omega}; \qquad \boldsymbol{B}_i = 2\boldsymbol{A}\boldsymbol{B}_{i-1} - \boldsymbol{B}_{i-2} \quad \text{for } i = 3, \dots, p.$$

We obtain the reduced matrix $\boldsymbol{AB}$ as a by-product. In practice, the Chebyshev polynomials must be shifted and scaled to adapt to the spectrum of $\boldsymbol{A}$.

1.4. Discussion. Our idea to combine subspace projection methods with sketching offers compelling advantages over the classic algorithms, especially in modern computing environments. Nevertheless, it must be acknowledged that this approach suffers from some of the same weaknesses as GMRES and RR. For example, when the basis $\boldsymbol{B}$ is a Krylov subspace, these methods are limited by the approximation power of Krylov subspaces.

With hindsight, our framework appears as an obvious application of the sketch-and-solve paradigm for overdetermined least-squares problems. A critical reader may even wonder whether this idea is actually novel. Let us respond to this concern.

First, the sketch-and-solve methodology is typically used to design fast, low-accuracy algorithms that have limited access to the problem data. Indeed, the ostensible purpose of sketching is to reduce the input data to a more manageable size. To the best of our knowledge, the literature has not evaluated sketching for *data-adaptive* problem formulations, such as the ones in this paper.

Second, we believe that we are the first authors to identify the natural connection between sketching and subspace projection methods for linear algebra problems. In the past, researchers in theoretical algorithms may have overlooked this opportunity because they did not recognize that subspace projection algorithms are variational methods. Meanwhile, numerical analysts who are familiar with the variational perspective may not have appreciated that sketching is compatible with accuracy.

Third, for our algorithms to be competitive, we must employ efficient constructions of Krylov subspace bases. These ideas have a long history, but they are not widespread. In the past, researchers presented these techniques as a way to *postpone* expensive orthogonalization steps in parallel computing environments [23, 34]. In contrast, sketching sometimes allows us to *eliminate* the orthogonalization steps. Thus, we can finally take full advantage of the potential of fast computational bases.

More broadly, orthogonal bases and transformations historically played an important role because they guarantee backward stability. In contrast, we will demonstrate that non-orthogonal bases can support fast and accurate matrix computations. We believe that there are many other settings where this insight will be valuable.

1.5. Roadmap. In section 2, we give a rigorous treatment of sketching for least-squares problems. Sections 3 to 5 develop and analyze the sGMRES method and associated basis constructions. Sections 6 and 7 contain the analogous developments for sRR. Computational experiments in section 8 confirm that these algorithms are fast, robust, and reliable. Sections 9 and 10 describe extensions and prospects.

1.6. Notation. The symbol * denotes the (conjugate) transpose of a vector or matrix. We write $\|\cdot\|_2$ for the ℓ_2 norm or the spectral norm, while $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm. The dagger $\dagger$ denotes the pseudoinverse. For a matrix $\boldsymbol{M} \in \mathbb{C}^{n\times d}$, define the largest singular value $\sigma_{\max}(\boldsymbol{M}) := \sigma_1(\boldsymbol{M})$ and the minimum singular value $\sigma_{\min}(\boldsymbol{M}) := \sigma_{\min\{n,d\}}(\boldsymbol{M})$. The condition number $\kappa_2(\boldsymbol{M}) := \sigma_{\max}(\boldsymbol{M})/\sigma_{\min}(\boldsymbol{M})$.

2. Background: Subspace embeddings. A subspace embedding is a linear map, usually from a high-dimensional space to a low-dimensional space, that preserves the ℓ_2 norm of every vector in a given subspace. This definition is due to Sarlós [41]; see also [52, 26].

DEFINITION 2.1 (Subspace embedding). *Suppose that the columns of $\boldsymbol{B} \in \mathbb{C}^{n\times d}$ span the subspace $\mathsf{L} \subseteq \mathbb{C}^n$. A matrix $\boldsymbol{S} \in \mathbb{C}^{s\times n}$ is called a subspace embedding for L with distortion $\varepsilon \in (0,1)$ if*

$$(1-\varepsilon)\cdot\|\boldsymbol{B}\boldsymbol{y}\|_2 \le \|\boldsymbol{S}\boldsymbol{B}\boldsymbol{y}\|_2 \le (1+\varepsilon)\cdot\|\boldsymbol{B}\boldsymbol{y}\|_2 \quad \text{for all } \boldsymbol{y}\in\mathbb{C}^d. \tag{2.1}$$

For matrix computations, we need to design subspace embeddings with several extra properties. First, the subspace embedding $\boldsymbol{S}$ should be equipped with a fast matrix–vector multiply so that we can perform the data reduction process efficiently. Second, for a fixed (but unknown) subspace L, we want to draw a subspace embedding *at random* to achieve (2.1) with high probability. Last, the embedding dimension s for a randomized subspace embedding of a d-dimensional subspace should have the optimal scaling $s \approx d/\varepsilon^2$. Owing to this relation, subspace embeddings are only

appropriate in settings where a moderate distortion, say $\varepsilon = 1/\sqrt{2}$, is enough for computational purposes.

Before turning to constructions in subsection 2.3, let us outline the applications of subspace embeddings that we will use in this paper.

2.1. Sketching for least-squares problems. As discussed in subsection 1.1, we can use a subspace embedding to reduce the dimension of an overdetermined least-squares problem. This idea is also due to Sarlós [41]; it serves as the foundation for a collection of methods called the *sketch-and-solve* paradigm [52, 26].

FACT 2.2 (Sketching for least-squares). *Let $\boldsymbol{M} \in \mathbb{C}^{n\times d}$ be a matrix, and suppose that $\boldsymbol{S} \in \mathbb{C}^{s\times n}$ is a subspace embedding for* range$([\boldsymbol{M}, \boldsymbol{f}])$ *with distortion $\varepsilon \in (0, 1)$. For every vector $\boldsymbol{y} \in \mathbb{C}^d$, we have the two-sided inequality*

$$(1-\varepsilon)\cdot\|\boldsymbol{M}\boldsymbol{y}-\boldsymbol{f}\|_2 \le \|\boldsymbol{S}(\boldsymbol{M}\boldsymbol{y}-\boldsymbol{f})\|_2 \le (1+\varepsilon)\cdot\|\boldsymbol{M}\boldsymbol{y}-\boldsymbol{f}\|_2. \tag{2.2}$$

In particular, the solution $\boldsymbol{y}_\star$ to the least-squares problem (1.1) and the solution $\hat{\boldsymbol{y}}$ to the sketched least-squares problem (1.2) satisfy residual norm bounds

$$\|\boldsymbol{M}\boldsymbol{y}_\star-\boldsymbol{f}\|_2 \le \|\boldsymbol{M}\hat{\boldsymbol{y}}-\boldsymbol{f}\|_2 \le \frac{1+\varepsilon}{1-\varepsilon}\cdot\|\boldsymbol{M}\boldsymbol{y}_\star-\boldsymbol{f}\|_2. \tag{2.3}$$

Equation (2.3) justifies the claim (1.3).

Remark 2.3 (Accurate sketch-and-solve methods?). The sketch-and-solve paradigm has a poor reputation among numerical analysts because of a perception that it yields results of deplorable accuracy. For least-squares problems, this criticism is valid when the residual norm is large, which is often the case in data-fitting applications. In contrast, we will apply the technique when the residual is small (say, 10^{-10}), so we can well afford a constant-factor loss.

2.2. Whitening the basis. Rokhlin & Tygert [36] observed that a subspace embedding yields an inexpensive way to precondition an iterative algorithm for the overdetermined least-squares problem. We can invoke the same idea to approximately orthogonalize, or *whiten*, a given basis.

FACT 2.4 (Whitening). *Let $\boldsymbol{B} \in \mathbb{C}^{n\times d}$ be a basis with full column rank. Let $\boldsymbol{S} \in \mathbb{C}^{s\times n}$ be a subspace embedding for* range$(\boldsymbol{B})$ *with distortion $\varepsilon \in (0, 1)$. Compute the QR factorization of the sketched basis, $\boldsymbol{S}\boldsymbol{B} = \boldsymbol{U}\boldsymbol{T}$. Then the whitened basis $\bar{\boldsymbol{B}} = \boldsymbol{B}\boldsymbol{T}^\dagger$ satisfies*

$$\kappa_2(\bar{\boldsymbol{B}}) = \frac{\sigma_{\max}(\bar{\boldsymbol{B}})}{\sigma_{\min}(\bar{\boldsymbol{B}})} \le \frac{1+\varepsilon}{1-\varepsilon}. \tag{2.4}$$

Furthermore, we have the condition number diagnostic

$$\frac{1-\varepsilon}{1+\varepsilon}\cdot\kappa_2(\boldsymbol{T}) \le \kappa_2(\boldsymbol{B}) \le \frac{1+\varepsilon}{1-\varepsilon}\cdot\kappa_2(\boldsymbol{T}). \tag{2.5}$$

2.3. Constructing a subspace embedding. There are many performant constructions of fast randomized subspace embeddings that work for an unknown subspace of bounded dimension [26, Sec. 9]. Let us summarize two that are most relevant for our purposes. In each case, for a subspace with dimension d, to obtain empirical distortion $\varepsilon \in (0, 1)$, we set the embedding dimension $s = d/\varepsilon^2$.

2.3.1. SRFTs. First, we introduce the subsampled random Fourier transform (SRFT) [2, 53, 47, 26]. This subspace embedding[2] takes the form

$$\boldsymbol{S} = \sqrt{\frac{n}{s}}\boldsymbol{DFE} \in \mathbb{C}^{s\times n}. \tag{2.6}$$

In this expression, $\boldsymbol{D} \in \mathbb{C}^{s\times n}$ is a diagonal projector onto s coordinates, chosen independently at random, $\boldsymbol{F} \in \mathbb{C}^{n\times n}$ is the unitary discrete Fourier transform (DFT), and $\boldsymbol{E} \in \mathbb{C}^{n\times n}$ is a diagonal matrix whose entries are independent Steinhaus[3] random variables. The cost of applying the matrix $\boldsymbol{S}$ to an $n \times d$ matrix is $O(nd \log d)$ operations using the subsampled FFT algorithm [53].

2.3.2. Sparse maps. Next, we describe the sparse dimension reduction map [27, 31, 10, 11, 26], which is useful for sparse data and may require less data movement. It takes the form

$$\boldsymbol{S} = \frac{1}{\sqrt{s}}[\boldsymbol{s}_1, \dots, \boldsymbol{s}_n] \in \mathbb{C}^{s\times n}. \tag{2.7}$$

The columns of $\boldsymbol{S}$ are statistically independent. Each column $\boldsymbol{s}_i$ has exactly ζ nonzero entries, drawn from the Steinhaus distribution, placed in uniformly random coordinates. For reliability, we can choose the sparsity level $\zeta = \lceil 2\log(1+d)\rceil$. We can apply $\boldsymbol{S}$ to a matrix $\boldsymbol{M}$ with $O(\zeta \cdot \mathtt{nnz}(\boldsymbol{M}))$ operations, but it may require a sparse arithmetic library to achieve the best performance.

3. Solving linear systems with sGMRES.

We return to the linear system

$$\text{Find } \boldsymbol{x} \in \mathbb{C}^n : \quad \boldsymbol{Ax} = \boldsymbol{f} \quad \text{where } \boldsymbol{A} \in \mathbb{C}^{n\times n} \text{ and } \boldsymbol{f} \in \mathbb{C}^n. \tag{3.1}$$

This section elaborates on the sGMRES method outlined in subsection 1.2. Section 4 discusses methods for constructing the basis required by sGMRES. Section 5 combines these ideas to obtain complete sGMRES algorithms.

3.1. Derivation of GMRES. We begin with a basis $\boldsymbol{B} \in \mathbb{C}^{n\times d}$ and the reduced matrix $\boldsymbol{AB} \in \mathbb{C}^{n\times d}$. Suppose that $\boldsymbol{x}_0 \in \mathbb{C}^n$ is an initial guess for the solution of (3.1) with residual $\boldsymbol{r}_0 := \boldsymbol{f} - \boldsymbol{Ax}_0$. Lacking prior information, we may take $\boldsymbol{x}_0 = \boldsymbol{0}$.

Consider the affine family of approximate solutions to (3.1) of the form $\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{By}$ where $\boldsymbol{y} \in \mathbb{C}^d$. Among this class, we may select a representative whose residual $\boldsymbol{r} = \boldsymbol{f} - \boldsymbol{Ax} = \boldsymbol{r}_0 - \boldsymbol{ABy}$ has the minimum ℓ_2 norm:

$$\text{minimize}_{\boldsymbol{y}\in\mathbb{C}^d} \quad \|\boldsymbol{ABy} - \boldsymbol{r}_0\|_2. \tag{3.2}$$

With some imprecision, we refer to (3.2) as the GMRES problem [40]. By calculus, the least-squares problem (3.2) is equivalent to an orthogonality principle:

$$\text{Find } \boldsymbol{y} \in \mathbb{C}^d : \quad (\boldsymbol{AB})^*(\boldsymbol{ABy} - \boldsymbol{r}_0) = \boldsymbol{0}. \tag{3.3}$$

We can stably solve (3.2) or (3.3) using a QR factorization of the reduced matrix [21, Ch. 20]. The cost is $O(nd^2)$ arithmetic operations, assuming that the reduced matrix $\boldsymbol{AB}$ is unstructured. Given a solution $\boldsymbol{y_B}$ to either problem, we obtain a new approximate solution $\boldsymbol{x_B} = \boldsymbol{x}_0 + \boldsymbol{By_B}$ with residual $\boldsymbol{r_B} = \boldsymbol{f} - \boldsymbol{Ax_B}$.

[2] For worst-case problems, a more elaborate SRFT construction may be needed [26, Sec. 9].

[3] A *Steinhaus random variable* is uniform on the complex unit circle $\{z \in \mathbb{C} : |z| = 1\}$.

The formulation (3.3) is called a *Petrov–Galerkin method* [37, Chap. 5] with approximation space $\boldsymbol{x}_0 + \operatorname{range}(\boldsymbol{B})$ and orthogonality space $\operatorname{range}(\boldsymbol{AB})$. The GMRES algorithm [39, 40] is a particular instance where $\boldsymbol{B}$ is an orthonormal basis for a Krylov subspace generated by $\boldsymbol{r}_0$. GMRES forms the basis $\boldsymbol{B}$ via the Arnoldi process (subsection 4.2), which involves d matvecs with $\boldsymbol{A}$ plus $O(nd^2)$ arithmetic. This reduces (3.2) to a structured least-squares problem that can be solved in $O(d^2)$ operations.

3.2. Derivation and analysis of sGMRES. To develop the sGMRES method, we just sketch the GMRES problem (3.2). Construct a subspace embedding $\boldsymbol{S} \in \mathbb{C}^{s \times n}$ for $\operatorname{range}([\boldsymbol{AB}, \boldsymbol{r}_0])$ with distortion $\varepsilon \in (0, 1)$. The sketched GMRES problem is

$$\text{minimize}_{\boldsymbol{y} \in \mathbb{C}^d} \quad \|\boldsymbol{S}(\boldsymbol{ABy} - \boldsymbol{r}_0)\|_2. \tag{3.4}$$

Let $\hat{\boldsymbol{y}} \in \mathbb{C}^d$ denote any solution of (3.4). Write $\hat{\boldsymbol{x}} = \boldsymbol{x}_0 + \boldsymbol{B}\hat{\boldsymbol{y}}$ and $\hat{\boldsymbol{r}} = \boldsymbol{f} - \boldsymbol{A}\hat{\boldsymbol{x}}$.

We have an *a priori* comparison of the GMRES (3.2) and sGMRES (3.4) residual norms because of the relation (2.3):

$$\|\boldsymbol{Ax}_{\boldsymbol{B}} - \boldsymbol{f}\|_2 \le \|\boldsymbol{A}\hat{\boldsymbol{x}} - \boldsymbol{f}\|_2 \le \frac{1+\varepsilon}{1-\varepsilon} \cdot \|\boldsymbol{Ax}_{\boldsymbol{B}} - \boldsymbol{f}\|_2. \tag{3.5}$$

Thus, sGMRES produces good solutions to (3.1) precisely when GMRES does. *A posteriori*, we can diagnose the quality of the computed solution $\hat{\boldsymbol{x}}$ by examining the sketched residual norm:

$$\hat{r}_{\text{est}} := \|\boldsymbol{S}(\boldsymbol{AB}\hat{\boldsymbol{y}} - \boldsymbol{r}_0)\|_2 = (1 \pm \varepsilon) \cdot \|\boldsymbol{A}\hat{\boldsymbol{x}} - \boldsymbol{f}\|_2. \tag{3.6}$$

The last display is a consequence of (2.2).

For both GMRES (3.2) and sGMRES (3.4), the fundamental challenge is to produce a basis $\boldsymbol{B}$ that captures an approximate solution to the linear system (3.1). We return to this matter in section 4.

3.3. Implementation. Let us outline a numerically stable implementation of sGMRES and describe some of the issues that arise.

The algorithm operates with either an SRFT (2.6) or a sparse embedding (2.7), depending on which is more appropriate to the computational environment. We recommend the embedding dimension $s = 2(d+1)$, which typically yields distortion $\varepsilon = 1/\sqrt{2}$. In view of (3.5), the sGMRES residual norm is less than $6\times$ the GMRES residual norm, although the discrepancy is often smaller in practice.

To obtain the data for the sGMRES problem (3.4), we sketch the reduced matrix ($\boldsymbol{SAB} \in \mathbb{C}^{s \times d}$) and the right-hand side ($\boldsymbol{Sr}_0 \in \mathbb{C}^s$) at a cost of $O(nd \log d)$ operations. To solve (3.4), we compute a thin QR decomposition of the sketched matrix: $\boldsymbol{SAB} = \boldsymbol{UT}$. A minimizer of the sGMRES problem is

$$\hat{\boldsymbol{y}} = (\boldsymbol{SAB})^{\dagger}(\boldsymbol{Sr}_0) = \boldsymbol{T}^{\dagger}(\boldsymbol{U}^*(\boldsymbol{Sr}_0)). \tag{3.7}$$

The sketched residual norm (3.6) admits the simple expression

$$\hat{r}_{\text{est}} = \|(\mathbf{I} - \boldsymbol{UU}^*)(\boldsymbol{Sr}_0)\|_2. \tag{3.8}$$

The two preceding displays require $O(d^3)$ arithmetic since $s = O(d)$. Last, we explicitly form the approximate solution $\hat{\boldsymbol{x}} = \boldsymbol{x}_0 + \boldsymbol{B}\hat{\boldsymbol{y}}$ at a cost of $O(nd)$ operations.

In summary, given the basis $\boldsymbol{B}$, the cost of forming and solving the sGMRES problem (3.4) is $O(d^3 + nd \log d)$ arithmetic. In contrast, for an unstructured basis, the cost of solving the GMRES problem (3.2) is $O(nd^2)$ arithmetic. Section 5 contains pseudocode for sGMRES, along with a more detailed accounting of the costs of forming the basis and solving the least-squares problem.

3.4. Stability. The classical stability result [21, Thm. 20.3] shows that standard numerical methods for the least-squares problem (3.4) produce a solution with essentially optimal residual as long as $\kappa_2(\boldsymbol{SAB}) \lesssim u^{-1}$. According to (2.5), this condition is equivalent to $\kappa_2(\boldsymbol{AB}) \lesssim u^{-1}$.

Our computational work (section 8) confirms that sGMRES is reliable unless the reduced matrix $\boldsymbol{AB}$ is very badly conditioned. In our experience, it suffices that $\kappa_2(\boldsymbol{AB}) \leq 10^{14}$ in double-precision arithmetic. Therefore, we have wide latitude to design bases that we can construct quickly; see section 4. We will provide evidence that sGMRES with a fast basis construction is more efficient than GMRES with a structured basis.

3.5. Restarting. Standard implementations of GMRES periodically restart [37, Sec. 6.5.5]. That is, they use a basis $\boldsymbol{B}$ to compute an approximate solution $\boldsymbol{x_B}$ to the linear system (3.1) with the residual vector $\boldsymbol{r_B} = \boldsymbol{r}_0 - \boldsymbol{Ax_B}$. If the residual norm $\|\boldsymbol{r_B}\|_2$ exceeds an error tolerance, the residual vector $\boldsymbol{r_B}$ is used to generate a new basis, which is fed back to GMRES to construct another approximate solution. This process is repeated until a solution of desired quality is obtained.

Restarting has a number of benefits for the process of basis construction. It allows us to work with bases that have fewer columns, which limits the cost of storing the basis. For orthogonal basis constructions, restarting reduces the cost of orthogonalization. For non-orthogonal basis constructions, the restarting process helps control the conditioning of the basis. On the other hand, restarted GMRES may not converge if the bases are not rich enough (see Figure 1). As we will discuss in section 5, sGMRES can help us manage all of these issues.

3.6. Preconditioning. For difficult linear systems where the matrix has badly distributed eigenvalues, we may need a preconditioner $\boldsymbol{P} \in \mathbb{C}^{n\times n}$ to solve it successfully with either GMRES or sGMRES. The preconditioned system has the form

$$\boldsymbol{P}^{-1}\boldsymbol{Ax} = \boldsymbol{P}^{-1}\boldsymbol{f}. \tag{3.9}$$

A good preconditioner has two features [37, Chaps. 9 and 10]. First, the matrix $\boldsymbol{P}^{-1}\boldsymbol{A}$ has a more "favorable" eigenvalue spectrum than $\boldsymbol{A}$. Second, we can solve $\boldsymbol{Pz} = \boldsymbol{g}$ efficiently. (Let us emphasize that we only interact with $\boldsymbol{P}^{-1}$ by solving linear systems!) Although preconditioning is critical in practice, it is heavily problem dependent, so we will not delve into examples.

We may derive sGMRES for the preconditioned system (3.9), following the same pattern as before. Note that we employ the preconditioned matrix $\boldsymbol{P}^{-1}\boldsymbol{A}$ when we construct the basis $\boldsymbol{B}$ and the reduced matrix $\boldsymbol{P}^{-1}(\boldsymbol{AB})$. The details are routine.

3.7. Variations. There are many other subspace projection methods for solving linear systems [37, Chap. 5]. We anticipate that some of these approaches may be accelerated by sketching. For example, we can sketch the flexible GMRES method [37, Chap. 9], which produces basis vectors that are not necessarily correlated.

4. Constructing a basis for sGMRES.

As we have seen, the success of both GMRES (3.2) and sGMRES (3.4) hinges on the approximation power of the basis. Krylov subspaces are, perhaps, the most natural way to capture solutions to a linear system when we access the matrix via products [37, Chaps. 6 and 7]. In this section, we describe a number of ways to compute bases for Krylov subspaces. Although these strategies are decades old, they deserve a fresh look because sGMRES has a fundamentally different computational profile from GMRES.

4.1. The single-vector Krylov subspace. Many iterative methods for solving the linear system (3.1) implicitly search for solutions in the Krylov subspace

$$\mathsf{K}_p(\boldsymbol{A}; \boldsymbol{r}) := \operatorname{span}\{\boldsymbol{r}, \boldsymbol{A}\boldsymbol{r}, \boldsymbol{A}^2\boldsymbol{r}, \dots, \boldsymbol{A}^{p-1}\boldsymbol{r}\} = \operatorname{span}\{\varphi(\boldsymbol{A})\boldsymbol{r} : \deg(\varphi) \le p-1\}.$$

In this context, the generating vector $\boldsymbol{r} \in \mathbb{C}^n$ is often the normalized residual $\boldsymbol{r}_0/\|\boldsymbol{r}_0\|_2$, defined by $\boldsymbol{r}_0 = \boldsymbol{f} - \boldsymbol{A}\boldsymbol{x}_0$, where $\boldsymbol{x}_0$ is an approximate solution to (3.1). The function φ ranges over polynomials with degree at most $p-1$.

A basis $\boldsymbol{B} \in \mathbb{C}^{n\times d}$ for the Krylov subspace $\mathsf{K}_p(\boldsymbol{A}; \boldsymbol{r})$ comprises a system of vectors that spans the subspace. We can write

$$\boldsymbol{B} = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_d] \quad \text{where} \quad \boldsymbol{b}_j = \varphi_j(\boldsymbol{A})\boldsymbol{r} \quad \text{for } j = 1, \dots, d.$$

The filter polynomials $(\varphi_j : j = 1, \dots, d)$ have degree at most $p-1$, and they are usually linearly independent (so $d = p$). In most cases, the polynomials are also graded ($\deg(\varphi_j) = j-1$), and they are constructed sequentially by a recurrence. This process delivers the reduced matrix $\boldsymbol{AB}$ without any extra work.

For example, the monomial basis takes the form $\boldsymbol{b}_1 = \boldsymbol{r}$ and $\boldsymbol{b}_j = \boldsymbol{A}\boldsymbol{b}_{j-1}$ for $j = 2, \dots, p$. The associated polynomials are $\varphi_j(t) = t^{j-1}$ for $j = 1, \dots, p$. For many matrices $\boldsymbol{A}$, the conditioning of the monomial basis for $\mathsf{K}_p(\boldsymbol{A}; \boldsymbol{r})$ grows exponentially with p, so it is inimical to numerical computation [17].

We will consider other constructions of Krylov subspace bases that are more suitable in practice. Our aim is to control the resources used to obtain the basis, including arithmetic, (working) storage, communication, synchronization, etc. We can advance these goals by relaxing the requirement that the basis be well-conditioned, which has historically been viewed as essential.

For theoretical analysis of the approximation power of Krylov subspaces in the context of linear system solvers, see [37, Sec. 6.11].

4.2. The Arnoldi process. It is supremely natural to build an *orthonormal* basis $\boldsymbol{Q} \in \mathbb{C}^{n\times p}$ for the Krylov subspace $\mathsf{K}_p(\boldsymbol{A}; \boldsymbol{r})$ sequentially. This is called the Arnoldi process [37, Sec. 6.3]. The initial vector $\boldsymbol{q}_1 = \boldsymbol{r}/\|\boldsymbol{r}\|_2$. After j steps, the method updates the partial basis $\boldsymbol{Q}_j = [\boldsymbol{q}_1, \dots, \boldsymbol{q}_j]$ by appending the vector

$$\boldsymbol{q}_{j+1} = \boldsymbol{w}_{j+1}/\|\boldsymbol{w}_{j+1}\|_2 \quad \text{where} \quad \boldsymbol{w}_{j+1} = (\mathbf{I} - \boldsymbol{Q}_j\boldsymbol{Q}_j^*)(\boldsymbol{A}\boldsymbol{q}_j).$$

The Arnoldi basis $\boldsymbol{Q}_p \in \mathbb{C}^{n\times p}$ has the happy property that

$$\boldsymbol{A}\boldsymbol{Q}_p = \boldsymbol{Q}_p\boldsymbol{H}_p + \boldsymbol{w}_p\mathbf{e}_p^* \quad \text{where } \boldsymbol{H}_p \in \mathbb{C}^{p\times p} \text{ is upper Hessenberg.}$$

As a consequence, we can solve the least-squares problem (3.2) with $\boldsymbol{B} = \boldsymbol{Q}_p$ in $O(p^2)$ time and produce the approximate solution $\boldsymbol{x}_{\boldsymbol{B}}$ in $O(np)$ operations. This is how the standard implementation of the GMRES algorithm operates [40].

The orthogonalization steps in the Arnoldi process are expensive. For p iterations, they expend $O(np^2)$ arithmetic, and they may also involve burdensome inner-products, communication, and synchronization. Robust implementations usually incorporate double Gram–Schmidt or Householder reflectors.

The literature contains many strategies for controlling the orthogonalization costs in the Arnoldi process [37, Chap. 6]. sGMRES motivates us to reevaluate techniques for building a nonorthogonal basis. For example, we can use k-partial orthogonalization as in (1.8). Provided the reduced matrix $\boldsymbol{AB}$ is reasonably conditioned, we can still obtain accurate solutions to the linear system via sGMRES (3.4).

4.3. The Lanczos recurrence. For this subsection, assume $\boldsymbol{A}$ is Hermitian. In this case, the Arnoldi process simplifies to a three-term recurrence [37, Sec. 6.6]:

$$\boldsymbol{q}_{j+1} = \boldsymbol{w}_{j+1}/\|\boldsymbol{w}_{j+1}\|_2 \quad \text{where} \quad \boldsymbol{w}_{j+1} = (\mathbf{I} - \boldsymbol{q}_j\boldsymbol{q}_j^* - \boldsymbol{q}_{j-1}\boldsymbol{q}_{j-1}^*)(\boldsymbol{A}\boldsymbol{q}_j).$$

The Lanczos basis $\boldsymbol{Q}_p = [\boldsymbol{q}_1, \dots, \boldsymbol{q}_p] \in \mathbb{C}^{n\times p}$ has the remarkable property that

$$\boldsymbol{A}\boldsymbol{Q}_p = \boldsymbol{Q}_p\boldsymbol{J}_p + \boldsymbol{w}_p\boldsymbol{e}_p^* \quad \text{where } \boldsymbol{J}_p \in \mathbb{C}^{p\times p} \text{ is tridiagonal.} \tag{4.1}$$

This allows us to solve the least-squares problem (3.2) with $\boldsymbol{B} = \boldsymbol{Q}_p$ in $O(p)$ time, and we construct the approximate solution $\boldsymbol{x}_{\boldsymbol{B}}$ with $O(np)$ arithmetic. This is how the MINRES algorithm operates [32].

For p iterations, the Lanczos recurrence costs just $O(np)$ operations, but it has complicated behavior in finite-precision arithmetic. This issue is not devastating when Lanczos is used to solve linear systems, but it can present a more serious challenge when solving eigenvalue problems [33, Chap. 13].

Although it is very efficient to solve the least-squares problem (3.2) by passing to the tridiagonal matrix $\boldsymbol{J}_p$, it is more reliable to sketch $\boldsymbol{S}(\boldsymbol{A}\boldsymbol{Q}_p)$ and to solve the sketched problem (3.4) instead. The approach based on sketching is competitive with MINRES when $p \ll n$.

The literature describes many approaches for maintaining the orthogonality of the Lanczos basis, such as selective orthogonalization [33, Chap. 13]. Sketching allows us to develop faster alternatives, such as omitting the extra orthogonalization. We can also use the sketched basis vectors to obtain coarse estimates for inner products [4]. Indeed, $(\boldsymbol{S}\boldsymbol{q}_i)^*(\boldsymbol{S}\boldsymbol{A}\boldsymbol{q}_j) \approx \boldsymbol{q}_i^*\boldsymbol{A}\boldsymbol{q}_j$ for all $i \le j$, so we can choose to orthogonalize $\boldsymbol{A}\boldsymbol{q}_j$ only against the basis vectors $\boldsymbol{q}_i$ where the inner product is nonnegligible.

4.4. The Chebyshev recurrence. In some settings, we may wish to avoid the orthogonalization steps entirely because they involve expensive inner products between basis vectors. We can achieve this goal by using other polynomial recurrences to construct a Krylov subspace basis. This idea is attributed to Joubert & Carey [23].

For simplicity, suppose that the spectrum of $\boldsymbol{A}$ is contained in the axis-aligned rectangle $[c \pm \delta_x, \pm\delta_y]$, and set $\varrho = \max\{\delta_x, \delta_y\}$. Then we can assemble a shifted-and-scaled Chebyshev basis $\boldsymbol{B} \in \mathbb{C}^{n\times p}$ via the following recurrence [25, 23]. Let $\boldsymbol{b}_1 = \boldsymbol{r}/\|\boldsymbol{r}\|_2$ and $\boldsymbol{b}_2 = (2\varrho)^{-1}(\boldsymbol{A} - c\mathbf{I})\boldsymbol{b}_1$. Then

$$\boldsymbol{b}_j = \frac{1}{\varrho}\left[(\boldsymbol{A} - c\mathbf{I})\boldsymbol{b}_{j-1} - \frac{\delta_x^2 - \delta_y^2}{4\varrho}\boldsymbol{b}_{j-2}\right] \quad \text{for } j = 3, \dots, p. \tag{4.2}$$

In practice, we also rescale each basis vector $\boldsymbol{b}_j$ to have unit ℓ_2 norm after it has played its role in the recurrence. The key theoretical fact is that the Chebyshev basis tends to have a condition number that grows *polynomially* in p, rather than exponentially. This claim depends on assumptions that the eigenvalues of the matrix are equidistributed over an ellipse [16, 25, 23, 34].

To implement this procedure, we may first apply a few iterations of the Arnoldi method (subsection 6.2) to estimate the spectrum of $\boldsymbol{A}$. More generally, we find a (transformed) ellipse that contains the spectrum. Then we adapt the Chebyshev polynomials to this ellipse [34]. The overall cost of constructing a Chebyshev basis for $\mathsf{K}_p(\boldsymbol{A}; \boldsymbol{r})$ is $O(np)$, and it involves no orthogonalization whatsoever.

4.5. Newton polynomials. The Newton polynomials provide another standard construction of a nonorthogonal basis for the Krylov subspace [34]. Suppose that

Algorithm 5.1 sGMRES + Arnoldi with k-partial orthogonalization

Input: Matrix $\boldsymbol{A} \in \mathbb{C}^{n\times n}$, right-hand side $\boldsymbol{f} \in \mathbb{C}^n$, initial guess $\boldsymbol{x} \in \mathbb{C}^n$, basis dimension d, number k of vectors for partial orthogonalization, stability tolerance $\texttt{tol} = O(u^{-1})$.
Output: Approximate solution $\hat{\boldsymbol{x}} \in \mathbb{C}^n$ to linear system (3.1) and estimated residual norm $\hat{r}_{\text{est}}$

function sGMRES
  Draw subspace embedding $\boldsymbol{S} \in \mathbb{C}^{s\times n}$ with $s = 2d + 1$ ▷ SRFT or sparse map
  Form residual and sketch: $\boldsymbol{r} = \boldsymbol{f} - \boldsymbol{A}\boldsymbol{x}$ and $\boldsymbol{g} = \boldsymbol{S}\boldsymbol{r}$
  Normalize basis vector $\boldsymbol{b}_1 = \boldsymbol{r}/\|\boldsymbol{r}\|_2$ and apply matrix $\boldsymbol{m}_1 = \boldsymbol{A}\boldsymbol{b}_1$
  for $j = 2, 3, 4, \dots, d$ **do**
    Partial Arnoldi: $\boldsymbol{w}_j = (\mathbf{I} - \boldsymbol{b}_{j-1}\boldsymbol{b}_{j-1}^* - \cdots - \boldsymbol{b}_{j-k}\boldsymbol{b}_{j-k}^*)\boldsymbol{m}_{j-1}$ ▷ $\boldsymbol{b}_{-i} = \mathbf{0}$ for $i \geq 0$
    Normalize basis vector $\boldsymbol{b}_j = \boldsymbol{w}_j/\|\boldsymbol{w}_j\|_2$ and apply matrix $\boldsymbol{m}_j = \boldsymbol{A}\boldsymbol{b}_j$
  Sketch reduced matrix: $\boldsymbol{C} = \boldsymbol{S}[\boldsymbol{m}_1, \dots, \boldsymbol{m}_d]$
  Thin QR factorization: $\boldsymbol{C} = \boldsymbol{U}\boldsymbol{T}$
  if $\kappa_2(\boldsymbol{T}) > \texttt{tol}$ **then** warning...
    Either whiten $\boldsymbol{B} \leftarrow \boldsymbol{B}/\boldsymbol{T}$, or compute new residual and restart (subsection 5.3)
  Solve least-squares problem: $\hat{\boldsymbol{y}} = \boldsymbol{T}\backslash(\boldsymbol{U}^*\boldsymbol{g})$
  Residual estimate: $\hat{r}_{\text{est}} = \|(\mathbf{I} - \boldsymbol{U}\boldsymbol{U}^*)\boldsymbol{g}\|_2$
  Construct solution: $\hat{\boldsymbol{x}} = \boldsymbol{x} + [\boldsymbol{m}_1, \dots, \boldsymbol{m}_j]\hat{\boldsymbol{y}}$

TABLE 1
GMRES versus sGMRES: Arithmetic. *This table compares the total arithmetic cost of solving an $n \times n$ linear system using a d-dimensional basis via standard GMRES and via sGMRES. For sGMRES, we consider both Arnoldi with k-partial orthogonalization and the Chebyshev basis. Heuristically, the parameters $k \ll d \ll n$. Constant factors are suppressed.*

	Matrix access	Form basis	Sketch	LS solve	Form soln.
Std. GMRES	dT_{matvec}	nd^2	—	d^2	nd
sGMRES-k	dT_{matvec}	ndk	$nd \log d$	d^3	nd
sGMRES-Cheb	dT_{matvec}	nd	$nd \log d$	d^3	nd

$\theta_1, \dots, \theta_p \in \mathbb{C}$ are complex-valued shift parameters. Then we can build a basis $\boldsymbol{B} \in \mathbb{C}^{n\times p}$ for $\mathsf{K}_p(\mathsf{A}; \boldsymbol{r})$ via the recurrence

$$\boldsymbol{b}_1 = \boldsymbol{r}/\|\boldsymbol{r}\|_2 \quad \text{and} \quad \boldsymbol{b}_j = (\boldsymbol{A} - \theta_{j-1}\mathbf{I})\boldsymbol{b}_{j-1} \quad \text{for } j = 2, \dots, p.$$

The shifts θ_i are often chosen to be estimated eigenvalues of $\boldsymbol{A}$, obtained from an invocation of the Arnoldi method (subsection 6.2). The overall computational profile of constructing the Newton basis is similar to constructing a Chebyshev basis.

4.6. Local orthogonalization. We can improve the conditioning of a computed basis $\boldsymbol{B} \in \mathbb{C}^{n\times d}$ by local orthogonalization. Indeed, it is generally helpful to orthogonalize subcollections of basis vectors, even if it proves too expensive to orthogonalize all of the basis vectors. In particular, scaling each column to have unit ℓ_2 norm is always appropriate. See Demmel's paper [14] for an analysis.

5. sGMRES algorithms. This section presents complete algorithms for solving the linear system (3.1) via sGMRES, including options for adaptive basis generation.

5.1. Basic implementation. Algorithm 5.1 contains pseudocode for a basic implementation of sGMRES using the k-partial Arnoldi basis (1.8). We recommend this version of the algorithm when the user lacks information about the spectrum of $\boldsymbol{A}$. Given bounds on the spectrum, one may replace the partial Arnoldi basis with a Chebyshev basis (subsection 4.4). See Table 1 for a summary of the arithmetic costs.

5.2. Iterative sGMRES. As noted, most methods for producing the Krylov subspace basis are recursive. They generate the columns of $\boldsymbol{B}$ and the reduced matrix $\boldsymbol{AB}$ in sequence. This observation suggests an iterative implementation of sGMRES. We sketch the columns of the reduced matrix as they arrive, incrementally solving the sGMRES problem (3.4) at each step.

Let $d_{\max}$ be the maximum allowable dimension of the Krylov subspace. Draw and fix a randomized subspace embedding $\boldsymbol{S} \in \mathbb{C}^{s\times n}$ with embedding dimension $s = 2d_{\max} + 1$. As we compute each column $\boldsymbol{Ab}_j$ of the reduced matrix, we can immediately form the sketch $\boldsymbol{S}(\boldsymbol{Ab}_j)$ and update the QR decomposition:

$$\boldsymbol{S}(\boldsymbol{AB}_j) = \boldsymbol{U}_j\boldsymbol{T}_j \quad \text{where} \quad \boldsymbol{B}_j = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_j]. \tag{5.1}$$

At each step, we obtain an approximate solution to the linear system:

$$\hat{\boldsymbol{y}}_j = \boldsymbol{T}_j^{\dagger}(\boldsymbol{U}_j^*(\boldsymbol{S}\boldsymbol{r}_0)) \quad \text{with} \quad \hat{r}_{\text{est},j} = \|(\mathbf{I} - \boldsymbol{U}_j\boldsymbol{U}_j)^*(\boldsymbol{S}\boldsymbol{r}_0)\|_2. \tag{5.2}$$

Repeat this process until the estimated residual norm $\hat{r}_{\text{est},j}$ is sufficiently small or we breach the threshold $d_{\max}$ for the size of the Krylov space. After d iterations, the arithmetic costs of (5.1) and (5.2) match the non-sequential implementation (subsection 3.3) with a basis of size d.

5.3. Adaptive restarting. There is a further opportunity to design an adaptive strategy for restarting. According to (2.5), the condition number $\kappa_2(\boldsymbol{T}_j)$ is comparable with $\kappa_2(\boldsymbol{AB}_j)$. When first $\kappa_2(\boldsymbol{T}_j) > \texttt{tol}$, we recognize that it is time to restart. We generate the new Krylov subspace using the *previous* residual $\hat{\boldsymbol{r}}_{j-1} = \boldsymbol{r}_0 - \boldsymbol{AB}_{j-1}\hat{\boldsymbol{y}}_{j-1}$. Alternatively, instead of restarting, we could approximately orthogonalize $\boldsymbol{B}$ by replacing it with $\boldsymbol{BT}^{-1}$, whose condition number is constant.

5.4. Storage-efficient versions. In situations where storage is at a premium, we can even avoid storing the reduced matrix $\boldsymbol{AB}_j$ by sketching its columns sequentially and discarding them immediately after sketching. Once the estimated residual norm $\hat{r}_{\text{est},j}$ is sufficiently small, we can construct the approximate solution

$$\hat{\boldsymbol{x}}_j = \boldsymbol{x}_0 + \boldsymbol{AB}_j\hat{\boldsymbol{y}}_j = \boldsymbol{x}_0 + \sum\nolimits_{i=1}^{j} (\boldsymbol{AB}_j)_i(\hat{\boldsymbol{y}}_j)_i$$

by iteratively regenerating the columns of the reduced matrix $\boldsymbol{AB}_j$ and linearly combining them on the fly. For some basis constructions (e.g., partial Arnoldi or Chebyshev), we only need to maintain a few columns of $\boldsymbol{B}$ and the j columns of $\boldsymbol{S}(\boldsymbol{AB}_j)$. Unfortunately, this modification doubles the arithmetic cost associated with basis generation (matvecs plus orthogonalization). A similar technique was used in [54].

5.5. Obtaining a solution with full accuracy. While the constant-factor loss in sGMRES is unlikely to be an issue, we can obtain a solution with the same quality as GMRES by using $\boldsymbol{T}$ as a preconditioner to solve (3.4) via an iterative method as in [36, 3]. This method still requires $\kappa_2(\boldsymbol{AB}) < u^{-1}$ to operate reliably.

6. The sketched Rayleigh–Ritz method. Let us turn to the nonsymmetric eigenvalue problem

$$\text{Find nonzero } \boldsymbol{x} \in \mathbb{C}^n \text{ and } \lambda \in \mathbb{C} : \quad \boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x} \quad \text{where } \boldsymbol{A} \in \mathbb{C}^{n\times n}. \tag{6.1}$$

We will provide an implementation and analysis of the sRR method outlined in subsection 1.3.1. Subsection 6.8 describes modifications for the symmetric eigenvalue problem. Section 7 covers techniques for constructing the basis for sRR.

6.1. The many faces of Rayleigh–Ritz. Let $\boldsymbol{B} \in \mathbb{C}^{n\times d}$ be a basis *with full rank*, and let $\boldsymbol{AB} \in \mathbb{C}^{n\times d}$ be the reduced matrix. Rayleigh–Ritz is best understood as a *Galerkin method* for computing eigenvalues [38, Sec. 4.3]. From the family $\boldsymbol{x} = \boldsymbol{By} \neq \boldsymbol{0}$, we seek a residual $\boldsymbol{r} = \boldsymbol{Ax} - \theta\boldsymbol{x}$ orthogonal to range($\boldsymbol{B}$). More precisely,

$$\text{Find nonzero } \boldsymbol{y} \in \mathbb{C}^d \text{ and } \theta \in \mathbb{C} : \qquad \boldsymbol{B}^*(\boldsymbol{ABy} - \theta\boldsymbol{By}) = \boldsymbol{0}. \tag{6.2}$$

Rearranging, we see that (6.2) can be posed as an ordinary eigenvalue problem:

$$\boldsymbol{M}_\star\boldsymbol{y} := \boldsymbol{B}^\dagger(\boldsymbol{AB})\boldsymbol{y} = \theta\boldsymbol{y} \quad \text{where } \boldsymbol{y} \neq \boldsymbol{0} \text{ and } \theta \in \mathbb{C}. \tag{6.3}$$

Recall that eigenvalue problems are invariant under similarity transforms. In the present context, the computed eigenpairs only depend on the range of $\boldsymbol{B}$, so they are invariant under the map $\boldsymbol{B} \leftarrow \boldsymbol{BT}$ for a nonsingular $\boldsymbol{T} \in \mathbb{C}^{d\times d}$. Therefore, if $\boldsymbol{Q} \in \mathbb{C}^{n\times d}$ is an orthonormal basis for range($\boldsymbol{B}$), then we may pass to

$$\boldsymbol{Q}^*(\boldsymbol{AQ})\boldsymbol{z} = \theta\boldsymbol{z} \quad \text{where } \boldsymbol{z} \neq \boldsymbol{0}. \tag{6.4}$$

Given a solution $(\boldsymbol{z}, \theta)$ to (6.4), we obtain an approximate eigenpair $(\boldsymbol{Qz}, \theta)$ of the matrix $\boldsymbol{A}$. This is the most typical presentation of RR.

In contrast, consider the problem of minimizing the residual over the subspace:

$$\text{minimize}_{\boldsymbol{y}\in\mathbb{C}^d, \theta\in\mathbb{C}} \quad \|\boldsymbol{ABy} - \theta\boldsymbol{By}\|_2 \quad \text{subject to } \|\boldsymbol{By}\|_2 = 1. \tag{6.5}$$

This formulation is sometimes called a *rectangular eigenvalue problem* [8, 22]. Let us emphasize that the RR method (6.3) does *not* solve the rectangular eigenvalue problem. Nevertheless, for any eigenpair $(\boldsymbol{y}_\star, \theta_\star)$ of the matrix $\boldsymbol{M}_\star$, it holds that

$$\|\boldsymbol{ABy}_\star - \theta_\star\boldsymbol{By}_\star\|_2 = \|(\boldsymbol{AB} - \boldsymbol{BM}_\star)\boldsymbol{y}_\star\|_2. \tag{6.6}$$

The matrix $\boldsymbol{M}_\star$ from (6.3) *does* solve a related variational problem [33, Thm. 11.4.2]:

$$\text{minimize}_{\boldsymbol{M}\in\mathbb{C}^{d\times d}} \quad \|\boldsymbol{AB} - \boldsymbol{BM}\|_{\mathrm{F}}. \tag{6.7}$$

These connections support the design and analysis of a sketched version of RR.

6.2. The Arnoldi method. The Arnoldi method is a classic algorithm [38, Sec. 6.2] for eigenvalue problems based on RR. First, it invokes the Arnoldi process (subsection 4.2) to build an orthonormal basis $\boldsymbol{Q} \in \mathbb{C}^{n\times d}$ for a Krylov subspace (generated by a random vector) at a cost of $O(nd^2)$ operations. This construction ensures that $\boldsymbol{Q}^*\boldsymbol{AQ}$ has upper Hessenberg form, so we can solve the eigenvalue problem (6.3) with $O(d^2)$ operations by means of the QR algorithm [38, Chap. 7]. Each eigenpair $(\boldsymbol{y}, \theta)$ of (6.3) induces an approximate eigenpair $(\boldsymbol{By}, \theta)$ of $\boldsymbol{A}$, which we can form with $O(nd)$ operations.

6.3. Derivation of sRR. We can view the sRR method as a sketched version of the matrix optimization problem (6.7). Consider a subspace embedding $\boldsymbol{S} \in \mathbb{C}^{s\times n}$ for range($[\boldsymbol{AB}, \boldsymbol{B}]$) with distortion $\varepsilon \in (0, 1)$. The sketched problem is

$$\text{minimize}_{\boldsymbol{M}\in\mathbb{C}^{d\times d}} \quad \|\boldsymbol{S}(\boldsymbol{AB} - \boldsymbol{BM})\|_{\mathrm{F}}. \tag{6.8}$$

The sRR method finds a solution $\hat{\boldsymbol{M}} \in \mathbb{C}^{d\times d}$ to this optimization problem. Then it poses the ordinary eigenvalue problem

$$\hat{\boldsymbol{M}}\boldsymbol{y} = \theta\boldsymbol{y} \quad \text{where } \boldsymbol{y} \neq \boldsymbol{0}. \tag{6.9}$$

This computation yields up to d eigenpairs $(\hat{\boldsymbol{y}}_i, \hat{\theta}_i)$ of the matrix $\hat{\boldsymbol{M}}$. We obtain approximate eigenpairs of $\boldsymbol{A}$ by the transformation $(\boldsymbol{B}\hat{\boldsymbol{y}}_i, \hat{\theta}_i)$.

Sketching allows us to obtain inexpensive *a posteriori* error bounds. For a computed eigenpair $(\hat{\boldsymbol{y}}, \hat{\theta})$ of $\hat{\boldsymbol{M}}$, it is cheap to form the sketched residual:

$$\hat{r}_{\text{est}} := \hat{r}_{\text{est}}(\hat{\boldsymbol{y}}, \hat{\theta}) := \|\boldsymbol{S}(\boldsymbol{AB}\hat{\boldsymbol{y}} - \hat{\theta}\boldsymbol{B}\hat{\boldsymbol{y}})\|_2 / \|\boldsymbol{SB}\hat{\boldsymbol{y}}\|_2. \tag{6.10}$$

By definition, the subspace embedding $\boldsymbol{S}$ ensures that the true residual satisfies

$$\frac{1-\varepsilon}{1+\varepsilon} \cdot \hat{r}_{\text{est}} \le \frac{\|\boldsymbol{AB}\hat{\boldsymbol{y}} - \hat{\theta}\boldsymbol{B}\hat{\boldsymbol{y}}\|_2}{\|\boldsymbol{B}\hat{\boldsymbol{y}}\|_2} \le \frac{1+\varepsilon}{1-\varepsilon} \cdot \hat{r}_{\text{est}}. \tag{6.11}$$

In other words, we can diagnose when the sRR method has (or has not) produced a high-quality approximate eigenpair $(\boldsymbol{B}\hat{\boldsymbol{y}}, \hat{\theta})$ of the original matrix $\boldsymbol{A}$.

6.4. Implementation of sRR. To implement sRR, we may use either an SRFT embedding (2.6) or a sparse embedding (2.7). We recommend the embedding dimension $s = 4d$, which typically results in distortion $\varepsilon = 1/\sqrt{2}$ for the range of $[\boldsymbol{AB}, \boldsymbol{B}]$.

We first sketch the reduced matrix ($\boldsymbol{SAB} \in \mathbb{C}^{s\times d}$) and the basis ($\boldsymbol{SB} \in \mathbb{C}^{s\times d}$) at a cost of $O(nd \log d)$ operations. Next, we compute a thin QR decomposition $\boldsymbol{SB} = \boldsymbol{UT}$ of the sketched basis. A minimizer of the sRR problem (6.8) is the matrix

$$\hat{\boldsymbol{M}} := (\boldsymbol{SB})^{\dagger}(\boldsymbol{SAB}) = \boldsymbol{T}^{-1}(\boldsymbol{U}^*(\boldsymbol{SAB})) \in \mathbb{C}^{d\times d}. \tag{6.12}$$

Afterward, we invoke the QR algorithm to solve the eigenvalue problem (6.9). Each of the last three steps costs $O(d^3)$ operations.

Given a computed eigenpair $(\hat{\boldsymbol{y}}, \hat{\theta})$, we can obtain the sketched residual value $\hat{r}_{\text{est}}(\hat{\boldsymbol{y}}, \hat{\theta})$ from (6.10) at a cost of $O(d^2)$ operations. If the residual estimate is sufficiently small, we declare that $(\boldsymbol{B}\hat{\boldsymbol{y}}, \hat{\theta})$ is an approximate eigenpair of $\boldsymbol{A}$. For maximum efficiency, we present the approximate eigenvector $\hat{\boldsymbol{x}} = \boldsymbol{B}\hat{\boldsymbol{y}} \in \mathbb{C}^n$ in factored form. If we need the full vector $\hat{\boldsymbol{x}}$, it costs $O(nd)$ operations. Ironically, if we extract a large number of explicit eigenvectors, this last step dominates the cost of the computation. Usually, the number of high-quality approximate eigenpairs is moderate.

In summary, given the basis $\boldsymbol{B}$, if we use sRR to solve (6.1), the cost of reporting the factored form of d approximate eigenpairs is $O(d^3 + nd \log d)$ operations. In contrast, RR requires $O(nd^2)$ arithmetic with an unstructured basis. Our numerical experience indicates that sRR is a robust alternative to RR so long as the condition number of the basis $\kappa_2(\boldsymbol{B}) \le 10^{14}$. This fact allows us to exploit fast basis constructions; see section 7. See Algorithm 6.1 for a simple implementation of sRR with a partial Arnoldi basis.

6.5. Stabilization. The output of sRR is almost identical to RR provided that $\kappa_2(\boldsymbol{B}) < u^{-1}$. This condition is very generous. In contrast, classical approaches demand that $\boldsymbol{B}$ to be nearly orthogonal. For example, it is known [33, Ch. 13] that the Lanczos algorithm produces accurate results when $\kappa_2(\boldsymbol{B}) < 1 + \sqrt{u}$.

If we see that the sketched basis $\boldsymbol{SB}$ is very badly conditioned ($\kappa_2(\boldsymbol{SB}) \gtrsim u^{-1}$), then the condition number diagnostic (2.5) implies that the basis $\boldsymbol{B}$ is also very badly conditioned. In this case, we can stabilize sRR by regularizing the basis. For example, we may compute the truncated SVD:

$$\boldsymbol{SB} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^* + \boldsymbol{E} \quad \text{where } \|\boldsymbol{\Sigma}\|_2 / \|\boldsymbol{E}\|_2 \le \texttt{tol}.$$

Algorithm 6.1 sRR + Arnoldi with k-partial orthogonalization

Input: Matrix $\boldsymbol{A} \in \mathbb{C}^{n\times n}$, initial vector $\boldsymbol{b} \in \mathbb{C}^n$, basis dimension d, number k of vector for partial orthogonalization, stability tolerance $\texttt{tol} = O(u^{-1})$, convergence tolerance τ.
Output: Approximate eigenpairs $(\boldsymbol{x}_i, \lambda_i)$ such that $\boldsymbol{A}\boldsymbol{x}_i \approx \lambda_i \boldsymbol{x}_i$ and estimated residual norms $\hat{r}_{\text{est},i}$.

function sRR
  Draw subspace embedding $\boldsymbol{S} \in \mathbb{C}^{s\times n}$ with $s = 4d_{\max}$ ▷ SRFT or sparse map
  Starting vector: $\boldsymbol{w}_1 = \texttt{randn}(n, 1)$
  Normalize basis vector $\boldsymbol{b}_1 = \boldsymbol{w}_1/\|\boldsymbol{w}_1\|_2$ and apply matrix $\boldsymbol{m}_1 = \boldsymbol{A}\boldsymbol{b}_1$
  for $j = 2, 3, 4, \dots, d$ **do**
    Partial Arnoldi: $\boldsymbol{w}_j = (\mathbf{I} - \boldsymbol{b}_{j-1}\boldsymbol{b}_{j-1}^* - \cdots - \boldsymbol{b}_{j-k}\boldsymbol{b}_{j-k}^*)\boldsymbol{m}_{j-1}$ ▷ $\boldsymbol{b}_{-i} = \boldsymbol{0}$ for $i \geq 0$
    Normalize $\boldsymbol{b}_j = \boldsymbol{w}_j/\|\boldsymbol{w}_j\|_2$ and apply matrix $\boldsymbol{m}_j = \boldsymbol{A}\boldsymbol{b}_j$
  Sketch basis $\boldsymbol{C} = \boldsymbol{S}[\boldsymbol{b}_1, \dots, \boldsymbol{b}_{d_{\max}}]$ and reduced matrix $\boldsymbol{D} = \boldsymbol{S}[\boldsymbol{m}_1, \dots, \boldsymbol{m}_{d_{\max}}]$.
  Thin QR factorization: $\boldsymbol{C} = \boldsymbol{U}\boldsymbol{T}$
  if $\kappa_2(\boldsymbol{T}) > \texttt{tol}$ **then** warning:
    Either whiten $\boldsymbol{B} \leftarrow \boldsymbol{B}/\boldsymbol{T}$, or stabilize and solve (6.13); see subsection 6.5
  Solve eigenvalue problem: $\boldsymbol{T}^{-1}\boldsymbol{U}^*\boldsymbol{D}\boldsymbol{y}_i = \lambda_i \boldsymbol{y}_i$ for $i = 1, \dots, d$
  Compute residual estimates $\|\boldsymbol{D}\boldsymbol{y}_i - \lambda_i \boldsymbol{C}\boldsymbol{y}_i\|_2/\|\boldsymbol{C}\boldsymbol{y}_i\|_2$. Let $\mathcal{I}$ be indices where residual is $< \tau$.
  Compute $\boldsymbol{x}_i = \boldsymbol{B}\boldsymbol{y}_i$ for $i \in \mathcal{I}$ and output $(\boldsymbol{x}_i, \lambda_i)$.

Then we use the QZ algorithm [18, §7.7] to solve the *generalized* eigenvalue problem[4]

$$(\boldsymbol{U}^*\boldsymbol{S}\boldsymbol{A}\boldsymbol{B})\boldsymbol{V}\boldsymbol{z} = \theta\boldsymbol{\Sigma}\boldsymbol{z}. \tag{6.13}$$

Each solution yields an sRR eigenpair $(\boldsymbol{V}\boldsymbol{z}, \theta)$ and an associated approximate eigenpair $(\boldsymbol{B}\boldsymbol{V}\boldsymbol{z}, \theta)$ of $\boldsymbol{A}$. The asymptotic cost is the same as the basic implementation.

6.6. Why does sRR work? We will argue that RR and sRR solve eigenvalue problems that are similar to a pair of nearby eigenvalue problems.

First, recall that $\boldsymbol{S}\boldsymbol{B} = \boldsymbol{U}\boldsymbol{T}$ is the QR decomposition of the sketched basis. Per (2.4), the whitened basis $\bar{\boldsymbol{B}} := \boldsymbol{B}\boldsymbol{T}^{-1}$ has conditioning $\kappa_2(\bar{\boldsymbol{B}}) \leq (1+\varepsilon)/(1-\varepsilon)$. Now, consider the variational problem (6.7) with respect to the whitened basis $\bar{\boldsymbol{B}}$ and its sketched version:

$$\begin{aligned} &\text{minimize}_{\boldsymbol{M}} \quad \|\boldsymbol{A}\bar{\boldsymbol{B}} - \bar{\boldsymbol{B}}\boldsymbol{M}\|_{\mathrm{F}} && \text{with solution } \bar{\boldsymbol{M}}_\star := \bar{\boldsymbol{B}}^\dagger \boldsymbol{A}\bar{\boldsymbol{B}}; \\ &\text{minimize}_{\boldsymbol{M}} \quad \|\boldsymbol{S}(\boldsymbol{A}\bar{\boldsymbol{B}} - \bar{\boldsymbol{B}}\boldsymbol{M})\|_{\mathrm{F}} && \text{with solution } \bar{\boldsymbol{M}} := (\boldsymbol{S}\bar{\boldsymbol{B}})^\dagger(\boldsymbol{S}\boldsymbol{A}\bar{\boldsymbol{B}}). \end{aligned}$$

Since $\boldsymbol{Q}$ is an orthonormal basis for range$(\bar{\boldsymbol{B}})$, we recognize that $\bar{\boldsymbol{M}}_\star$ is similar to the RR matrix $\boldsymbol{Q}^*\boldsymbol{A}\boldsymbol{Q}$. In view of (6.12) and the relations $\boldsymbol{S}\bar{\boldsymbol{B}} = \boldsymbol{S}\boldsymbol{B}\boldsymbol{T}^{-1} = \boldsymbol{U}$, the sketched solution is similar to the sRR matrix $\hat{\boldsymbol{M}}$:

$$\bar{\boldsymbol{M}} = \boldsymbol{U}^*(\boldsymbol{S}\boldsymbol{A}\boldsymbol{B})\boldsymbol{T}^{-1} = \boldsymbol{T}\hat{\boldsymbol{M}}\boldsymbol{T}^{-1}.$$

Eigenvalue problems are invariant under similarity, so it suffices to show $\bar{\boldsymbol{M}} \approx \bar{\boldsymbol{M}}_\star$.

To that end, we invoke (2.3) columnwise to obtain the comparison

$$\|\boldsymbol{A}\bar{\boldsymbol{B}} - \bar{\boldsymbol{B}}\bar{\boldsymbol{M}}_\star\|_{\mathrm{F}} \leq \|\boldsymbol{A}\bar{\boldsymbol{B}} - \bar{\boldsymbol{B}}\bar{\boldsymbol{M}}\|_{\mathrm{F}} \leq \frac{1+\varepsilon}{1-\varepsilon} \cdot \|\boldsymbol{A}\bar{\boldsymbol{B}} - \bar{\boldsymbol{B}}\bar{\boldsymbol{M}}_\star\|_{\mathrm{F}}. \tag{6.14}$$

Using the definitions of $\bar{\boldsymbol{M}}_\star$ and $\boldsymbol{Q}$, we find that

$$\|\boldsymbol{A}\bar{\boldsymbol{B}} - \bar{\boldsymbol{B}}\bar{\boldsymbol{M}}_\star\|_{\mathrm{F}} = \|(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^*)\boldsymbol{A}\bar{\boldsymbol{B}}\|_{\mathrm{F}} \leq \sigma_{\max}(\bar{\boldsymbol{B}}) \cdot \|(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^*)\boldsymbol{A}\boldsymbol{Q}\|_{\mathrm{F}}.$$

[4] When $\kappa_2(\boldsymbol{B}) \gtrsim u^{-1}$, numerical experiments suggest this approach is more stable than reducing to a standard eigenvalue problem as in (6.12).

From the last two displays, a short argument using the triangle inequality and the conditioning of the whitened basis produces

$$\|\bar{\boldsymbol{M}} - \bar{\boldsymbol{M}}_\star\|_{\mathrm{F}} \leq \frac{1}{\sigma_{\min}(\bar{\boldsymbol{B}})} \cdot \|(\bar{\boldsymbol{M}} - \bar{\boldsymbol{M}}_\star)\bar{\boldsymbol{B}}\|_{\mathrm{F}} \leq \frac{2(1+\varepsilon)}{(1-\varepsilon)^2} \cdot \|(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^*)\boldsymbol{A}\boldsymbol{Q}\|_{\mathrm{F}}.$$

Here is an interpretation. If $\operatorname{range}(\boldsymbol{Q}) = \operatorname{range}(\boldsymbol{B})$ is close to an invariant subspace of $\boldsymbol{A}$, then $(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^*)\boldsymbol{A}\boldsymbol{Q} \approx \mathbf{0}$. In this case, $\bar{\boldsymbol{M}} \approx \bar{\boldsymbol{M}}_\star$. Therefore, sRR and RR solve nearby eigenvalue problems, and we deduce that sRR is a backward stable approximation to RR in exact arithmetic.

More generally, as long as $\operatorname{range}(\boldsymbol{B})$ contains an approximate eigenvector of $\boldsymbol{A}$ with small residual, (6.11) shows that the same vector yields a comparably small residual for the sketched eigenproblem (6.9). Unfortunately, even in this case, there is no guarantee that sRR will find an approximate eigenpair with a small residual. Indeed, the behavior of classic RR is already complicated, with pathological examples [45, p. 282]. Nevertheless, RR is known to provide excellent outputs in the vast majority of cases; see [30, 38, 44] for the analysis.

6.7. sRR with a Krylov subspace basis. When $\boldsymbol{B}$ is a (graded) basis for a Krylov subspace, the analysis of sRR simplifies further. In this case, the solutions $\boldsymbol{M}_\star$ and $\hat{\boldsymbol{M}}$ to (6.7) and (6.8) differ only in the final column! To verify this point, observe that (6.7) decouples into a family of d least-squares problems:

$$\text{minimize}_{\boldsymbol{m}_i \in \mathbb{C}^d} \quad \|\boldsymbol{A}\boldsymbol{b}_i - \boldsymbol{B}\boldsymbol{m}_i\|_{\mathrm{F}} \quad \text{for } i = 1, \dots, d$$

By construction, the vector $\boldsymbol{A}\boldsymbol{b}_i$ lies in the span of $\boldsymbol{B}$ for $i = 1, \dots, d-1$. Each of these problems has a unique solution with zero residual. Thus, the sketched problem (6.8) correctly identifies the exact solution. This fact illuminates the power of sRR.

6.8. The symmetric case. Consider the symmetric eigenvalue problem

$$\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x} \quad \text{where } \boldsymbol{A} = \boldsymbol{A}^*. \tag{6.15}$$

We can apply sRR directly to (6.15). Unfortunately, sRR is not guaranteed to (and in fact does not always) return real eigenvalue estimates. At root, the sketched eigenvalue problem (6.9) is not (similar to) a symmetric problem. Accordingly, the computed eigenvectors need not be orthogonal. This is an inherent drawback.

Fortunately, for (6.15), sRR often computes eigenvalue estimates that are real (or nearly real), and the associated eigenvectors tend to be nearly orthogonal. We can anticipate this outcome when the whitened matrices satisfy $\bar{\boldsymbol{M}} \approx \bar{\boldsymbol{M}}_\star$. Indeed, the eigenvalues of a symmetric matrix are well-conditioned under nonsymmetric perturbations [24, 46]. As for the eigenvectors, if two approximate eigenpairs $(\hat{\boldsymbol{x}}_1, \hat{\lambda}_1)$ and $(\hat{\boldsymbol{x}}_2, \hat{\lambda}_2)$ of a symmetric matrix $\boldsymbol{A}$ have small residuals and sufficient gap $|\hat{\lambda}_1 - \hat{\lambda}_2|$, then it follows that $\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2$ are nearly orthogonal [33]. This forces the high-quality eigenvectors computed by sRR to be nearly orthogonal.

For a real symmetric matrix $\boldsymbol{A}$, our implementation of sRR simply extracts real part of the computed eigenvalues and eigenvectors. When $\boldsymbol{A}$ is complex Hermitian, we force the eigenvalues to be real (but not eigenvectors). The design of a fast algorithm that respects symmetry remains an open problem.

7. Constructing a basis for sRR. The performance of RR and sRR depends on the quality of the basis construction. For these problems, it is natural to consider *block* Krylov bases generated by random vectors. As before, we can consider nonorthogonal basis constructions. Owing to the overlap with the discussion of single-vector Krylov spaces, our presentation here is more telegraphic.

7.1. Block Krylov subspaces. For the eigenvalue problem (6.1), we can search for solutions using sRR with a block Krylov subspace. Let $\boldsymbol{\Omega} \in \mathbb{C}^{n\times b}$ be an initial matrix; the dimension b is called the *block size*. Define

$$\mathsf{K}_p(\boldsymbol{A}; \boldsymbol{\Omega}) := \operatorname{span}\{\boldsymbol{\Omega}, \boldsymbol{A\Omega}, \dots, \boldsymbol{A}^{p-1}\boldsymbol{\Omega}\} = \operatorname{span}\{\varphi(\boldsymbol{A})\boldsymbol{\Omega} : \deg(\varphi) \le p-1\}.$$

Setting $d = bp$, we can express a basis $\boldsymbol{B} \in \mathbb{C}^{n\times d}$ for this subspace in the form

$$\boldsymbol{B} = [\boldsymbol{B}_1, \dots, \boldsymbol{B}_p] = [\varphi_1(\boldsymbol{A})\boldsymbol{\Omega}, \dots, \varphi_p(\boldsymbol{A})\boldsymbol{\Omega}].$$

For eigenvalue problems, the generating matrix $\boldsymbol{\Omega} \in \mathbb{C}^{n\times b}$ may be drawn at random from a standard normal distribution.[5]

Historically, the NLA literature has prescribed a small block size, say $b \le 4$, and a large depth p. More recent research [26, Sec. 11] has identified an opportunity to use a large block size b, say 10s or 100s, with a much smaller depth, say $p \le 10$. This shift in perspective has already transformed the computational profile of block Krylov subspaces, especially in modern computing environments. For instance, we can parallelize the computation over the columns of $\boldsymbol{\Omega}$ (or over the filter polynomials φ_i). In combination with sRR, nonorthogonal basis constructions promise further benefits. See [28, 49, 48] for theoretical analysis of block Krylov subspaces for low-rank matrix approximation and symmetric eigenvalue problems.

Remark 7.1 (Other kinds of bases). There are other subspace projection methods for solving eigenvalue problems that use bases other than Krylov subspaces. For example, the Jacobi–Davidson method and the LOBPCG algorithm uses alternative ideas to build a search space. These methods may also be combined with sRR.

7.2. Basis diagnostics and restarting. As with single-vector Krylov subspaces, we can sketch basis vectors as they are generated to collect summary information about the quality of the basis. Indeed, if $\boldsymbol{B} \in \mathbb{C}^{n\times d}$ is a basis, then the condition number of the sketched basis $\kappa_2(\boldsymbol{SB})$ serves as a proxy for $\kappa_2(\boldsymbol{B})$; see (2.5). When the basis is poor, it can be important to use stabilized sRR (subsection 6.5).

It can also be effective to restart production of the Krylov subspace when the quality of the basis starts to decline. For example, we may compute a basis for the Krylov subspace $\mathsf{K}_p(\boldsymbol{A}; \boldsymbol{\Omega})$, and we can feed this basis to sRR to extract a matrix $\boldsymbol{X}$ whose columns approximately span the desired invariant subspace of $\boldsymbol{A}$. Then we pass to the Krylov subspace $\mathsf{K}_p(\boldsymbol{A}; \boldsymbol{X})$, and so forth. Randomized subspace iteration [20] is a simple version of this technique.

Another possibility for restarting is to deflate converged eigenpairs by working in their orthogonal complement. Convergence of Ritz pairs and loss of orthogonality are known to be tightly linked [33, Ch. 11]. Felicitously, sRR is able to identify such eigenpairs cheaply. Optimizing the sRR restarting strategy is left as future work.

7.3. Block monomial basis with orthogonalization. Although the monomial basis is anathema for large-degree polynomials, we can still use it for shallow Krylov subspaces (say, when $p < 5$). In this case, we can assemble a basis $\boldsymbol{B} = [\boldsymbol{B}_1, \dots, \boldsymbol{B}_p]$ for $\mathsf{K}_p(\boldsymbol{A}; \boldsymbol{\Omega})$ as follows. Set $\boldsymbol{B}_1 = \texttt{orth}(\boldsymbol{\Omega})$, and iterate

$$\boldsymbol{B}_j = \texttt{orth}(\boldsymbol{\Omega}_j) \quad \text{where} \quad \boldsymbol{\Omega}_j = \boldsymbol{AB}_{j-1} \quad \text{for } j = 2, 3, \dots, p.$$

We acquire the reduced matrix $\boldsymbol{AB}$ as a by-product of this computation.

[5] In this context, we do not derive much computational benefit from fancier nonadaptive distributions, such as SRFTs or sparse embeddings.

The block monomial basis has been used in the "blanczos" method [35, 19, 28, 48, 26] for low-rank matrix approximation, but it requires an expensive full orthogonalization of $\boldsymbol{B}$ in the final step. When used as an input to sRR, it may not be necessary to reorthogonalize the block monomial basis $\boldsymbol{B}$.

7.4. Block Arnoldi with partial orthogonalization. We can mitigate the rapid condition number growth of the block monomial basis by adding extra orthogonalization steps. For recurrence length $k \in \mathbb{N}$, we set $\boldsymbol{B}_1 = \mathtt{orth}(\boldsymbol{\Omega})$ and iterate

$$\boldsymbol{B}_j = \mathtt{orth}(\boldsymbol{\Omega}_j) \quad \text{where} \quad \boldsymbol{\Omega}_j = (\mathbf{I} - \boldsymbol{B}_{j-1}\boldsymbol{B}_{j-1}^* - \cdots - \boldsymbol{B}_{j-k}\boldsymbol{B}_{j-k}^*)(\boldsymbol{A}\boldsymbol{B}_{j-1}).$$

The resulting basis $\boldsymbol{B} = [\boldsymbol{B}_1, \dots, \boldsymbol{B}_p]$ serves as an input to sRR. The choice $k = 1$ or $k = 2$ already improves substantially over the block monomial basis.

When $\boldsymbol{A}$ is Hermitian, the choice $k = 2$ corresponds to the block Lanczos method without reorthogonalization [33, Chap. 13]. Historically, the reorthogonalization step has been regarded as important for achieving robustness. If we use block Lanczos with sRR, then we can often dispense with reorthogonalization.

As with sGMRES, in the unblocked case ($b = 1$), we recommend partial orthogonalization with a modest k as shown in Algorithm 6.1. However, for eigenvalue computations, there is compelling reason to take the block size $b \gg 1$. As the block size b increases, partial orthogonalization quickly becomes cumbersome.

7.5. Block Chebyshev recurrence. When computing block Krylov subspaces, the cost of (partial) orthogonalization can be devastating. By employing other polynomial recurrences, we can potentially *eliminate* all orthogonalization steps involving vectors of length n. In particular, the shifted-and-scaled Chebyshev recurrence emerges as an appealing option.

Suppose that we have prior knowledge that the spectrum of $\boldsymbol{A}$ is contained in the axis-aligned rectangle $[c \pm \delta_x, \pm\delta_y]$, and set $\varrho = \max\{\delta_x, \delta_y\}$. Then we can form a block Chebyshev basis $\boldsymbol{B} = [\boldsymbol{B}_1, \dots, \boldsymbol{B}_p]$ for $\mathsf{K}_p(\boldsymbol{A}; \boldsymbol{\Omega})$ as follows.

$$\boldsymbol{B}_1 = \boldsymbol{\Omega}; \qquad \boldsymbol{B}_2 = \frac{1}{2\varrho}(\boldsymbol{A} - c\mathbf{I})\boldsymbol{\Omega}; \qquad \boldsymbol{B}_j = \frac{1}{\varrho}\left[(\boldsymbol{A} - c\mathbf{I})\boldsymbol{B}_{j-1} - \frac{\delta_x^2 - \delta_y^2}{4\varrho}\boldsymbol{B}_{j-2}\right].$$

To implement this procedure, we typically need to perform a coarse initial eigenvalue computation (using sRR + block Arnoldi) to obtain a rough estimate for the spectrum of $\boldsymbol{A}$. For this purpose, a small block size b and depth p usually suffice. We may also consider Chebyshev polynomials based on rotated ellipses, as in [34].

A remarkable feature of this approach is that we can compute the block Chebyshev basis for $\mathsf{K}_p(\boldsymbol{A}; \boldsymbol{\Omega})$ with $b(p-1)$ matvecs plus $O(nbp)$ operations. In contrast, it requires $O(n(bp)^2)$ extra operations to produce a fully orthogonal basis. Beyond that, the Chebyshev recurrence can be implemented efficiently in parallel or with SIMD processors, and the lack of inner products and orthogonalization steps allows us to evade communication and synchronization costs.

8. Computational experiments. This section presents some numerical examples that showcase the potential of sGMRES and sRR for solving large linear systems and eigenvalue problems. All computations were performed in MATLAB version 2020a on a workstation with 256GB memory and 96 cores, each clocked at 3.3 GHz.

8.1. Solving linear systems with sGMRES. This subsection applies the sGMRES method to solve symmetric and nonsymmetric linear systems.

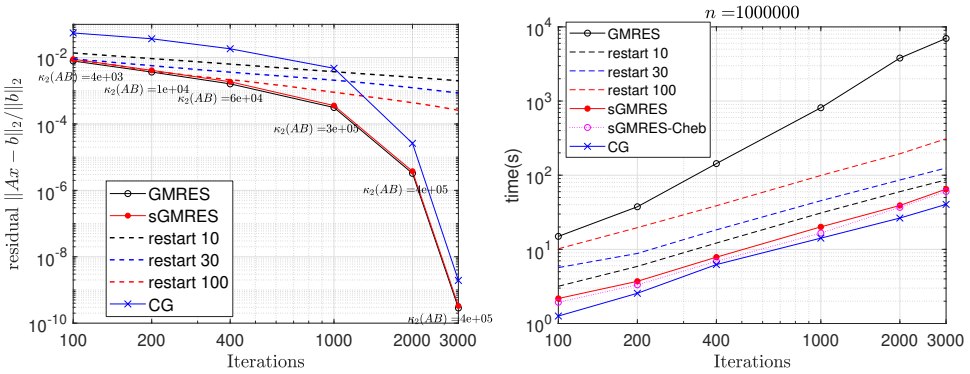

FIG. 3. ***sGMRES versus GMRES: Laplacian system.*** *These panels compare the performance of MATLAB* `gmres` *(with and without restarting) against the sGMRES algorithm (with 2-partial orthogonalization or the Chebyshev basis). The sparse linear system* $\boldsymbol{Ax} = \boldsymbol{f}$ *involves a 2D Laplacian matrix with dimension* $n = 10^6$. ***Left:*** *Relative residual and conditioning* $\kappa_2(\boldsymbol{AB})$ *of the reduced matrix associated with the partial Arnoldi basis.* ***Right:*** *Total runtime including basis generation.*

8.1.1. Algorithm details. Our main implementation of sGMRES follows the pseudocode in Algorithm 5.1 with minor changes to improve numerical stability. In particular, we construct a basis using k-partial orthogonalization with small values $k \in \{2, 4\}$, unless otherwise noted. In one example, we consider a Chebyshev basis, as described in subsection 4.4. We do not whiten the basis or restart sGMRES. The subspace embedding is based on an SRFT matrix (2.6) where $\boldsymbol{F}$ is a discrete cosine transform (DCT2), so our sketch uses $O(nd \log n)$ operations rather than $O(nd \log d)$.

We do not report tests involving the more elaborate algorithms discussed in section 5, because fine-tuning for optimal performance is outside the scope of this exploratory research.

8.1.2. A nonsymmetric linear system. This subsection offers details about solving the nonsymmetric linear system, documented in Figure 1 of the introduction. The matrix $\boldsymbol{A}$ is the sparse instance `t2em` with dimension $n = 921,632$ from the SuiteSparse Matrix Collection [13]. The right-hand side is a random vector drawn from the standard normal distribution. We compare sGMRES with the MATLAB command `gmres` without restarting and with restarting frequencies $\{10, 30, 100\}$. Observe that the k-partial Arnoldi basis leads to a reduced matrix $\boldsymbol{AB}$ whose condition number grows quickly, but the condition number remains below the tolerance u^{-1} throughout the computation. This property ensures that sGMRES is effective.

8.1.3. A symmetric linear system. We consider a symmetric test matrix $\boldsymbol{A}$ with dimension $n = 10^6$, obtained by discretizing the 2D Laplacian.[6] The matrix is positive semidefinite with kernel $\mathbf{e} = [1, 1, \dots, 1]^*$. We solve the Poisson problem $\boldsymbol{Ax} = \boldsymbol{f}$ where the right-hand side $\boldsymbol{f}$ is a standard normal random vector in the orthogonal complement of $\mathbf{e}$. For this problem, we would prefer CG over GMRES.

Figure 3 describes the progress of sGMRES where the basis is generated by k-partial orthogonalization for $k = 2$ and where the basis is generated by the Chebyshev recurrence (more below). We compare with the MATLAB commands `pcg` and `gmres` without restart and with restart frequencies in $\{10, 30, 100\}$. For all methods, the

[6] To generate the matrix we used the code in https://www.mathworks.com/matlabcentral/fileexchange/27279-laplacian-in-1d-2d-or-3d.

initial solution $\boldsymbol{x}_0 = \mathbf{0}$.

For both versions of sGMRES, the cost of d iterations is 10% to 20% slower than d iterations of CG. Nevertheless, for the same number d of iterations, sGMRES achieves ℓ_2 residual norms that are about $5\times$ smaller than CG. According to this metric, the sGMRES method is more efficient than CG. As we saw for the nonsymmetric problem, sGMRES is up to $6\times$ faster than the restarted versions of GMRES, which do not converge to high accuracy. Meanwhile, sGMRES achieves the same accuracy as GMRES, but the sketched version is up to $100\times$ faster after 3000 iterations.

The Laplacian matrix is a natural candidate for testing the Chebyshev basis because we have prior knowledge about the spectrum. We use the fact that the eigenvalues are real numbers in the interval $[0, 8]$ to select the parameters for the Chebyshev recurrence (subsection 4.4). The Chebyshev basis construction is slightly faster than the k-partial Arnoldi construction because it requires no inner products or orthogonalization steps. Even so, the quality of the Chebyshev basis is decent; after 3000 iterations, the reduced matrix has condition number $\kappa_2(\boldsymbol{AB}) \approx 10^8$, which is good enough for sGMRES to succeed. Figure 3 shows that the k-partial Arnoldi basis is still better conditioned. This experiment is intriguing because the Chebyshev basis can offer dramatic benefits in parallel computing environments [23, 34, 5, 9].

8.2. sGMRES: Hard examples. It is important to acknowledge that the sGMRES method is not always an effective tool for linear systems. In some cases, sGMRES inherits its weaknesses from GMRES, but there are also new phenomena that arise.

First, there are linear systems where classic GMRES cannot produce a small residual because the Krylov subspace does not have sufficient approximation power. sGMRES cannot cure this debility. In these cases, preconditioning is critical.

Second, sGMRES is not especially useful for problems where the matrix–vector multiply $\boldsymbol{x} \mapsto \boldsymbol{Ax}$ is costly relative to the other arithmetic. For example, when $\boldsymbol{A}$ is dense, over 99% of the runtime of GMRES or sGMRES may be devoted to matvecs.

Third, and most seriously, there are linear systems where it is very difficult to construct a numerically full-rank basis for the Krylov subspace without meticulous orthogonalization. The rest of this subsection documents one such problem instance.

Consider the matrix `FS 680 1` from Matrix Market, which is known to instigate Krylov bases with bad behavior [34, Table 2]. In this case, the basis $\boldsymbol{B}$ and the reduced matrix $\boldsymbol{AB}$ and their sketches $\boldsymbol{SB}$ and $\boldsymbol{SAB}$ have rapidly increasing condition number. Once $\kappa_2(\boldsymbol{SAB}) > u^{-1}$, numerical errors can cause sGMRES to fail, even when GMRES is successful. See Figure 4 for an illustration, which shows that increasing the extent k of partial orthogonalization does not help.

We can always monitor the conditioning of the reduced matrix $\boldsymbol{AB}$ inexpensively by means of its sketch $\boldsymbol{SAB}$. Unfortunately, we are not aware of a reliable mechanism for controlling the conditioning, short of full orthogonalization. Indeed, k-partial orthogonalization does not even guarantee monotone decrease of the condition number as k increases. This issue remains a major challenge for sGMRES. The ideas from [4] may be useful here.

8.3. Solving eigenvalue problems with sRR. This subsection studies the performance of sRR for solving nonsymmetric and symmetric eigenvalue problems.

8.3.1. Algorithm details. Our implementation of sRR follows the pseudocode in Algorithm 6.1 with minor changes to facilitate comparison with the MATLAB `eigs` command. In particular, we focus on a single-vector Krylov subspace ($b = 1$), and we use k-partial orthogonalization to form the basis. In one example, we consider a

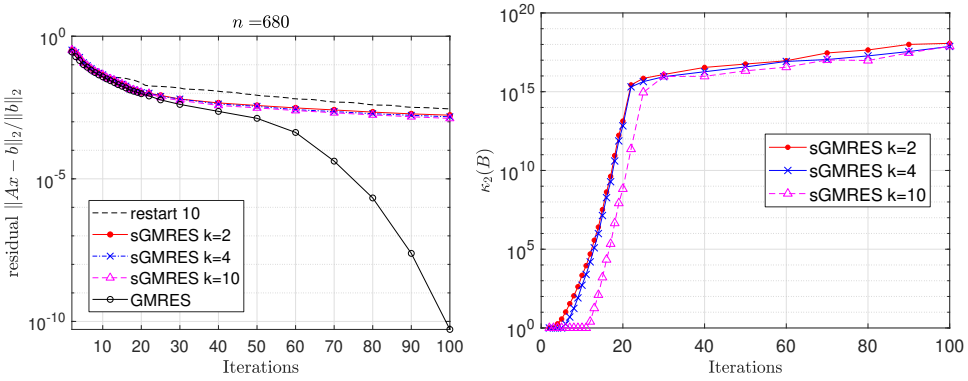

FIG. 4. ***sGMRES: Hard problems.*** *For some linear systems, it is expensive to construct a well-conditioned basis* $\boldsymbol{B}$ *for the Krylov subspace. When* $\kappa_2(\boldsymbol{AB}) > u^{-1}$*, the sGMRES algorithm may fail to match the GMRES algorithm.* ***Left:*** *Relative residual norms.* ***Right:*** *Conditioning* $\kappa_2(\boldsymbol{B})$ *of the k-partial Arnoldi basis. When* $\kappa_2(\boldsymbol{B}) > u^{-1}$*, the reported values are unreliable.*

block Krylov subspace with a Chebyshev basis, as described in subsection 7.5. We do not use restarting or stabilization, except as noted. The subspace embedding is based on an SRFT matrix (2.6) where $\boldsymbol{F}$ is a discrete cosine transform (DCT2).

When using `eigs`, we set the option `opts.p=r; opts.maxit=1;` to suppress restart. We set the flag `opts.issym` to reflect whether the problem is symmetric.

For eigenvalue computations, we anticipate that block Krylov subspaces can yield significant advantages over single-vector Krylov subspaces. Some of these improvements derive from higher-order BLAS. We can also take advantage of SIMD architectures, and we can reduce costs of communication and synchronization in parallel computing environments.

8.3.2. Nonsymmetric eigenvalue problems. This section describes the nonsymmetric eigenvalue problem that forms the basis for Figure 2. This computation is modeled on the trust-region subproblem (TRS) from optimization [12]:

$$\text{minimize}_{\boldsymbol{x}\in\mathbb{R}^n} \quad \frac{1}{2}\boldsymbol{x}^*\boldsymbol{A}\boldsymbol{x} + \boldsymbol{g}^*\boldsymbol{x} \quad \text{subject to} \quad \|\boldsymbol{x}\|_2 \le \Delta. \tag{8.1}$$

This quadratic program can be reduced to a nonsymmetric eigenvalue problem [1]:

$$\begin{bmatrix} \boldsymbol{A} & \Delta^{-2}\boldsymbol{g}\boldsymbol{g}^* \\ -\mathbf{I} & \boldsymbol{A} \end{bmatrix} \boldsymbol{x} = \lambda \boldsymbol{x}. \tag{8.2}$$

To obtain a solution to (8.1), we extract a (scaled) eigenvector of (8.2) corresponding to the right-most eigenvalue (which must be real).

We consider an instance of (8.2) where $\boldsymbol{A}$ is an $n \times n$ tridiagonal matrix with equispaced values in $[-1, 1]$ on the main diagonal and with 1s on the off-diagonals. In this case, the block matrix admits a fast matrix–vector multiplication operation. The vector $\boldsymbol{g} \in \mathbb{R}^n$ is drawn from the standard normal distribution and scaled so that $\|\boldsymbol{g}\|_2 = 0.01$. The constraint value $\Delta = 1$.

We solved (8.2) using sRR, as described in subsection 8.3.1. The Krylov subspace was generated from the initial vector $\mathbf{0} \oplus \boldsymbol{g}$. The results appear in Figure 2. The modest loss of accuracy in sRR after 1500 iterations can be remedied by using the stabilization process (subsection 6.5), which approximately doubles the runtime.

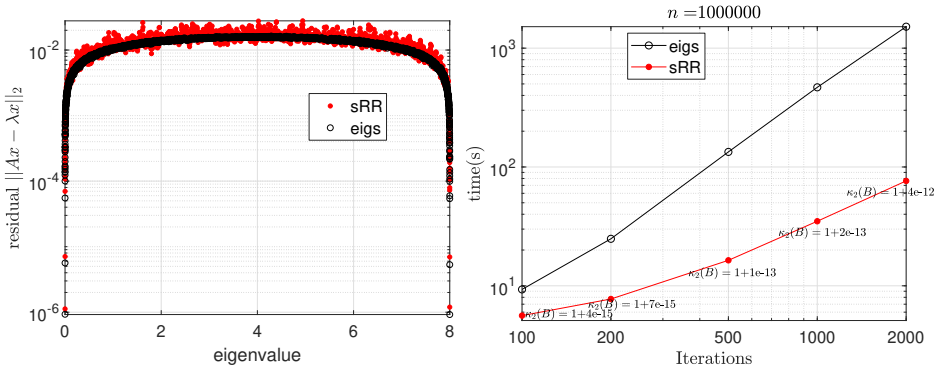

FIG. 5. ***sRR versus RR: Symmetric eigenvalue problem.*** *These panels compare the performance of MATLAB* `eigs` *(without restarting) against the sRR algorithm (where the basis* $\boldsymbol{B}$ *is computed by the Lanczos recurrence). The sparse, symmetric eigenvalue problem* $\boldsymbol{Ax} = \lambda\boldsymbol{x}$ *has dimension* $n = 10^6$*, and it arises from a 2D Laplacian.* ***Left:*** *Relative residuals as a function of computed eigenvalues after 2000 iterations.* ***Right:*** *Total runtime, including basis generation, and condition* $\kappa_2(\boldsymbol{B})$ *of the basis.*

8.3.3. Symmetric eigenvalue problems. Next, we present an example of a symmetric eigenvalue problem. The matrix $\boldsymbol{A}$ is the 2D Laplacian matrix with dimension $n = 10^6$ that was introduced in subsection 8.1.3. The initial vector for the Krylov subspace is drawn from the standard normal distribution, and it is shared between sRR and `eigs`. We use the Lanczos recurrence (i.e., 2-partial orthogonalization) to construct the subspace basis. For sRR, we report the real parts of the eigenvalues and eigenvectors, as discussed in subsection 6.8.

Figure 5 displays the results of the experiment. We see that sRR identifies the same eigenvalues as RR, and the sRR residual norms are within a small factor of the RR residual norms. For 2000 iterations, sRR runs 12× faster than `eigs`. The difference is likely because `eigs` enforces orthogonality to ensure that the Lanczos method remains robust. In this instance, the Lanczos method constructs a basis that is almost orthogonal, but sRR does not require this outcome to succeed.

When the Lanczos method is used to reduce the matrix to (partial) tridiagonal form, it is critical that the Lanczos basis remain almost perfectly orthogonal. Loss of orthogonality of the basis leads to *ghost eigenvalues*, which are repeated estimates of a single eigenvalue [15, Ch. 7]. (Selective) orthogonalization is a traditional remedy [42, 43], but it can be costly. In our experience, sRR rarely produces ghost eigenvalues because it does not need the basis to reduce the matrix to tridiagonal form.

8.3.4. Poorly conditioned bases and stabilization. When the computed basis $\boldsymbol{B}$ is ill-conditioned, sRR may not produce reliable eigenvalues estimates. We can stanch this loss by stabilization, as discussed in subsection 6.5.

Without loss of generality, we consider a diagonal matrix $\boldsymbol{A}$. The dimension $n = 2^{19}$, and the eigenvalues are ten randomly spaced numbers in $[-1, -0.1]$, along with $2^{19} - 10$ equispaced numbers in $[0, 1]$. Via the block Chebyshev recurrence (subsection 7.5), we construct a (nonorthogonal) basis $\boldsymbol{B} \in \mathbb{R}^{n\times(bp)}$ for the block Krylov subspace with block size $b = 100$ and increasing depth p. We then use classic RR, sRR, and sRRstab (the stabilized version) to compute eigenpairs of $\boldsymbol{A}$.

Figure 6 displays the results. The condition number $\kappa_2(\boldsymbol{B})$ of the basis grows with the depth p of the block Krylov subspace. Even so, sRR computes accurate eigenpairs

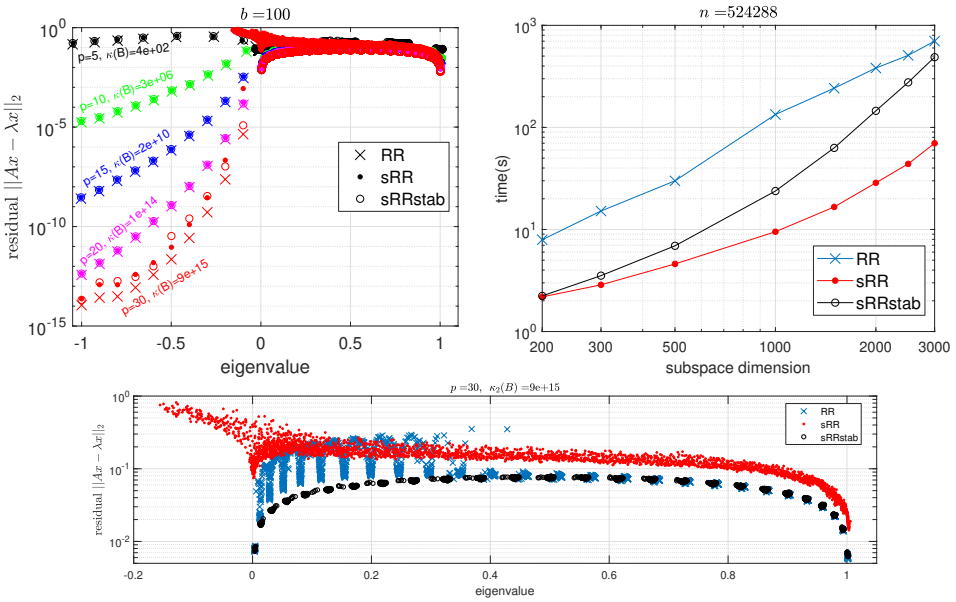

FIG. 6. ***sRR: Ill-conditioned bases and stabilization.*** *This diagram shows how eigenvalue residuals improve with the depth p of a block Krylov–Chebyshev subspace $\boldsymbol{B} \in \mathbb{R}^{n\times(bp)}$ with block size $b = 100$.* ***Left:*** *Residuals as a function of eigenvalue estimates, along with the condition $\kappa_2(\boldsymbol{B})$ of the basis.* ***Right:*** *Runtime for eigenvalue extraction, excluding basis generation.* ***Bottom:*** *Magnification of the left panel for $p = 30$ to illustrate (interior) eigenvalue estimates in $[0, 1]$.*

while $\kappa_2(\boldsymbol{B}) \lesssim u^{-1}$. When the condition number of the basis is larger, sRRstab produces more accurate estimates for both the extremal and interior eigenpairs of the matrix $\boldsymbol{A}$ than RR. Excluding the cost of basis generation, sRR runs up to 15× faster than classic RR. The stabilized algorithm also runs somewhat faster than RR.

9. Variations and extensions. The ideas underlying sRR can be adapted to address a wide variety of eigenvalue and singular value computations.

9.1. Generalized eigenvalue problems. Consider the problem

$$\text{Find nonzero } \boldsymbol{x} \in \mathbb{C}^n \text{ and } \lambda \in \mathbb{C} : \quad \boldsymbol{H}\boldsymbol{x} = \lambda \boldsymbol{J}\boldsymbol{x} \quad \text{where } \boldsymbol{H}, \boldsymbol{J} \in \mathbb{C}^{n\times n}. \tag{9.1}$$

Suppose that $\boldsymbol{B} \in \mathbb{C}^{n\times d}$ is a basis that captures approximate solutions to (9.1). Following the development in subsection 6.1, the classic RR method can be interpreted as a variational problem:

$$\text{minimize}_{\boldsymbol{M}\in\mathbb{C}^{d\times d}} \quad \|\boldsymbol{H}\boldsymbol{B} - \boldsymbol{J}\boldsymbol{B}\boldsymbol{M}\|_{\mathrm{F}}. \tag{9.2}$$

Given a solution $\boldsymbol{M}_\star = (\boldsymbol{J}\boldsymbol{B})^\dagger(\boldsymbol{H}\boldsymbol{B})$ to (9.2), we pose the ordinary eigenvalue problem $\boldsymbol{M}_\star\boldsymbol{y} = \theta\boldsymbol{y}$. Each eigenpair $(\boldsymbol{y}, \theta)$ induces an approximate solution $(\boldsymbol{B}\boldsymbol{y}, \theta)$ to (9.1).

Given a subspace embedding $\boldsymbol{S} \in \mathbb{C}^{s\times n}$ for range$([\boldsymbol{H}\boldsymbol{B}, \boldsymbol{J}\boldsymbol{B}])$, we can pass to the sketched problem

$$\text{minimize}_{\boldsymbol{M}\in\mathbb{C}^{d\times d}} \quad \|\boldsymbol{S}(\boldsymbol{H}\boldsymbol{B} - \boldsymbol{J}\boldsymbol{B}\boldsymbol{M})\|_{\mathrm{F}}. \tag{9.3}$$

The solution $\hat{\boldsymbol{M}} = (\boldsymbol{S}\boldsymbol{J}\boldsymbol{B})^\dagger(\boldsymbol{S}\boldsymbol{H}\boldsymbol{B})$. Then frame the ordinary eigenvalue problem $\hat{\boldsymbol{M}}\boldsymbol{y} = \theta\boldsymbol{y}$. Each eigenpair $(\hat{\boldsymbol{y}}, \hat{\theta})$ induces an approximate solution $(\boldsymbol{B}\hat{\boldsymbol{y}}, \hat{\theta})$ to (9.1).

Excluding basis generation, we can solve the generalized eigenvalue problem via sketching with $O(d^3 + nd \log d)$ operations. In contrast, the classic RR approach typically requires $O(nd^2)$ operations.

9.2. Low-rank matrix approximation. The most successful application of randomized matrix computation has been to approximate truncated singular value decompositions efficiently [20, 26]. Using the new insights from our paper, we can accelerate these algorithms by sketching. The resulting techniques share some genes with sketch-based algorithms for low-rank matrix approximation [52, 50, 51, 29], but they are different in spirit.

Let $\boldsymbol{A} \in \mathbb{C}^{m\times n}$ be a matrix. Let $\boldsymbol{B} \in \mathbb{C}^{n\times d}$ be a basis, and suppose that we have access to the reduced matrix $\boldsymbol{AB} \in \mathbb{C}^{n\times d}$. We can frame low-rank matrix approximation as a variational problem:

$$\text{minimize}_{\boldsymbol{M}\in\mathbb{C}^{d\times d}} \quad \|\boldsymbol{ABM} - \boldsymbol{A}\|_{\mathrm{F}}. \tag{9.4}$$

The solution $\boldsymbol{M}_\star = (\boldsymbol{AB})^\dagger \boldsymbol{A}$ produces the rank-d matrix approximation

$$\hat{\boldsymbol{A}} = \boldsymbol{ABM}_\star = (\boldsymbol{AB})(\boldsymbol{AB})^\dagger \boldsymbol{A} = \boldsymbol{QQ}^*\boldsymbol{A},$$

where $\boldsymbol{Q} \in \mathbb{C}^{n\times d}$ is an orthonormal basis for the range of $\boldsymbol{AB}$. If we choose $\boldsymbol{B}$ at random, we obtain the Halko et al. randomized SVD algorithm [20]. If we form an adapted basis $\boldsymbol{B}$ by means of subspace iteration [35, 20] or block Krylov methods [35, 19, 28, 48], we obtain much better approximations, as described in the citations.

Let $\boldsymbol{S} \in \mathbb{C}^{s\times n}$ be an "affine space" embedding with $s = 2d$. (The SRFT (2.6) and sparse map (2.7) both qualify.) We pose the sketched problem

$$\text{minimize}_{\boldsymbol{M}\in\mathbb{C}^{d\times d}} \quad \|\boldsymbol{S}(\boldsymbol{ABM} - \boldsymbol{A})\|_{\mathrm{F}}. \tag{9.5}$$

The solution $\hat{\boldsymbol{M}} = (\boldsymbol{SAB})^\dagger(\boldsymbol{SA})$ yields the rank-d matrix approximation

$$\hat{\boldsymbol{A}}_{\text{sketch}} = (\boldsymbol{AB})(\boldsymbol{SAB})^\dagger(\boldsymbol{SA}). \qquad \textbf{[Unstable!]} \tag{9.6}$$

This formula is completely unsuitable for practical computation, but it can be replaced with a stable and efficient variant [29]. If we choose $\boldsymbol{B}$ to be a second sketching map, we obtain the (low-accuracy) sketched SVD algorithms from [52, 50, 29].

Our work delivers the novel insight that using an *adapted* basis $\boldsymbol{B}$ in (9.6) leads to a fast *and* accurate algorithm for low-rank matrix approximation. Excluding the cost of basis generation, we can stably form the approximation in $O(d^3 + (m+n)d \log d)$ operations. In contrast with sketched SVD algorithms, we attain errors similar to randomized subspace iteration [20] or randomized block Krylov methods [28, 48].

10. Prospects. We believe that our framework for combining sketching with subspace projection methods presents many exciting opportunities and challenges. Let us close by highlighting some of the prospects.

First, aside from GMRES and RR, there are many subspace projection methods that might benefit from sketching. For instance, there is an important class of algorithms (BiCG, BiCGstab, CGS, QMR, etc.) for solving linear systems by means of Lanczos biorthogonalization. These methods form Krylov subspaces with respect to both $\boldsymbol{A}$ and $\boldsymbol{A}^*$ using three-term recurrences, but they have complicated stability properties. Perhaps, with sketching, we can improve the profile of these methods.

Second, our work suggests that traditional strategies for building high-dimensional Krylov subspace bases merit a fresh look. For example, our experiments indicate that

we can easily run thousands of iterations of sGMRES, whereas orthogonalization dominates the cost of classic GMRES after, say, a few dozen iterations. One consequence is that it would suffice to find a "mediocre" preconditioner for linear systems that reduces the iteration complexity of sGMRES to 1000s of iterations, rather than the historical goal of 10s of iterations. Other aspects of basis generation that deserve further attention include restarting, deflation, and pruning.

Third, we believe that the performance advantages of sGMRES and sRR algorithms would be maximized in modern computing environments where communication and synchronization costs dominate computation [6]. For example, we can trivially parallelize the computation of block Krylov subspaces. Likewise, sketching allows us to perform approximate orthogonalization of distributed vectors by means of short messages. While our experiments focused on a serial computing environment, there are clear opportunities for efficient implementations on GPUs, multicore and parallel processors, distributed and cloud computing systems, and so forth.

Finally, let us mention one remaining difficulty. At present, we lack a reliable mechanism for guaranteeing that the condition number of basis $\boldsymbol{B}$ and the reduced matrix $\boldsymbol{AB}$ do not explode. Partial orthogonalization is a practical approach that often works well, but it can fail. It would be valuable to identify strategies for inexpensively producing computational bases that are numerically full rank.

Acknowledgments. The authors would like to thank Alice Cortinovis, Ethan Epperly, Ilse Ipsen, Gunnar Martinsson, Maike Meier, Florian Schaefer for valuable discussions and feedback.

REFERENCES

[1] S. Adachi, S. Iwata, Y. Nakatsukasa, and A. Takeda, *Solving the trust-region subproblem by a generalized eigenvalue problem*, SIAM J. Optim., 27 (2017), pp. 269–291.

[2] N. Ailon and B. Chazelle, *The fast Johnson-Lindenstrauss transform and approximate nearest neighbors*, SIAM J. Comput., 39 (2009), pp. 302–322, https://doi.org/10.1137/060673096.

[3] H. Avron, P. Maymounkov, and S. Toledo, *Blendenpik: Supercharging LAPACK's least-squares solver*, SIAM J. Sci. Comp, 32 (2010), pp. 1217–1236.

[4] O. Balabanov and L. Grigori, *Randomized Gram-Schmidt process with application to GMRES*, arXiv:2011.05090, (2020).

[5] G. Ballard, E. Carson, J. Demmel, M. Hoemmen, N. Knight, and O. Schwartz, *Communication lower bounds and optimal algorithms for numerical linear algebra*, Acta Numerica, 23 (2014), pp. 1–155.

[6] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz, *Minimizing communication in numerical linear algebra*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 866–901.

[7] B. Beckermann, *The condition number of real Vandermonde, Krylov and positive definite Hankel matrices*, Numer. Math., 85 (2000), pp. 553–577.

[8] G. Boutry, M. Elad, G. H. Golub, and P. Milanfar, *The generalized eigenvalue problem for nonsquare pencils using a minimal perturbation approach*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 582–601.

[9] T. Chen and E. Carson, *Predict-and-recompute conjugate gradient variants*, SIAM J. Sci. Comp, 42 (2020), pp. A3084–A3108.

[10] K. L. Clarkson and D. P. Woodruff, *Low-rank approximation and regression in input sparsity time*, Journal of the ACM, 63 (2017), pp. 1–45.

[11] M. B. Cohen, *Nearly tight oblivious subspace embeddings by trace inequalities*, in Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms, SIAM, 2016, pp. 278–287.

[12] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust Region Methods*, vol. 1, SIAM, Philadelphia, PA, USA, 2000.

[13] T. A. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Trans. Math. Soft., 38 (2011), pp. 1–25.

[14] J. Demmel, *The condition number of equivalence transformations that block diagonalize matrix pencils*, SIAM J. Numer. Anal., 20 (1983), pp. 599–610.
[15] J. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, USA, 1997.
[16] W. Gautschi, *The condition of orthogonal polynomials*, Math. Comp., 26 (1972), pp. 923–924, https://doi.org/10.2307/2005876.
[17] W. Gautschi, *The condition of polynomials in power form*, Math. Comp., 33 (1979), pp. 343–352, https://doi.org/10.2307/2006047.
[18] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, 4th ed., 2012.
[19] N. Halko, P.-G. Martinsson, Y. Shkolnisky, and M. Tygert, *An algorithm for the principal component analysis of large data sets*, SIAM J. Sci. Comput., 33 (2011), pp. 2580–2594, https://doi.org/10.1137/100804139.
[20] N. Halko, P.-G. Martinsson, and J. A. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.
[21] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, PA, USA, second ed., 2002.
[22] S. Ito and K. Murota, *An algorithm for the generalized eigenvalue problem for nonsquare matrix pencils by minimal perturbation approach*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 409–419.
[23] W. D. Joubert and G. F. Carey, *Parallelizable restarted iterative methods for nonsymmetric linear systems. Part I: Theory*, Center for Numerical Analysis CNA-251, University of Texas at Austin, May 1991.
[24] W. Kahan, *Spectra of nearly Hermitian matrices*, Proc. Amer. Math. Soc., 48 (1975), pp. 11–17.
[25] T. A. Manteuffel, *The Tchebychev iteration for nonsymmetric linear systems*, Numer. Math., 28 (1977), pp. 307–327, https://doi.org/10.1007/BF01389971.
[26] P.-G. Martinsson and J. A. Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numerica, (2020), pp. 403—572.
[27] X. Meng and M. W. Mahoney, *Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression*, in STOC'13—Proceedings of the 2013 ACM Symposium on Theory of Computing, ACM, New York, 2013, pp. 91–100, https://doi.org/10.1145/2488608.2488621.
[28] C. Musco and C. Musco, *Stronger and faster approximate singular value decomposition via the block Lanczos method*, in Advances in Neural Information Processing Systems (NIPS 2015), 2014, pp. 14243–14253. Available at http://arXiv.org/abs/1504.05477.
[29] Y. Nakatsukasa, *Fast and stable randomized low-rank matrix approximation*, arXiv:2009.11392, (2020).
[30] Y. Nakatsukasa, *Sharp error bounds for Ritz vectors and approximate singular vectors*, Math. Comp., 89 (2020), pp. 1843–1866.
[31] J. Nelson and H. L. Nguyên, *Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings*, in 2013 ieee 54th annual symposium on foundations of computer science, IEEE, 2013, pp. 117–126.
[32] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.
[33] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 1998.
[34] B. Philippe and L. Reichel, *On the generation of Krylov subspace bases*, Appl. Numer. Math., 62 (2012), pp. 1171–1186.
[35] V. Rokhlin, A. Szlam, and M. Tygert, *A randomized algorithm for principal component analysis*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 1100–1124.
[36] V. Rokhlin and M. Tygert, *A fast randomized algorithm for overdetermined linear least-squares regression*, Proc. Natl. Acad. Sci., 105 (2008), pp. 13212–13217.
[37] Y. Saad, *Iterative methods for sparse linear systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, second ed., 2003, https://doi.org/10.1137/1.9780898718003.
[38] Y. Saad, *Numerical methods for large eigenvalue problems*, vol. 66 of Classics in Applied Mathematics, SIAM, Philadelphia, PA, 2011, https://doi.org/10.1137/1.9781611970739.ch1. Revised edition of the 1992 original.
[39] Y. Saad and M. H. Schultz, *Conjugate gradient-like algorithms for solving nonsymmetric linear systems*, Math. Comp., 44 (1985), pp. 417–424, https://doi.org/10.2307/2007961.
[40] Y. Saad and M. H. Schultz, *GMRES - A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comp., 7 (1986), pp. 856–869.
[41] T. Sarlos, *Improved approximation algorithms for large matrices via random projections*,

in 2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), IEEE, 2006, pp. 143–152.

[42] H. D. Simon, *Analysis of the symmetric Lanczos algorithm with reorthogonalization methods*, Linear Algebra Appl., 61 (1984), pp. 101–131.

[43] H. D. Simon, *The Lanczos algorithm with partial reorthogonalization*, Math. Comp., 42 (1984), pp. 115–142.

[44] G. W. Stewart, *A generalization of Saad's theorem on Rayleigh-Ritz approximations*, Linear Algebra Appl., 327 (1999), pp. 115–119.

[45] G. W. Stewart, *Matrix Algorithms Volume II: Eigensystems*, SIAM, Philadelphia, 2001.

[46] G. W. Stewart and J.-G. Sun, *Matrix Perturbation Theory (Computer Science and Scientific Computing)*, Academic Press, 1990.

[47] J. A. Tropp, *Improved analysis of the subsampled randomized Hadamard transform*, Advances in Adaptive Data Analysis, 3 (2011), pp. 115–126.

[48] J. A. Tropp, *Analysis of randomized block Krylov methods*, ACM Technical Report 2018-02, California Institute of Technology, 2018.

[49] J. A. Tropp, *Randomized block krylov methods for approximating extreme eigenvalues*, arXiv:2110.00649, (2021).

[50] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Practical sketching algorithms for low-rank matrix approximation*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 1454–1485.

[51] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher, *Streaming low-rank matrix approximation with an application to scientific simulation*, SIAM J. Sci. Comp, 41 (2019), p. A2430–A2463.

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

[53] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert, *A fast randomized algorithm for the approximation of matrices*, Appl. Comput. Harmon. Anal., 25 (2008), pp. 335–366.

[54] A. Yurtsever, J. A. Tropp, O. Fercoq, M. Udell, and V. Cevher, *Scalable semidefinite programming*, SIAM J. Math. Data Sci., 3 (2021), pp. 171–200, https://doi.org/10.1137/19M1305045.