FAST & ACCURATE RANDOMIZED ALGORITHMS FOR LINEAR SYSTEMS AND EIGENVALUE PROBLEMS

YUJI NAKATSUKASA and JOEL A. TROPP

Caltech ACM Report No. 2021-01
doi:10.7907/cmyh-va31

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 dimension reduction ("sketching") to accelerate standard subspace projection methods, such as GMRES and Rayleigh–Ritz. This modification makes it possible to incorporate nontraditional bases for the approximation subspace that are easier to construct. As compared with classic methods, the new algorithms have similar accuracy but run faster and may use less storage. For model problems, numerical experiments show large advantages over MATLAB's optimized routines, including a 70× 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.

In many scientific computing and machine learning problems, we expend the majority of our computational resources in solving large-scale linear algebra problems [50, 51, 32], typically linear systems $\boldsymbol{Ax} = \boldsymbol{f}$ or eigenvalue problems $\boldsymbol{Ax} = \lambda\boldsymbol{x}$. Numerical linear algebra (NLA) is a research field that attempts to devise practical algorithms for solving these problems.

Arguably, the most exciting recent development in 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 [26, 36] to compute truncated singular value decompositions of large matrices. Randomized preconditioning [49, 3] allows us to solve highly overdetermined least-squares problems faster than any previous algorithm. Yet the NLA community has made less progress on other core challenges, especially problems involving nonsymmetric square matrices.

This paper exposes a new class of algorithms for solving nonsymmetric linear systems and eigenvalue problems. The methods can also be competitive for symmetric problems. Our framework combines subspace projection methods [50, 51], such as GMRES and the Rayleigh–Ritz process, with the modern technique of randomized dimension reduction [54, 68, 36]. 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 classic counterparts, without much loss of accuracy. In retrospect, the marriage of these ideas appears inevitable.

This introduction offers an overview of our approach and some evidence about potential applications in scientific computing and optimization. The rest of the paper fleshes out the proposal by providing more technical details, examples, and extensions. The technical report [41] contains further information.

1.1. Sketching a least-squares problem.

The *sketch-and-solve* paradigm [54, 68, 36] 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 low-dimensional subspace and to solve the smaller problem instead. The solution of this "sketched problem"

[*]Date: 29 October 2021. Revised: 4 February 2022 and 23 September 2022.

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@caltech.edu).

sometimes serves in place of the solution to the original computational problem. This section offers a short example; see section 2 for full details.

A *sketching matrix* is a random matrix $\boldsymbol{S} \in \mathbb{C}^{s\times n}$ that preserves the squared length of a fixed vector in expectation:

$$\mathbb{E}\,\|\boldsymbol{S}\boldsymbol{x}\|_2^2 = \|\boldsymbol{x}\|_2^2 \quad \text{for each } \boldsymbol{x} \in \mathbb{C}^n. \tag{1.1}$$

The expectation $\mathbb{E}$ averages over the randomness in $\boldsymbol{S}$, and $\|\cdot\|_2$ denotes the ℓ_2 norm. There are many constructions of fast sketching matrices that we can apply to a vector in $\mathbb{C}^n$ with $O(n \log n)$ arithmetic operations. The choice of the embedding dimension s depends on the application.

Consider the overdetermined $n \times d$ least-squares problem:

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

where $\boldsymbol{M} \in \mathbb{C}^{n\times d}$ is a tall matrix with $n \gg d$. The right-hand side $\boldsymbol{f} \in \mathbb{C}^n$. Draw a random sketching matrix $\boldsymbol{S} \in \mathbb{C}^{s\times n}$ with embedding dimension $s = 2(d+1)$. We formulate and solve the smaller $s \times d$ sketched problem:

$$\text{minimize}_{\boldsymbol{y}\in\mathbb{C}^d} \quad \|\boldsymbol{S}(\boldsymbol{M}\boldsymbol{y} - \boldsymbol{f})\|_2. \tag{1.3}$$

With high probability[1] (whp) over the choice of the sketching matrix $\boldsymbol{S}$, this procedure produces a satisfactory result. Indeed, we can compare the residual norms of the solution $\hat{\boldsymbol{y}}$ to the sketched problem (1.3) and the solution $\boldsymbol{y}_\star$ to the original problem (1.2). More precisely, the sketching method ensures that

$$\|\boldsymbol{M}\boldsymbol{y}_\star - \boldsymbol{f}\|_2 \quad\leq\quad \|\boldsymbol{M}\hat{\boldsymbol{y}} - \boldsymbol{f}\|_2 \quad\leq\quad 6\cdot\|\boldsymbol{M}\boldsymbol{y}_\star - \boldsymbol{f}\|_2 \quad \text{whp.} \tag{1.4}$$

Provided that the original problem has a tiny residual, the solution to the sketched problem also yields a tiny residual! For a fast sketching matrix $\boldsymbol{S}$, the whole sketch-and-solve process can be significantly faster than solving (1.2) directly.

1.2. Solving linear systems by sketched GMRES.

Now, suppose that we wish to solve the (nonsymmetric, nonsingular) linear system

$$\text{Find } \boldsymbol{x} \in \mathbb{C}^n : \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{f} \quad \text{where } \boldsymbol{A} \in \mathbb{C}^{n\times n} \text{ and } \boldsymbol{f} \in \mathbb{C}^n. \tag{1.5}$$

All algorithms in this paper access the matrix via products: $\boldsymbol{x} \mapsto \boldsymbol{A}\boldsymbol{x}$. This is a good modeling assumption for a wide range of problems, especially when $\boldsymbol{A}$ approximates a differential operator or integral kernel. It is also useful when $\boldsymbol{A}$ is sparse or has some algebraic structure (e.g., a Gram matrix).

Our approach to solving (1.5) builds on a standard template, called a *subspace projection method* [50], 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 (1.5). 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. In typical

[1]In practice, the sketching method rarely fails. The worst case scenario is that the constant 6 in the error (1.3) may be slightly larger. For this reason, we avoid quantifying the probabilities.

situations, the basis has very low dimension: $d \ll n$. We will discuss the construction of a basis in subsection 1.2.2.

At its heart, the GMRES algorithm [53, 52, 50] is a subspace projection method that replaces the linear system (1.5) with the overdetermined least-squares problem

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

The solution $\boldsymbol{y}_\star$ to (1.6) yields an approximate solution $\boldsymbol{x}_{\boldsymbol{B}} = \boldsymbol{B}\boldsymbol{y}_\star$ to the linear system (1.5). 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 (1.6) is a natural candidate for sketching. Draw a fast random sketching matrix $\boldsymbol{S} \in \mathbb{C}^{s\times n}$ with $s = 2(d+1)$, and sketch the problem:

$$\text{minimize}_{\boldsymbol{y}\in\mathbb{C}^d} \quad \|\boldsymbol{S}(\boldsymbol{A}\boldsymbol{B}\boldsymbol{y} - \boldsymbol{f})\|_2. \tag{1.7}$$

The solution $\hat{\boldsymbol{y}}$ of the sketched problem (1.7) induces an approximate solution $\hat{\boldsymbol{x}} = \boldsymbol{B}\hat{\boldsymbol{y}}$ to the linear system (1.5). According to (1.4), the residual norm of the sketched solution is comparable with the original residual norm:

$$\|\boldsymbol{A}\boldsymbol{x}_{\boldsymbol{B}} - \boldsymbol{f}\|_2 \quad \leq \quad \|\boldsymbol{A}\hat{\boldsymbol{x}} - \boldsymbol{f}\|_2 \quad \leq \quad 6 \cdot \|\boldsymbol{A}\boldsymbol{x}_{\boldsymbol{B}} - \boldsymbol{f}\|_2 \quad \text{whp.}$$

In summary, the sketched formulation (1.7) is effective if and only if the subspace $\text{range}(\boldsymbol{B})$ contains an accurate approximate solution of the linear system.

We refer to (1.7) as the *sketched GMRES* problem (sGMRES). For an unstructured basis $\boldsymbol{B}$, the sGMRES approach is faster than solving the original least-squares problem (1.6), both in theory and in practice. With careful implementation, sGMRES is robust, even when the conditioning of the reduced matrix $\boldsymbol{A}\boldsymbol{B}$ is poor. Indeed, it suffices that $\kappa_2(\boldsymbol{A}\boldsymbol{B}) \lesssim u^{-1}$ where u is the unit roundoff[2]; see subsection 3.4. 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 basis that captures an approximate solution to the linear system (1.5). Where can we get this basis? Consider a Krylov subspace of the form

$$\mathsf{K}_p(\boldsymbol{A}; \boldsymbol{f}) := \text{span}\{\boldsymbol{f}, \boldsymbol{A}\boldsymbol{f}, \boldsymbol{A}^2\boldsymbol{f}, \dots, \boldsymbol{A}^{p-1}\boldsymbol{f}\}. \tag{1.8}$$

This is a natural subspace to build because we access $\boldsymbol{A}$ via matrix–vector products. Furthermore, the Krylov subspace often contains an excellent approximate solution to the linear system, even when the depth $p \ll n$. See [50, Chaps. 6 and 7].

For computations, we need an explicit basis $\boldsymbol{B}$ whose columns span the Krylov subspace. Although it is straightforward to form the monomial basis visible in (1.8), the condition number of the basis often grows exponentially [10, 23], rendering the basis useless for numerical purposes. The standard practice is to fully orthogonalize the monomial basis (using the Arnoldi process of subsection 4.2), which is costly. But sGMRES does not require an orthogonal basis to operate! Instead, we will consider other procedures that quickly construct Krylov subspace bases with a modest condition number. Section 4 outlines several possible approaches.

For concreteness, we focus on the *k-truncated Arnoldi process*; the parameter k is a small natural number. This algorithm assembles a basis $\boldsymbol{B} = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_d] \in \mathbb{C}^{n\times d}$

[2] In standard IEEE double-precision arithmetic, the unit roundoff $u \approx 2 \cdot 10^{-16}$.

Algorithm 1.1 sGMRES + k-truncated Arnoldi

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 truncated orthogonalization, stability tolerance $\texttt{tol} = O(u^{-1})$.
Output: Approximate solution $\hat{\boldsymbol{x}} \in \mathbb{C}^n$ to linear system (1.5) and estimated residual norm $\hat{r}_{\text{est}}$

```
1  function sGMRES
2      Draw subspace embedding $\boldsymbol{S} \in \mathbb{C}^{s\times n}$ with $s = 2(d+1)$        ▷ See subsection 2.4
3      Form residual and sketch: $\boldsymbol{r} = \boldsymbol{f} - \boldsymbol{Ax}$ and $\boldsymbol{g} = \boldsymbol{Sr}$
4      Normalize basis vector $\boldsymbol{b}_1 = \boldsymbol{r}/\|\boldsymbol{r}\|_2$ and apply matrix $\boldsymbol{m}_1 = \boldsymbol{Ab}_1$
5      for $j = 2, 3, 4, \dots, d$ do
6          Truncated 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 $i \ge 0$
7          Normalize basis vector $\boldsymbol{b}_j = \boldsymbol{w}_j/\|\boldsymbol{w}_j\|_2$ and apply matrix $\boldsymbol{m}_j = \boldsymbol{Ab}_j$
8      Sketch reduced matrix: $\boldsymbol{C} = \boldsymbol{S}[\boldsymbol{m}_1, \dots, \boldsymbol{m}_d]$
9      Thin QR factorization: $\boldsymbol{C} = \boldsymbol{UT}$
10     if condition number $\kappa_2(\boldsymbol{T}) > \texttt{tol}$ then warning...
11         Either whiten $\boldsymbol{B} \leftarrow \boldsymbol{BT}^{-1}$ or form new residual and restart        ▷ See subsection 5.3
12     Solve least-squares problem: $\hat{\boldsymbol{y}} = \boldsymbol{T}^{-1}(\boldsymbol{U}^*\boldsymbol{g})$        ▷ See (3.7)
13     Residual estimate: $\hat{r}_{\text{est}} = \|(\mathbf{I} - \boldsymbol{UU}^*)\boldsymbol{g}\|_2$        ▷ See (3.8)
14     Construct solution: $\hat{\boldsymbol{x}} = \boldsymbol{x} + [\boldsymbol{m}_1, \dots, \boldsymbol{m}_j]\hat{\boldsymbol{y}}$
```

Implementation: In line 6, use double Gram–Schmidt for stability. In line 9, the QR factorization may require pivoting. In lines 11–12, apply $\boldsymbol{T}^{-1}$ via triangular substitution.

iteratively by orthogonalizing each new vector against the previous k vectors in the basis. In detail, define $\boldsymbol{b}_{-i} = \mathbf{0}$ for $i \ge 0$. Set $\boldsymbol{b}_1 = \boldsymbol{f}/\|\boldsymbol{f}\|_2$. For each $j = 2, \dots, d$,

$$\boldsymbol{b}_j = \boldsymbol{w}_j/\|\boldsymbol{w}_j\|_2 \quad \text{where} \quad \boldsymbol{w}_j = (\mathbf{I} - \boldsymbol{b}_{j-1}\boldsymbol{b}_{j-1}^* - \cdots - \boldsymbol{b}_{j-k}\boldsymbol{b}_{j-k}^*)(\boldsymbol{Ab}_{j-1}). \tag{1.9}$$

We have written * for the (conjugate) transpose. Note that we obtain the reduced matrix $\boldsymbol{AB}$ as a by-product of this computation.

We have found that k-truncated Arnoldi often yields a basis with moderate condition number, even when $k = 2$ or $k = 4$. Nevertheless, we are not aware of any fast, universal procedure for constructing a Krylov subspace basis with full numerical rank, short of strategies that perform more costly orthogonalization steps. This is a matter for further research.

1.2.3. Comparison with GMRES. To recap, the standard version of the GMRES algorithm [53] applies the expensive Arnoldi process (with full orthogonalization; see subsection 4.2) to build an orthonormal basis for the Krylov subspace, and it exploits the structure of this basis to solve the least-squares problem (1.6) efficiently.

In contrast, we propose to use a quick-and-dirty construction, such as the k-truncated Arnoldi process, to obtain a basis for the Krylov subspace. Then we solve the sGMRES least-squares problem (1.7) to produce an approximate solution of the linear system. When the basis dimension $d \ll n$, the sGMRES approach has lower arithmetic costs than classic GMRES, while attaining similar accuracy:

$$\boxed{\text{GMRES: } O(nd^2) \text{ operations} \quad \text{vs.} \quad \text{sGMRES: } O(d^3 + nd\log d) \text{ operations.}}$$

This expression assumes that sGMRES uses k-truncated Arnoldi for k constant, as well as a fast sketching matrix (subsection 2.4). See Algorithm 1.1 for pseudocode.

As evidence for the benefits of using sGMRES, we use it to solve a finite-element (FEM) discretization of a convection–diffusion equation in the diffusion-dominant regime. Figure 1 depicts a **70× speedup** when the discretized linear system has dimension $n = 1,050,625$. In this case, sGMRES with 2-truncated Arnoldi attains

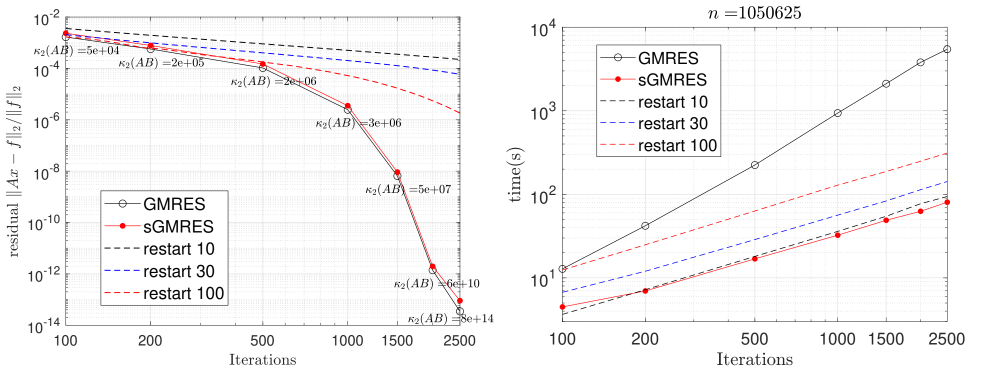

FIG. 1. ***sGMRES versus GMRES: Finite element discretization of a convection–diffusion equation.*** *These panels compare the performance of MATLAB* `gmres` *(with and without restarting) against the sGMRES algorithm (where the basis $\boldsymbol{B}$ is computed by k-truncated Arnoldi with $k = 2$).* ***Left:*** *Relative residual norms and condition number $\kappa_2(\boldsymbol{AB})$ of the reduced matrix.* ***Right:*** *Total runtime including basis generation.*

the same accuracy as GMRES with full orthogonalization. sGMRES is faster than restarted GMRES with restarting frequency 10, whose convergence is significantly impaired. Section 8 details the experimental setup and offers more illustrations.

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

$$\text{Find nonzero } \boldsymbol{x} \in \mathbb{C}^n \text{ and } \lambda \in \mathbb{C} : \quad \boldsymbol{Ax} = \lambda \boldsymbol{x} \quad \text{where } \boldsymbol{A} \in \mathbb{C}^{n\times n}. \tag{1.10}$$

As before, we access the matrix via products: $\boldsymbol{x} \mapsto \boldsymbol{Ax}$. 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 have procured a basis $\boldsymbol{B} \in \mathbb{C}^{n\times d}$ and the reduced matrix $\boldsymbol{AB} \in \mathbb{C}^{n\times d}$. The range of the basis should contain approximate eigenvectors $\boldsymbol{x}$ for which $\boldsymbol{Ax} \approx \lambda\boldsymbol{x}$. In this setting, the most commonly employed strategy is the Rayleigh–Ritz (RR) method.

We begin with the classic variational formulation [45, Thm. 11.4.2] of RR:

$$\text{minimize}_{\boldsymbol{M}\in\mathbb{C}^{d\times d}} \quad \|\boldsymbol{AB} - \boldsymbol{BM}\|_{\mathrm{F}}. \tag{1.11}$$

The solution is $\boldsymbol{M}_\star = \boldsymbol{B}^\dagger\boldsymbol{AB}$, where the dagger † denotes the Moore–Penrose pseudoinverse. At this point, RR frames the small $d \times d$ eigenvalue problem $\boldsymbol{M}_\star\boldsymbol{y} = \theta\boldsymbol{y}$. Each solution yields an approximate eigenpair $(\boldsymbol{By}, \theta)$ of the matrix $\boldsymbol{A}$.

Evidently, the least-squares problem (1.11) is ripe for sketching. Draw a sketching matrix $\boldsymbol{S} \in \mathbb{C}^{s\times n}$ with $s = 4d$, and pass to the sketched RR problem:

$$\text{minimize}_{\boldsymbol{M}\in\mathbb{C}^{d\times d}} \quad \|\boldsymbol{S}(\boldsymbol{AB} - \boldsymbol{BM})\|_{\mathrm{F}}. \tag{1.12}$$

We can compute the solution $\hat{\boldsymbol{M}} = (\boldsymbol{SB})^\dagger(\boldsymbol{SAB})$ to the sketched problem (1.12) faster than we can obtain $\boldsymbol{M}_\star$. As before, we frame an ordinary eigenvalue problem:

$$\hat{\boldsymbol{M}}\boldsymbol{y} = \theta\boldsymbol{y}. \tag{1.13}$$

Algorithm 1.2 sRR + k-truncated Arnoldi

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 `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 $s = 4d$    ▷ See subsection 2.4
    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
        Truncated 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 $\boldsymbol{b}_j = \boldsymbol{w}_j/\|\boldsymbol{w}_j\|_2$ 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 $\kappa_2(\boldsymbol{T}) >$ tol then warning:
        Either whiten $\boldsymbol{B} \leftarrow \boldsymbol{B}\boldsymbol{T}^{-1}$ 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 $i = 1, \dots, d$    ▷ See (6.12)
    Form residual estimates $\|\boldsymbol{D}\boldsymbol{y}_i - \lambda_i\boldsymbol{C}\boldsymbol{y}_i\|_2/\|\boldsymbol{C}\boldsymbol{y}_i\|_2$    ▷ See (6.10), subsection 6.4
    Identify set $\mathcal{I}$ of indices $i$ where residual is at most $\tau$
    Compute $\boldsymbol{x}_i = \boldsymbol{B}\boldsymbol{y}_i$ and normalize $\boldsymbol{x}_i := \boldsymbol{x}_i/\|\boldsymbol{x}_i\|_2$ for $i \in \mathcal{I}$, and output $(\boldsymbol{x}_i, \lambda_i)$
```

Implementation: In line 6, use double Gram–Schmidt for stability. In line 9, the QR factorization may require pivoting. In lines 11–12, apply $\boldsymbol{T}^{-1}$ via triangular substitution.

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 of (1.13) are competitive with the eigenpairs obtained from RR.

We refer to (1.12) as the *sketched Rayleigh–Ritz* (sRR) formulation. Although it demands a careful implementation, sRR is faster than the original least-squares method (1.11) for an unstructured basis $\boldsymbol{B}$. Moreover, sRR is robust, even when the basis $\boldsymbol{B}$ has poor conditioning. Indeed, it suffices that $\kappa_2(\boldsymbol{B}) \lesssim u^{-1}$.

1.3.2. Comparison with Arnoldi + Rayleigh–Ritz. We can also deploy a Krylov subspace (1.8) for eigenvalue computations [45, 51]. In this case, it is appropriate to 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 [51, Sec. 6.2] applies the Arnoldi process (with full orthogonalization; see subsection 4.2) 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-truncated Arnoldi (1.9), with the sRR eigenvalue problem (1.13). When the basis dimension $d \ll n$, this algorithm uses less arithmetic than the classic approach:

$$\boxed{\text{RR: } O(nd^2) \text{ operations} \quad \text{vs.} \quad \text{sRR: } O(d^3 + nd\log d) \text{ operations.}}$$

This expression includes basis generation via k-truncated Arnoldi for k constant, and sRR uses a fast sketching matrix (subsection 2.4). See Algorithm 1.2 for pseudocode.

As evidence that sRR is effective, Figure 2 highlights an eigenvalue computation arising from the trust region subproblem in numerical optimization. In this instance, 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.

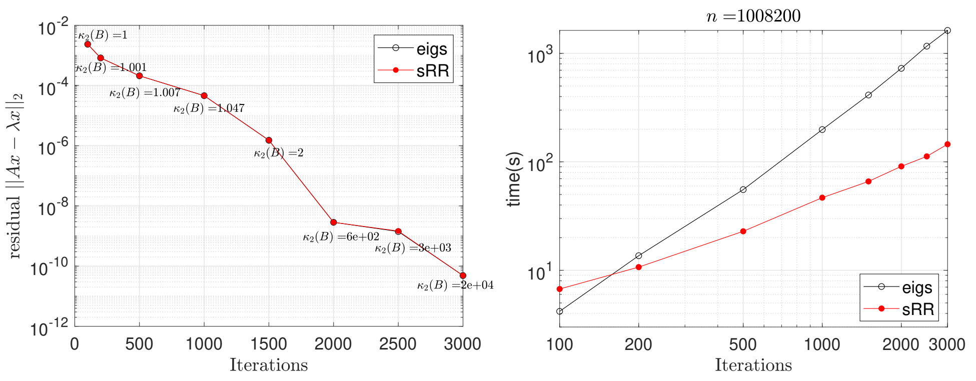

Fig. 2. ***RR versus sRR: Nonsymmetric eigenvalue problem.*** *These panels compare the performance of MATLAB* `eigs` *against the sRR algorithm (where the basis $\boldsymbol{B}$ is computed by k-truncated Arnoldi with $k = 10$). The sparse eigenvalue problem $\boldsymbol{Ax} = \lambda\boldsymbol{x}$ has dimension $n = 710^2 > 10^6$, and it arises from a trust-region subproblem in optimization.* ***Left:*** *Relative residual norm for the right-most eigenpair and the condition number $\kappa_2(\boldsymbol{B})$ of the basis. Throughout, all eigenvectors are normalized to have unit norm.* ***Right:*** *Total runtime including basis generation.*

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

$$\mathsf{K}_p(\boldsymbol{A};\boldsymbol{\Omega}) := \operatorname{span}\{\boldsymbol{\Omega}, \boldsymbol{A\Omega}, \boldsymbol{A}^2\boldsymbol{\Omega}, \dots, \boldsymbol{A}^{p-1}\boldsymbol{\Omega}\} \quad \text{where } \boldsymbol{\Omega} \in \mathbb{C}^{n\times b}. \tag{1.14}$$

We commonly generate the Krylov subspace from a random matrix $\boldsymbol{\Omega}$. The standard prescription recommends a very small block size b and a large depth p, but recent research [36, Sec. 11] has shown the value of a large block size with a moderate depth.

We must take care in constructing the block Krylov subspace. Truncated Arnoldi 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 $\boldsymbol{B} = [\boldsymbol{B}_1, \dots, \boldsymbol{B}_p] \in \mathbb{C}^{n\times bp}$:

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

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 $\boldsymbol{A}$. See section 7 for details and alternative methods for fast basis construction.

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, all these methods are limited by the approximation power of Krylov subspaces. Furthermore, we are not aware of a universal method for quickly computing a numerically full-rank basis for the Krylov subspace, short of costly strategies based on full orthogonalization. Both points merit further attention.

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.

We believe that we are the first authors to identify the natural connection between sketching and subspace projection methods for linear algebra problems. Indeed, we are not aware of any prior work where authors sketch the minimum-residual formulations (1.7) and (1.12) of general linear systems and eigenvalue problems.

Second, to obtain algorithms that are asymptotically faster than classic methods, we must also employ efficient constructions of Krylov subspace bases. In the past, researchers have regarded these techniques as a way to *postpone* expensive orthogonalization steps in parallel computing environments [29, 46]. 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.

In particular, there is a remarkable opportunity to design a new class of preconditioners for iterative solution of large-scale linear algebra problems. Indeed, since our approach allows for more flexible bases and mitigates orthogonalization costs, we can still derive benefits from a mediocre preconditioner that only reduces the iteration complexity to 100s or 1000s of iterations. We are excited about this prospect.

1.5. Related work. There has been relatively little research on the application of sketching for high-accuracy matrix computations. After this paper was completed, we learned about several papers [6, 7, 4, 5] that involve both subspace projection methods and sketching, but they are different in spirit from our work.

First, Balabanov & Nouy [6, 7] have argued that randomized algorithms may be used to accelerate reduced-order modeling. They focus on sketching an approximation subspace that captures solutions to a parameterized system of linear equations. In this setting, many additional complications arise from the need to scope out the parameter manifold and to interpolate between solutions at different parameter values. Their work does not suggest the simple, fast algorithms that we propose here.

Second, Balabanov & Grigori [4] have observed that sketching can reduce the cost of orthogonalization in the (block) Arnoldi method. Although this is an elegant idea, it only reduces the leading constant in the cost of the basis construction—but not its asymptotic scaling. In contrast, our approach yields asymptotically faster algorithms. As suggested in their follow-up work [5], there are appealing opportunities to combine their approach with ours.

1.6. 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 survey extensions and prospects.

1.7. 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 on sketching.

In the introduction, we defined a sketching map as a matrix that satisfies the norm-preservation relation (1.1). While this definition is intuitive, we actually need the sketching map to satisfy a more stringent condition called *subspace embedding*. We begin with a deterministic statement of the subspace embedding condition and its consequences, and then we argue that certain random matrices can serve as subspace embeddings.

2.1. 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 each vector in a given subspace. This definition is due to Sarlós [54]; see also [68, 36].

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*

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

For matrix computations, we need to design subspace embeddings that have several additional 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, the subspace L is typically unknown, so we must draw a subspace embedding $\boldsymbol{S}$ *at random*. A random embedding can achieve (2.1) with high probability. Last, to randomly embed a d-dimensional subspace with distortion ε, the optimal scaling of the embedding dimension s follows the law $s \approx d/\varepsilon^2$. 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 2.4, let us outline the applications of subspace embeddings that we will need in this paper.

2.2. 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 [54]; it serves as the foundation for a collection of methods called the *sketch-and-solve* paradigm [68, 36].

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*

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

In particular, the solution $\boldsymbol{y}_\star$ to the least-squares problem (1.2) and the solution $\hat{\boldsymbol{y}}$ to the sketched least-squares problem (1.3) satisfy residual norm bounds

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

Equation (2.3) justifies the statement (1.4) when $\varepsilon = 1/\sqrt{2}$.

2.3. Whitening the basis. Rokhlin & Tygert [49] 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 2.3 (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 a QR factorization of the sketched basis: $\boldsymbol{SB} = \boldsymbol{UT}$ with $\boldsymbol{U} \in \mathbb{C}^{s\times d}$ orthonormal and $\boldsymbol{T} \in \mathbb{C}^{d\times d}$ permuted triangular. Then the whitened basis $\bar{\boldsymbol{B}} := \boldsymbol{BT}^{-1}$ 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

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

2.4. Constructing a subspace embedding. There are many performant constructions of fast randomized subspace embeddings that work for an unknown subspace of bounded dimension [36, Sec. 9]. Let us summarize two that are most relevant for our purposes. In each case, when the subspace dimension is at most d, to obtain a random subspace embedding that has empirical distortion $\varepsilon \in (0,1)$ with high probability, we set the embedding dimension $s = d/\varepsilon^2$. In summary,

$$\boxed{\text{For distortion } \varepsilon\text{, set the embedding dimension } s = d/\varepsilon^2.}$$

We focus on the complex field; modifications for the real field are straightforward.

2.4.1. SRFTs. First, we introduce the subsampled random Fourier transform (SRFT) [2, 70, 62, 36]. This subspace embedding[3] 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[4] 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 [70]. It is an easy exercise to verify that $\boldsymbol{S}$ satisfies (1.1).

2.4.2. Sparse maps. Next, we describe the sparse dimension reduction map [37, 42, 14, 15, 36], which is useful for sparse data and may require less data movement. It takes the form

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

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 choose the sparsity level $\zeta = \lceil 2\log(1+d)\rceil$. We can apply $\boldsymbol{S}$ to a matrix $\boldsymbol{M}$ with $O(\zeta \cdot \texttt{nnz}(\boldsymbol{M}))$ operations, but it may require a sparse arithmetic library to achieve the best performance. You may check that $\boldsymbol{S}$ satisfies (1.1).

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. Fix a *full rank* 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 $\boldsymbol{x}_0 = \boldsymbol{0}$.

To find a solution of (3.1), consider the affine family of vectors of the form $\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{By}$ where $\boldsymbol{y} \in \mathbb{C}^d$. Among this class, we may search for a representative whose residual $\boldsymbol{r} = \boldsymbol{f} - \boldsymbol{Ax} = \boldsymbol{r}_0 - \boldsymbol{ABy}$ has the minimum ℓ_2 norm:

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

[3]For worst-case problems, a more elaborate SRFT construction may be needed [36, Sec. 9].

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

With some imprecision, we refer to (3.2) as the GMRES problem [53]. By calculus, the least-squares problem (3.2) is equivalent to the normal equations:

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

We can stably solve (3.2) using a QR factorization of the reduced matrix $\boldsymbol{AB}$ [27, 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 $\boldsymbol{x_B} = \boldsymbol{x}_0 + \boldsymbol{By_B}$ to (3.1) with residual $\boldsymbol{r_B} = \boldsymbol{f} - \boldsymbol{Ax_B}$.

The GMRES algorithm [52, 53] arises as a mechanism for solving (3.2) with a special choice of basis. The algorithm forms an orthonormal basis $\boldsymbol{B}$ for the Krylov subspace $\mathsf{K}_d(\boldsymbol{A}; \boldsymbol{r}_0)$ via the Arnoldi process (subsection 4.2). The basis construction requires $(d-1)$ matvecs with $\boldsymbol{A}$ plus $O(nd^2)$ arithmetic. This reduces (3.2) to a Hessenberg-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 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 the 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):

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

Thus, sGMRES produces approximate solutions to (3.1) with small ℓ_2 residuals precisely when GMRES does. *A posteriori*, we can diagnose the quality of the computed solution $\hat{\boldsymbol{x}}$ by examining the sketched residual norm:

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

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 robust 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 should be 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, pivoted QR decomposition of the sketched matrix: $\boldsymbol{SAB} = \boldsymbol{UT}$ where $\boldsymbol{U} \in \mathbb{C}^{s\times d}$ is orthonormal and $\boldsymbol{T} \in \mathbb{C}^{d\times d}$ is a triangular matrix with permuted columns. A minimizer of the sGMRES problem is

$$\hat{\boldsymbol{y}} = (\boldsymbol{SAB})^\dagger(\boldsymbol{Sr}_0) = \boldsymbol{T}^{-1}(\boldsymbol{U}^*(\boldsymbol{Sr}_0)). \tag{3.7}$$

Of course, we apply the inverse by triangular substitution. The sketched residual norm (3.6) admits the simple expression

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

We can obtain the quantities in the last two displays with $O(d^3)$ arithmetic since $s = O(d)$. Last, we construct the approximate solution $\hat{\boldsymbol{x}} = \boldsymbol{x}_0 + \boldsymbol{B}\hat{\boldsymbol{y}}$ at a cost of $O(nd)$ operations.

In summary, given any basis $\boldsymbol{B} \in \mathbb{C}^{n\times d}$, 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 gives an accounting of the total computational cost, including basis construction.

3.4. Stability. The classical stability result [27, 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 $\kappa_2(\boldsymbol{AB}) \leq 10^{15}$ 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 [50, Sec. 6.5.5]. That is, they use a basis $\boldsymbol{B}$ to compute an approximate solution $\boldsymbol{x}_{\boldsymbol{B}}$ to the linear system (3.1) with the residual vector $\boldsymbol{r}_{\boldsymbol{B}} = \boldsymbol{r}_0 - \boldsymbol{A}\boldsymbol{x}_{\boldsymbol{B}}$. If the residual norm $\|\boldsymbol{r}_{\boldsymbol{B}}\|_2$ exceeds an error tolerance, the residual vector $\boldsymbol{r}_{\boldsymbol{B}}$ 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, 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

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

A good preconditioner has two features [50, Chaps. 9 and 10]. First, the matrix $\boldsymbol{P}^{-1}\boldsymbol{A}$ has a more "favorable" structure 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. We believe that sGMRES opens up new opportunities for designing preconditioners because it is faster and more flexible than GMRES.

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 elegant way to capture solutions to a linear system when we access the matrix via products [50, Chaps. 6 and 7]. In this setting, the standard practice is to form an orthonormal basis for the Krylov subspace, but this approach may be prohibitively expensive. Remarkably, sGMRES does not require orthogonal bases, so we can look for cheaper constructions.

In this section, we describe several strategies for producing a nonorthogonal basis for a Krylov subspace. Although these strategies are decades old, they warrant a fresh look because sGMRES has a fundamentally different computational profile from GMRES. See the technical report [41] for further discussion.

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. The recurrence often 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 filter 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 [23].

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.

For theoretical analysis of the approximation power of Krylov subspaces in the context of linear system solvers, see [50, Sec. 6.11].

4.2. The Arnoldi process.

For motivation, we begin with the classic approach for producing a fully orthonormal Krylov basis. 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 [50, 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 roughly how the standard implementation of the GMRES algorithm operates [53].

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 modified or double Gram–Schmidt or else use Householder reflectors.

The literature contains many strategies for controlling the orthogonalization costs in the Arnoldi process [50, Chap. 6]. sGMRES motivates us to reevaluate techniques for building a nonorthogonal basis. For example, we can use k-truncated Arnoldi, as in (1.9), which reduces the cost of basis generation to $O(nk^2)$. Provided the reduced matrix $\boldsymbol{AB}$ is reasonably conditioned, we can still obtain accurate solutions to the linear system via sGMRES (3.4).

Relatedly, Balabanov & Grigori [4] have proposed to use a low-dimensional sketch of the basis to implement an approximate orthogonalization strategy inspired by Fact 2.3. Their approach has the same asymptotic cost as full orthogonalization, but similar ideas might be invoked to accelerate partial or selective orthogonalization.

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 [50, 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 fact 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 roughly how the MINRES algorithm operates [44].

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 [34, Chap. 5], but it can present a more serious challenge when solving eigenvalue problems [45, Chap. 13].

Although it is very efficient to solve the least-squares problem (3.2) by passing to the tridiagonal matrix $\boldsymbol{J}_p$, this approach can be fussy. 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 the depth $p \ll n$.

The literature describes many approaches for maintaining the orthogonality of the Lanczos basis, such as selective orthogonalization [45, Chap. 13]. When using the Lanczos basis for sGMRES, we can omit the extra orthogonalization steps. Alternatively, we may adopt the approximate orthogonalization strategies from subsection 4.2.

4.4. The Chebyshev recurrence. In some settings, we may wish to avoid the orthogonalization steps entirely because they involve operations on high-dimensional 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 [29]. Here is one appealing example of this technique.

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 [35, 29]. 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

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

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 k-truncated Arnoldi 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

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 [22, 35, 29, 46].

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 [46]. 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 [46]. Suppose that $\theta_1, \dots, \theta_p \in \mathbb{C}$ 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 [67] 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 1.1 contains pseudocode for a simple implementation of sGMRES using the k-truncated Arnoldi basis (1.9). 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 truncated Arnoldi basis with a Chebyshev basis (subsection 4.4). Table 1 summarizes the arithmetic costs.

5.2. Iterative sGMRES. As noted, most methods for producing the Krylov subspace basis generate the columns of $\boldsymbol{B}$ and the reduced matrix $\boldsymbol{AB}$ sequentially. 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. Here are the details.

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

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

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

$$\hat{\boldsymbol{y}}_j = \boldsymbol{T}_j^{-1}(\boldsymbol{U}_j^*(\boldsymbol{S}\boldsymbol{r}_0)) \quad \text{with} \quad \hat{r}_{\text{est},j} = \|(\mathbf{I} - \boldsymbol{U}_j\boldsymbol{U}_j)^*(\boldsymbol{S}\boldsymbol{r}_0)\|_2. \tag{5.2}$$

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 and whitening. 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 $\kappa_2(\boldsymbol{A}\boldsymbol{B}_j)$. 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 $\hat{\boldsymbol{r}}_{j-1} = \boldsymbol{r}_0 - \boldsymbol{A}\boldsymbol{B}_{j-1}\hat{\boldsymbol{y}}_{j-1}$.

Alternatively, instead of restarting, we can approximately orthogonalize $\boldsymbol{B}_{j-1}$ by replacing it with the whitened basis $\bar{\boldsymbol{B}}_{j-1} := \boldsymbol{B}_{j-1}\boldsymbol{T}_{j-1}^{-1}$, whose condition number is constant. Once we have done so, we may continue generating new basis vectors. Unfortunately, given $\boldsymbol{T}_{j-1}$, whitening still requires $O(j^2 n)$ arithmetic operations.

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

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

by iteratively regenerating the columns of the reduced matrix $\boldsymbol{A}\boldsymbol{B}_j$ and linearly combining them on the fly. For some basis constructions (e.g., truncated Arnoldi or Chebyshev), we only need to maintain a few columns of $\boldsymbol{B}$ and the j columns of $\boldsymbol{S}(\boldsymbol{A}\boldsymbol{B}_j)$. Note that regenerating vectors doubles the arithmetic cost associated with basis formation. A similar technique was used in [71].

5.5. Obtaining a solution with full accuracy. While the constant-factor loss (3.5) 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 (3.2) via an iterative method as in [49, 3]. This method still requires $\kappa_2(\boldsymbol{A}\boldsymbol{B}) \lesssim u^{-1}$ 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. Perspectives on Rayleigh–Ritz. Fix a *full-rank* basis $\boldsymbol{B} \in \mathbb{C}^{n \times d}$, 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 [51, Sec. 4.3]. Among nonzero vectors of the form $\boldsymbol{x} = \boldsymbol{By}$, 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 $\boldsymbol{B} \mapsto \boldsymbol{BT}$ 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* [12, 28]. 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 [45, 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 [51, 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 the Krylov subspace $\mathsf{K}_d(\boldsymbol{A}; \boldsymbol{\omega})$ generated by a random vector $\boldsymbol{\omega} \in \mathbb{C}^n$ at a cost of $O(nd^2)$ operations. This construction ensures that $\boldsymbol{Q}^*\boldsymbol{AQ}$ has Hessenberg form, so we can solve the eigenvalue problem (6.3) with $O(d^2)$ operations by means of the QR algorithm [51, 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

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

First, 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:

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

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

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

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 $[\boldsymbol{A}\boldsymbol{B}, \boldsymbol{B}]$.

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

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

We apply the inverse by triangular substitution. Then 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 much smaller than d.

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 $\kappa_2(\boldsymbol{B}) \leq 10^{15}$. This fact allows us to exploit fast non-orthogonal basis constructions; see section 7. See Algorithm 1.2 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 $\kappa_2(\boldsymbol{B}) \lesssim u^{-1}$. This condition is very generous. In contrast, recall that the standard stability analysis [45, Chap. 13] for the Lanczos algorithm asks that $\kappa_2(\boldsymbol{B}) < 1 + \sqrt{u}$.

If the sketched basis $\boldsymbol{S}\boldsymbol{B}$ is badly conditioned ($\kappa_2(\boldsymbol{S}\boldsymbol{B}) \gtrsim u^{-1}$), then the condition number diagnostic (2.5) implies that the basis $\boldsymbol{B}$ is also badly conditioned. In this case, we can stabilize sRR by restarting or whitening the basis; see subsection 7.2.

Alternatively, we can regularize the sketched basis and use a more reliable computational procedure. First, compute the truncated SVD:

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

Then we use the QZ algorithm [24, §7.7] to solve the *generalized* eigenvalue problem[5]

$$(\boldsymbol{U}^*\boldsymbol{SAB})\boldsymbol{Vz} = \theta\boldsymbol{\Sigma z}. \tag{6.13}$$

Each solution yields an sRR eigenpair $(\boldsymbol{Vz}, \theta)$ and an associated approximate eigenpair $(\boldsymbol{BVz}, \theta)$ 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 $\boldsymbol{SB} = \boldsymbol{UT}$ is the QR decomposition of the sketched basis. Per (2.4), the whitened basis $\bar{\boldsymbol{B}} := \boldsymbol{BT}^{-1}$ has conditioning $\kappa_2(\bar{\boldsymbol{B}}) \le (1+\varepsilon)/(1-\varepsilon)$. Now, consider the variational problem (6.7) with respect to the whitened basis $\bar{\boldsymbol{B}}$ and its sketched version:

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

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

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

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

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

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

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

From the last two displays, a short argument using the triangle inequality (add and subtract $\boldsymbol{A}\bar{\boldsymbol{B}}$) and the conditioning of the whitened basis produces

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

Here is an interpretation. If range$(\boldsymbol{Q})$ = range$(\boldsymbol{B})$ is close to an invariant subspace of $\boldsymbol{A}$, then $(\mathbf{I} - \boldsymbol{QQ}^*)\boldsymbol{AQ} \approx \mathbf{0}$. 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 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, and pathological examples are known [60, p. 282]. Nevertheless, RR provides excellent outputs in the vast majority of cases; see [40, 51, 58] for the analysis.

[5] 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).

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.

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 [30, 61]. 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 [45]. 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 the real part of the computed eigenvalues and eigenvectors. When $\boldsymbol{A}$ is complex Hermitian, we force the eigenvalues (but not the eigenvectors) to be real. 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

$$\mathsf{K}_p(\boldsymbol{A}; \boldsymbol{\Omega}) := \text{range}[\boldsymbol{\Omega}, \boldsymbol{A}\boldsymbol{\Omega}, \dots, \boldsymbol{A}^{p-1}\boldsymbol{\Omega}] = \text{span}\{\varphi(\boldsymbol{A})\boldsymbol{\Omega} : \deg(\varphi) \le p-1\}. \tag{7.1}$$

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

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

Here, $\{\varphi_i : i = 1, \dots, d\}$ is a linearly independent family of filter polynomials. For eigenvalue problems, the generating matrix $\boldsymbol{\Omega} \in \mathbb{C}^{n \times b}$ may be drawn at random from a standard normal distribution.

Historically, the NLA literature has prescribed a small block size, say $b \le 4$, and a very large depth p. More recent research [36, Sec. 11] has identified an opportunity to use a larger block size b, say 10s or 100s, and a moderate depth p. This shift in perspective has already transformed the computational profile of block Krylov methods for low-rank approximation. 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 [38, 64, 63] 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 subspace projection methods for eigenvalue problems that involve bases other than Krylov subspaces. For example, the Jacobi–Davidson method [57] and the LOBPCG algorithm [31] use 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. The Krylov–Schur algorithm [59] is a standard restarting method, but other approaches are possible here. 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 [26] 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 [45, 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

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

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

The block monomial basis has been used in the "blanczos" method [48, 25, 38, 63, 36] 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, however, it may not be necessary to reorthogonalize the block monomial basis $\boldsymbol{B}$.

7.4. Block Arnoldi with truncation. 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 $\boldsymbol{B}_1 = \texttt{orth}(\boldsymbol{\Omega})$ and iterate

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

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 [45, 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 k-truncated Arnoldi with a modest k as shown in Algorithm 1.2. However, for eigenvalue computations, there is compelling reason to take the block size $b \gg 1$. As the block size b increases, the cost of orthogonalization quickly becomes devastating.

7.5. Block Chebyshev recurrence. By employing other polynomial recurrences, we can potentially *eliminate* all expensive computations that involve orthogonalizing high-dimensional basis vectors. 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.

$$\boldsymbol{B}_1 = \boldsymbol{\Omega}; \quad \boldsymbol{B}_2 = \frac{1}{2\varrho}(\boldsymbol{A} - c\mathbf{I})\boldsymbol{\Omega}; \quad \boldsymbol{B}_j = \frac{1}{\varrho}\left[(\boldsymbol{A} - c\mathbf{I})\boldsymbol{B}_{j-1} - \frac{\delta_x^2 - \delta_y^2}{4\varrho}\boldsymbol{B}_{j-2}\right]. \tag{7.2}$$

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 [46].

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 an (approximately) 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 numerics that showcase the potential of sGMRES and sRR for solving large linear systems and eigenvalue problems. All examples involve real-valued matrices, with appropriate modifications to the methodology. All computations were performed in MATLAB version 2020a on a workstation with 256GB memory and 96 cores, each clocked at 3.3 GHz. See the technical report [41] for numerical work we have not shown in detail.

8.1. Solving linear systems with sGMRES. This subsection shows how the sGMRES method can solve nonsymmetric and symmetric linear systems.

8.1.1. Algorithm details. Our implementation of sGMRES follows the pseudocode in Algorithm 1.1. We construct a basis using k-truncated Arnoldi (1.9) with small values $k \in \{2, 4\}$ or with the Chebyshev basis (subsection 4.4). We do not whiten the basis or restart sGMRES. The subspace embedding is based on an SRFT matrix (2.6) where $\boldsymbol{E}$ has independent Rademacher[6] entries and $\boldsymbol{F}$ is a discrete cosine transform (DCT2). This sketch is easy to implement in MATLAB, but it uses $O(nd \log n)$ operations rather than $O(nd \log d)$.

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

8.1.2. Nonsymmetric linear systems. We begin with details about the nonsymmetric linear system discussed in the introduction (Figure 1). We use the IFISS

[6]A *Rademacher random variable* takes values ± 1 with equal probability.

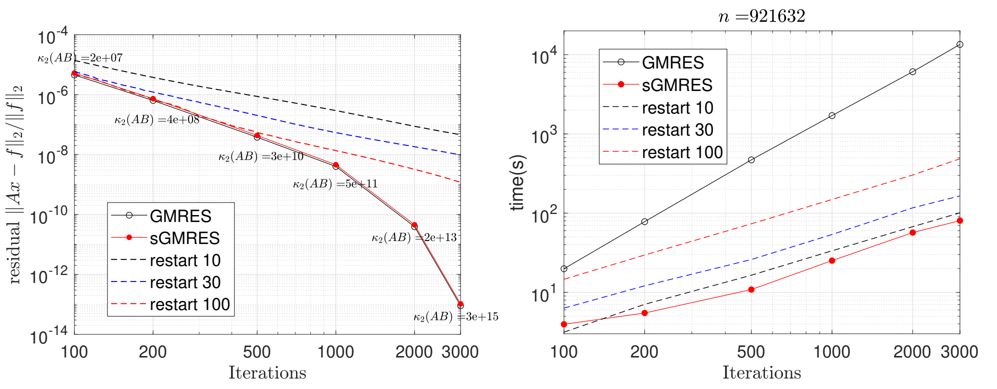

FIG. 3. ***GMRES versus sGMRES: Nonsymmetric linear system.*** *These panels compare the performance of MATLAB* `gmres` *(with and without restarting) against the sGMRES algorithm (where the basis* $\boldsymbol{B}$ *is computed by* k*-truncated Arnoldi with* $k = 4$*). The sparse linear system* $\boldsymbol{Ax} = \boldsymbol{f}$ *has dimension* $n = 921,632$*.* ***Left:*** *Relative residual and condition number* $\kappa_2(\boldsymbol{AB})$ *of the reduced matrix with the truncated Arnoldi basis.* ***Right:*** *Total runtime including basis generation.*

toolbox [20] to generate a finite-element discretization of the convection–diffusion (CD) equation

$$\begin{cases} -\varepsilon\nabla^2 u + \boldsymbol{w}\cdot\nabla u = 0, & \text{on } \Omega; \\ u = g, & \text{on } \partial\Omega. \end{cases}$$

Our example is based on the first CD test problem from Elman et al. [21, Ex. 6.1.1]. The domain is the unit square $\Omega = [-1,+1]^2$ with “hard” Dirichlet boundary conditions designed to produce a sharp interface in the solution. The viscosity parameter $\varepsilon = 10$ and the velocity $\boldsymbol{w} = (0,1)$, which corresponds to a diffusion-dominant regime. The dimension of the discretized linear system is $n = 2^{20} \approx 1.05 \cdot 10^6$.

We compare sGMRES with the MATLAB command `gmres` without restarting and with restarting frequencies $\{10, 30, 100\}$. The k-truncated Arnoldi basis (with $k = 2$) leads to a reduced matrix $\boldsymbol{AB}$ whose condition number grows significantly, but the condition number remains below the tolerance u^{-1} throughout the computation. This property ensures that sGMRES is effective.

Figure 1 illustrates that GRMES and sGMRES both solve the discretized CD problem with 14 digits of accuracy (as compared with the analytical solution) after building a basis with dimension $d \approx 2500$. For this problem instance, sGMRES runs about **70× faster** than GMRES. Restarted versions of GMRES do not attain accurate solutions, and they are still slower than sGMRES. For this 2-dimensional problem, the cost of sGMRES is similar to a sparse direct solver. For 3-dimensional CD equations, we anticipate that sGMRES would improve over the direct solver.

For another nonsymmetric example, we consider a linear system $\boldsymbol{Ax} = \boldsymbol{f}$ with sparse matrix `t2em` with dimension $n = 921,632$ from the SuiteSparse Matrix Collection [17]. The right-hand side $\boldsymbol{f}$ is generated via $\boldsymbol{f} = \boldsymbol{Ag}$, where $\boldsymbol{g}$ is a standard normal random vector. We solve this system using GMRES (with and without restarting) and using sGMRES (with 4-truncated Arnoldi). Figure 3 shows that GMRES and sGMRES both attain 13 digits of accuracy with a basis of dimension 3000, but sGMRES runs about 165× faster. Restarted GMRES is both slower and less accurate.

8.1.3. Symmetric linear systems. Surprisingly, for *symmetric* linear systems, sGMRES may even be competitive with CG and MINRES. As a simple example,

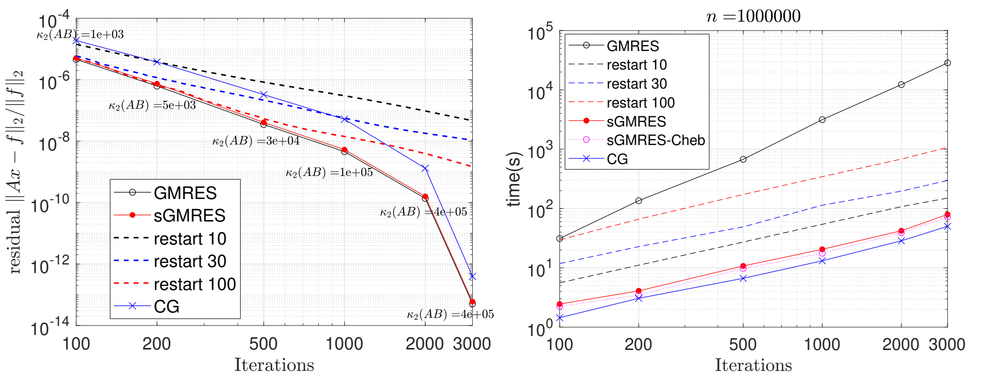

FIG. 4. ***sGMRES versus GMRES: Laplacian system.*** *For a 2D Laplacian system* $\boldsymbol{Ax} = \boldsymbol{f}$ *with dimension* $n = 10^6$*, these panels compare the performance of MATLAB* `gmres` *(with and without restarting) against the sGMRES algorithm (with 2-partial orthogonalization or the Chebyshev basis).* ***Left:*** *Relative residual and condition number* $\kappa_2(\boldsymbol{AB})$ *of the reduced matrix associated with the k-truncated Arnoldi basis.* ***Right:*** *Total runtime including basis generation.*

consider a symmetric test matrix $\boldsymbol{A}$ with dimension $n = 10^6$, obtained from a finite-difference approximation of the 2-dimensional Laplacian[7]. We solve $\boldsymbol{Ax} = \boldsymbol{f}$ where the right-hand side $\boldsymbol{f} = \boldsymbol{Ax}_0$ for a traceless vector $\boldsymbol{x}_0$. Although specialized algorithms (e.g., multigrid) are preferable to Krylov subspace methods, this example still offers an inspiring illustration of our methodology.

Figure 4 describes the progress of sGMRES when the basis is generated by k-truncated orthogonalization for $k = 2$ and when the basis is generated by the Chebyshev recurrence. We compare with the MATLAB commands `pcg` and `gmres` without restart and with restart frequencies in $\{10, 30, 100\}$. sGMRES achieves the same accuracy as GMRES, but the sketched version is up to **100× faster** after 3000 iterations. Restarted GMRES is both slower and less accurate. For comparison, d iterations of sGMRES take about 50% longer than d iterations of CG. Nevertheless, for the same number d of iterations, sGMRES achieves ℓ_2 residual norms up to **10× smaller** than CG. By this metric, the sGMRES method is more efficient than CG.

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-truncated 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. The k-truncated Arnoldi basis is still better conditioned, (see Figure 10). This experiment is intriguing because the Chebyshev basis can offer dramatic benefits in parallel computing environments [29, 46, 8, 13].

For indefinite systems, sGMRES can also be a valuable tool when MINRES is unstable due to loss of orthogonality in the Lanczos vectors [19]. We have found examples where sGMRES outperforms MINRES, but the difference was not dramatic.

8.1.4. sGMRES: Hard examples. We must acknowledge that sGMRES is not always an effective tool for solving linear systems. In some cases, sGMRES inherits

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

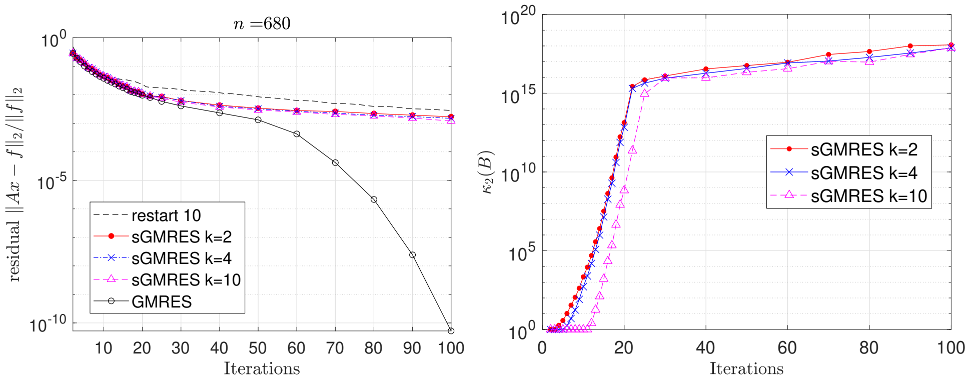

FIG. 5. ***sGMRES: Hard problems.*** *For some linear systems, it is expensive to construct a well-conditioned basis $\boldsymbol{B}$ for the Krylov subspace. When $\kappa_2(\boldsymbol{AB}) > u^{-1}$, the sGMRES algorithm may fail to match the GMRES algorithm with full orthogonalization.* ***Left:*** *Relative residual norms.* ***Right:*** *Condition number $\kappa_2(\boldsymbol{B})$ of the k-truncated Arnoldi basis. When $\kappa_2(\boldsymbol{B}) > u^{-1}$, the reported values are unreliable.*

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 [46, 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 5 for an illustration, which also shows that increasing the extent k of the truncation 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-truncated Arnoldi does not even guarantee monotone decrease of the condition number as k increases. This issue remains a challenge for sGMRES. The ideas from [4] may be useful, but they increase the arithmetic cost from $O(nd \log d)$ to $O(nd^2)$.

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

8.2.1. Algorithm details. Our implementation of sRR follows the pseudocode in Algorithm 1.2 with minor changes to facilitate comparison with the MATLAB `eigs` command. In one example, we use a single-vector Krylov subspace ($b = 1$), and we use k-truncated Arnoldi to form the basis. In another example, we consider a block Krylov subspace with a Chebyshev basis, as described in subsection 7.5. We do not use restarting or stabilization in these experiments. The subspace embedding is a modified SRFT (2.6) where $\boldsymbol{E}$ is diagonal Rademacher and $\boldsymbol{F}$ is a DCT4 matrix.

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.

8.2.2. Nonsymmetric eigenvalue problems. This section details the nonsymmetric eigenvalue problem that forms the basis for Figure 2. This computation is modeled on the classic trust-region subproblem (TRS) from optimization [16]:

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

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

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

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 a TRS instance based on the first test problem in [47]. We form the indefinite symmetric matrix $\boldsymbol{C} = \boldsymbol{L} - 5\mathbf{I}$, where $\boldsymbol{L}$ is the 2D Laplacian on a 710×710 square grid. The eigenvalue problem in (8.2) has dimension $n = 2 \cdot 710^2 > 10^6$. The constraint value $\Delta = 100$. The vector $\boldsymbol{g}$ is drawn from a standard normal distribution and rescaled so that $\|\boldsymbol{g}\|_2 = 0.1$.

We solve the TRS eigenvalue problem (8.2) with `eigs` and with sRR using 2-truncated Arnoldi. For both algorithms, the starting vector for the Krylov subspace is $\mathbf{0} \oplus \boldsymbol{g}$. The results appear in Figure 2. With a basis of dimension $d = 3000$, both methods construct eigenpairs with residuals below 10^{-10}. The solution trajectories are almost identical, but sRR is ultimately **10× faster**.

8.2.3. Symmetric eigenvalue problems. For symmetric eigenvalue problems, sRR remains a competitive algorithm because it can operate reliably with a general computational basis. In contrast, most Krylov methods expend a lot of energy to maintain near-orthogonality of the basis and to control the size of the basis. From a user's point of view, these issues manifest themselves in long runtimes or incorrect outputs for challenging problems (e.g., with clustered eigenvalues).

In this section, we describe a stylized application of sRR to a symmetric eigenvalue problem from data science. Consider the positive-semidefinite, normalized Laplacian matrix $\boldsymbol{L} \in \mathbb{R}^{n\times n}$ of an undirected graph on n vertices. This Laplacian matrix captures information about the geometric structure of the graph. To that end, it is valuable to compute an eigenspace of the Laplacian matrix associated with the *smallest* eigenvalues. This subspace can be used for dimension reduction, graph analysis, or graph signal processing tasks [11, 32, 43].

Krylov subspace methods can be used to find a few hundred of the smallest eigenpairs of a (sparse) Laplacian matrix $\boldsymbol{L}$. This approach is efficient because it only requires matrix–vector multiplies with the Laplacian. In contrast, approaches based on inverse iteration require us to solve linear systems in the Laplacian matrix, which is usually a nontrivial task.

We will compare several different methods that rely on a block Krylov subspace (7.1). As a baseline (RR), we form the basis using the block Lanczos method with full orthogonalization, and we solve the resulting block tridiagonal eigenvalue problem; this is essentially the same as applying RR. Second (Block-Lan), we form the basis using block Lanczos without orthogonalization, and we solve the block tridiagonal eigenvalue problem. Third (sRR-BLan), we form the basis with block Lanczos

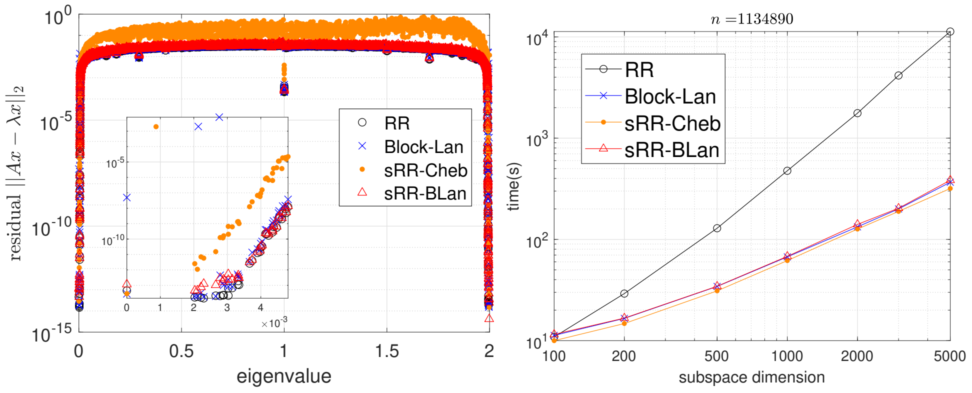

FIG. 6. ***Laplacian eigenspaces.*** *For a normalized graph Laplacian matrix with dimension* $n = 1,134,890$*, these panels illustrate eigenvalue computations using a block Krylov subspace with block size* $b = 10$*.* ***Left:*** *Residuals versus computed eigenvalues when the subspace dimension* $d = bp = 5000$*.* ***Inset:*** *Enlargement of the eigenvalues near zero.* ***Right:*** *Runtime with varying dimension d, including basis generation.*

without orthogonalization, and we use sRR to solve the eigenvalue problem. Last (sRR-Cheb), we form the basis using block Chebyshev without orthogonalization, and we invoke sRR to solve the eigenvalue problem.

As an example, we will compute eigenvalues and eigenvectors of the Laplacian of the YouTube social network graph ($n = 1,134,890$) from the SNAP data set [33], after preprocessing to remove vertices with no edges. We consider bases with block size $b = 10$ and depths from $p = 10$ to $p = 500$; the subspace size $d = bp$. The results appear in Figure 6. For the largest subspace, sRR-Cheb is **35× faster** than RR, while sRR-BLan and Block-Lan are **28× faster**. Among the methods, RR is the most accurate, but sRR-BLan enjoys similar performance. Meanwhile, both Block-Lan and sRR-Cheb exhibit some instability, producing several "ghost eigenvalues" near zero with large residuals. The reason is that Block-Lan requires a nearly orthogonal basis, but the computed basis has condition number around 10^7. Likewise, sRR-Cheb stumbles because it computes a basis whose condition number exceeds 10^{15}. Even so, sRR-Cheb allows us to estimate residuals and remove ghost eigenvalues inexpensively, whereas it is costly to identify ghost eigenvalues with Block-Lan.

We can also study the performance of these methods for smaller approximation subspaces. When $bp \leq 3,000$, all methods produce eigenpairs whose residuals have comparable size (Figure 7, left panel). On the other hand, we need to use larger approximation subspaces to obtain smaller residuals. As we increase the size of the subspace, the behavior of the methods begins to diverge (Figure 7, right panel).

Figure 8 illustrates the condition numbers of the computed bases. It is worth emphasizing that sRR succeeds when $\kappa_2(\boldsymbol{B}) < u^{-1}$ (which is violated at $bp = 4,000$). In contrast, Block-Lan requires $\kappa_2(\boldsymbol{B}) \approx 1$ to be stable. After 500 steps of block Lanczos $bp = 5,000$ we have $\kappa_2(\boldsymbol{B}) \approx 3 \times 10^7$. This conditioning can be devastating for Block-Lan[8], but it is not an issue for sRR. Indeed, sRR-BLan reliably finds the relevant eigenpairs with good accuracy. Furthermore, with sRR, we can easily detect instability by evaluating the condition number $\kappa_2(\boldsymbol{SB})$ of the sketched basis.

[8]Perhaps for this reason, `eigs` (based on Lanczos $b = 1$) enforces orthogonality to ensure that the Lanczos method remains robust, resulting in high stability at the cost of reduced efficiency. In

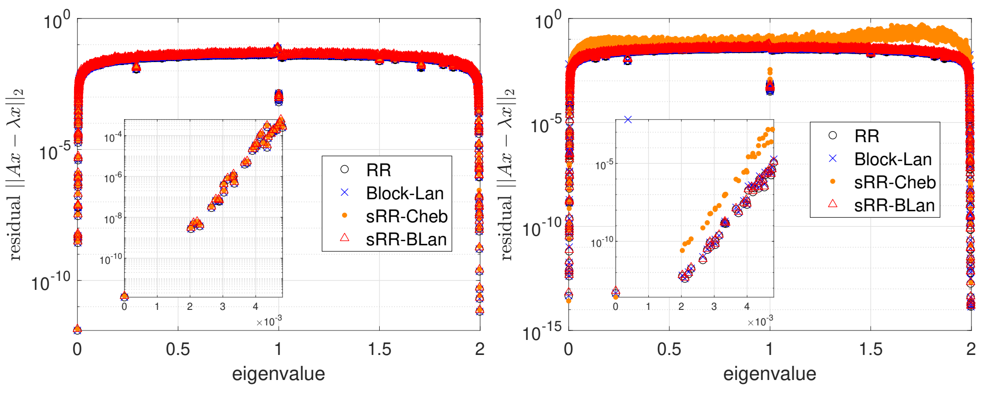

FIG. 7. ***Laplacian eigenspaces, more residual plots.*** *In the setup as in Figure 6, these plots show the eigenpair residuals when* $bp = 3000$ *(left) and* $bp = 4000$ *(right).*

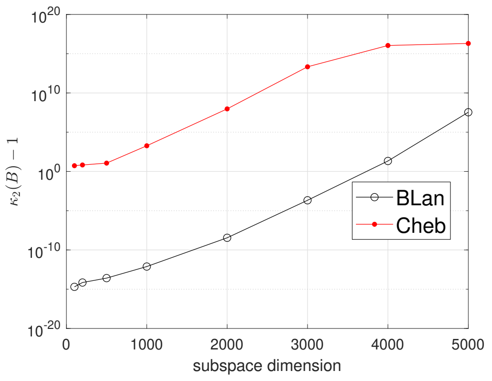

FIG. 8. ***Laplacian eigenspaces, condition number of bases.*** *This plot shows* $\kappa_2(\boldsymbol{B})$ *in the same setup as Figures 6 and 7. Block Lanczos requires* $\kappa_2(\boldsymbol{B}) \approx 1$ *to guarantee accuracy, whereas for sRR it suffices to have* $\kappa_2(\boldsymbol{B}) < u^{-1}$.

Figure 9 shows the results of the same experiment with the larger block size $b = 100$. We see clearer benefits of using the block Chebyshev basis: sRR-Cheb runs faster than sRR-BLan and achieves the same accuracy. On the other hand, we can obtain smaller residuals with the smaller block size $b = 10$.

8.2.4. More experiments with sRR. Let us present another example of a symmetric eigenvalue problem. The matrix $\boldsymbol{A}$ is the 2-dimensional 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 basic Lanczos recurrence (i.e., 2-truncation without extra 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 10 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. Completing 2000 iterations, sRR runs **12× faster** than `eigs`. The difference is likely because `eigs` enforces orthogonality to ensure that the Lanczos

the experiments here, RR can be regarded as a block version of `eigs`.

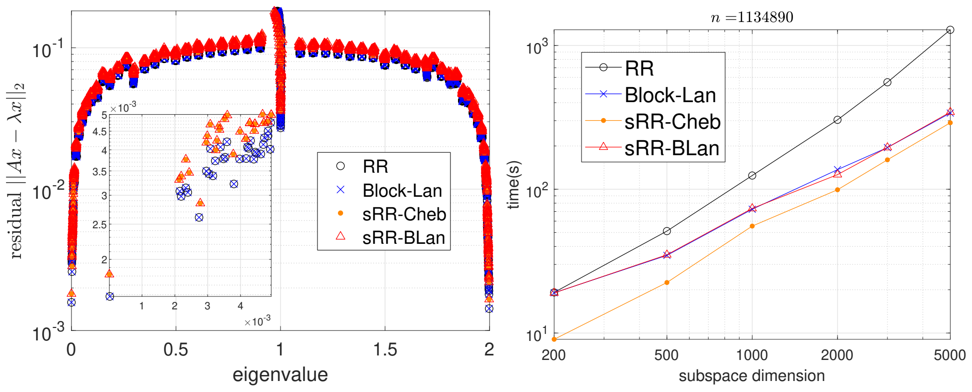

FIG. 9. ***Laplacian eigenspaces, block size $b = 100$.*** *This experiment parallels Figure 6 with block size $b = 100$.*

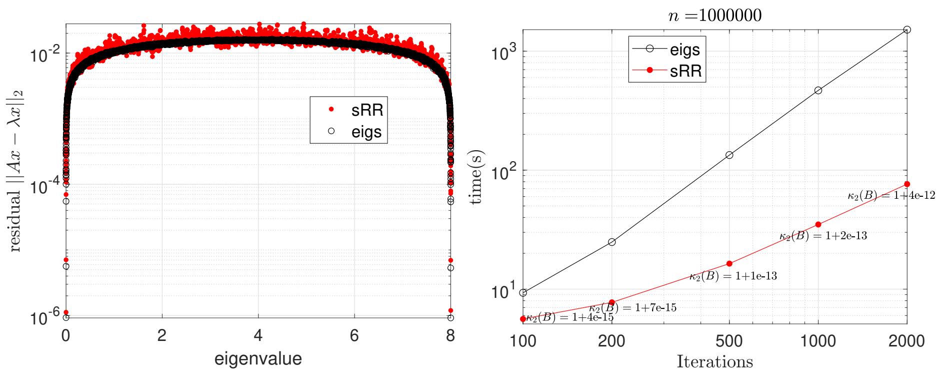

FIG. 10. ***sRR versus RR: Symmetric eigenvalue problem.*** *These panels compare the performance of MATLAB* **eigs** *(without restarting) against the sRR algorithm (where the basis $\boldsymbol{B}$ is computed by the Lanczos recurrence). The sparse, symmetric eigenvalue problem $\boldsymbol{Ax} = \lambda\boldsymbol{x}$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

method remains robust. In this instance, the basic Lanczos method already constructs a basis that is almost orthogonal, but sRR does not require this property to succeed.

Lack of ghost eigenvalues with sRR. A feature worth emphasizing in the previous experiments is that the sRR avoids computing *ghost eigenvalues*, which are repeated estimates of a single eigenvalue [18, Ch. 7]. 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. (Selective) orthogonalization is a traditional remedy [55, 56], 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. Figure 11 illustrates this point.

A possible explanation for the lack of ghost eigenvalues is that they are caused by treating a nonorthonormal basis $\boldsymbol{B}$ as orthonormal and performing the RR process. This corresponds to taking $\boldsymbol{B}^T\boldsymbol{AB} \approx \boldsymbol{J}_p$ in (4.1), which is a poor approximation unless $\boldsymbol{B}$ is nearly orthonormal. By contrast, sRR treats $\boldsymbol{B}$ as a general basis matrix

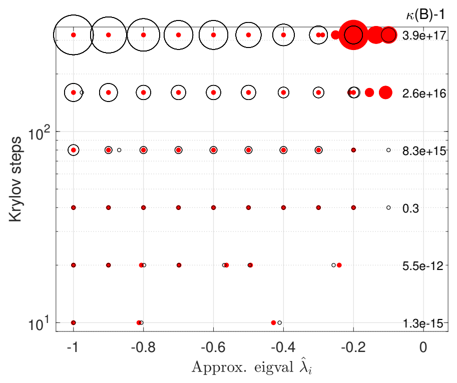

FIG. 11. ***Lack of ghost eigenvalues with sRR.*** *The figure illustrates the location and near-multiplicity of the computed eigenvalues as we vary the Lanczos steps. Black circle: Lanczos (without reorthogonalization), red dots: sRR. The size of the circles and dots represent the multiplicity (redundancy) of the computed eigenvalues, counted as the number of computed eigenvalues within distance* 10^{-2}. *The matrix* $\boldsymbol{A}$ *has exact eigenvalues at* $-0.1i$, $1 \le i \le 10$ *(plus* $2^{13} - 10$ *equispaced in* $[0, 1]$*). Without reorthogonalization, Lanczos starts to suffer from loss of orthogonality, resulting in the appearance of ghost eigenvalues. sRR, by contrast, largely removes this pitfall, especially in the extremal eigenvalues. With stabilized sRR (6.13) (not shown), ghost eigenvalues disappear even near 0.*

and avoids this pitfall.

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

Without loss of generality, we consider a diagonal matrix $\boldsymbol{A}$. We let the dimension $n = 2^{19}$, and the eigenvalues are ten equispaced 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 (6.13)) to compute eigenpairs of $\boldsymbol{A}$.

Figure 12 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 while $\kappa_2(\boldsymbol{B}) \lesssim u^{-1}$. When the condition number of the basis is larger, sRRstab even produces more accurate estimates for the interior eigenpairs than RR does. Excluding the cost of basis generation, sRR runs up to 15× faster than classic RR. The stabilized algorithm usually runs faster than RR, but the relatively large constant in the $O(d^3)$ cost of the SVD and QZ algorithms dominates the runtime as the basis dimension d grows.

9. Variations and extensions.

The ideas underlying sGMRES and 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:

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

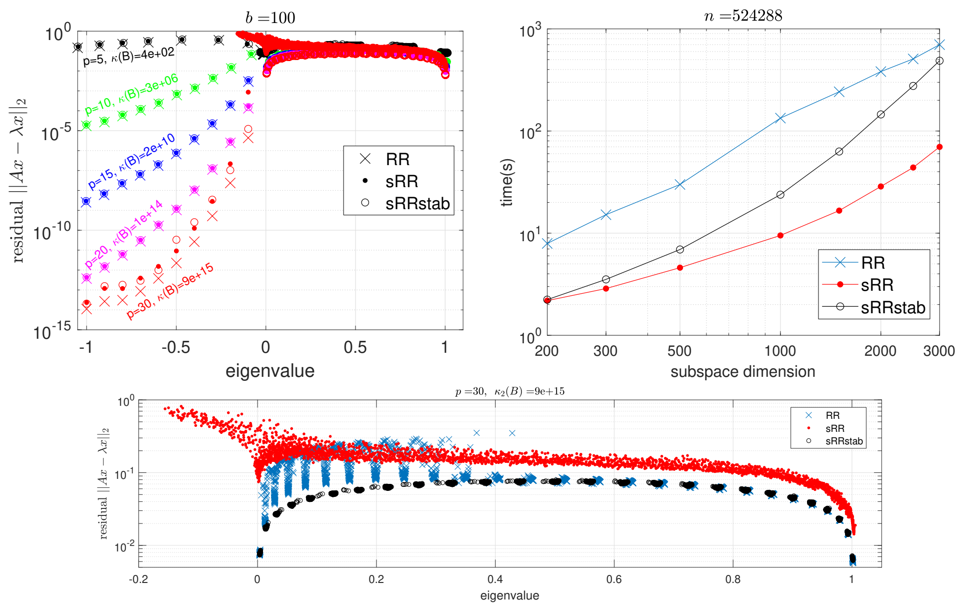

FIG. 12. ***sRR: Ill-conditioned bases and stabilization.*** *This diagram shows how eigenvalue residuals improve with the depth p of a block Krylov–Chebyshev subspace $\boldsymbol{B} \in \mathbb{R}^{n\times(bp)}$ with block size $b = 100$.* ***Left:*** *Residuals as a function of eigenvalue estimates, along with the condition number $\kappa_2(\boldsymbol{B})$ 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 $[0, 1]$.*

Given a solution $\boldsymbol{M}_\star = (\boldsymbol{JB})^\dagger(\boldsymbol{HB})$ 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 $(\boldsymbol{By}, \theta)$ to (9.1).

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

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

The solution $\hat{\boldsymbol{M}} = (\boldsymbol{SJB})^\dagger(\boldsymbol{SHB})$. 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 [26, 36]. Using the new insights from our paper, we can now accelerate these algorithms by sketching. The resulting techniques share some genes with sketch-based algorithms for low-rank matrix approximation [68, 65, 66, 39], 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 [26]. If we form an adapted basis $\boldsymbol{B}$ by means of subspace iteration [48, 26] or block Krylov methods [48, 25, 38, 63], we obtain much better approximations, as described in the cited work.

Let $\boldsymbol{S} \in \mathbb{C}^{s \times n}$ be an "affine space" embedding [69] 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

$$\hat{\boldsymbol{A}}_{\text{sketch}} = (\boldsymbol{AB})(\boldsymbol{SAB})^\dagger (\boldsymbol{SA}). \tag{9.6}$$

The formula (9.6) is wholly unsuitable for practical computation, but it can be replaced with a stable and efficient variant [39]. If we choose $\boldsymbol{B}$ to be a second sketching map, we obtain the (low-accuracy) sketched SVD algorithms from [68, 65, 39].

Our work delivers a novel insight. Introducing 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 [26] or randomized block Krylov methods [38, 63].

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, 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.

Second, 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 [9]. For example, we can trivially parallelize the computation of block Krylov subspaces. While our experiments take place in 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. Truncated 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 Oleg Balabanov, Alice Cortinovis, Ethan Epperly, Laura Grigori, Ilse Ipsen, Daniel Kressner, Gunnar Martinsson, Maike Meier, Florian Schaefer, Daniel Szyld, Alex Townsend, Nick Trefethen, and Rob Webber for valuable discussions and feedback. We are grateful to David Silvester for his help with the IFISS toolbox and to Zdeněk Strakoš for sharing his insights on Krylov subspace methods and the prospects for sGMRES and sRR.

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] O. Balabanov and L. Grigori, *Randomized block Gram–Schmidt process for solution of linear systems and eigenvalue problems.* Available at https://arXiv.org/abs/2111.14641, Nov. 2021.
[6] O. Balabanov and A. Nouy, *Randomized linear algebra for model reduction. Part I: Galerkin methods and error estimation*, Adv. Comput. Math., 45 (2019), pp. 2969–3019, https://doi.org/10.1007/s10444-019-09725-6.
[7] O. Balabanov and A. Nouy, *Randomized linear algebra for model reduction—part II: minimal residual methods and dictionary-based approximation*, Adv. Comput. Math., 47 (2021), pp. Paper No. 26, 54, https://doi.org/10.1007/s10444-020-09836-5.
[8] 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.
[9] 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.
[10] B. Beckermann, *The condition number of real Vandermonde, Krylov and positive definite Hankel matrices*, Numer. Math., 85 (2000), pp. 553–577.
[11] M. Belkin and P. Niyogi, *Laplacian eigenmaps for dimensionality reduction and data representation*, Neural Computation, 15 (2003), pp. 1373–1396.
[12] 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.
[13] T. Chen and E. Carson, *Predict-and-recompute conjugate gradient variants*, SIAM J. Sci. Comp, 42 (2020), pp. A3084–A3108.
[14] 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.
[15] 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.
[16] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust Region Methods*, vol. 1, SIAM, Philadelphia, PA, USA, 2000.
[17] T. A. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Trans. Math. Soft., 38 (2011), pp. 1–25.
[18] J. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, USA, 1997.
[19] T. A. Driscoll, K.-C. Toh, and L. N. Trefethen, *From potential theory to matrix iterations in six steps*, SIAM Rev., 40 (1998), pp. 547–578.
[20] H. C. Elman, A. Ramage, and D. J. Silvester, *IFISS: A computational laboratory for investigating incompressible flow problems*, SIAM Rev., 56 (2014), pp. 261–273.
[21] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*, Oxford University Press, USA, 2014.
[22] W. Gautschi, *The condition of orthogonal polynomials*, Math. Comp., 26 (1972), pp. 923–924, https://doi.org/10.2307/2005876.
[23] W. Gautschi, *The condition of polynomials in power form*, Math. Comp., 33 (1979), pp. 343–352, https://doi.org/10.2307/2006047.
[24] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, 4th ed., 2012.
[25] N. Halko, P.-G. Martinsson, Y. Shkolnisky, and M. Tygert, *An algorithm for the princi-*

pal component analysis of large data sets, SIAM J. Sci. Comput., 33 (2011), pp. 2580–2594, https://doi.org/10.1137/100804139.

[26] 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.

[27] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, PA, USA, second ed., 2002.

[28] 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.

[29] 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.

[30] W. Kahan, *Spectra of nearly Hermitian matrices*, Proc. Amer. Math. Soc., 48 (1975), pp. 11–17.

[31] A. V. Knyazev, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM J. Sci. Comp, 23 (2001), pp. 517–541.

[32] E. Kokiopoulou, J. Chen, and Y. Saad, *Trace optimization and eigenproblems in dimension reduction methods*, Numerical Linear Algebra with Applications, 18 (2011), pp. 565–602, https://doi.org/10.1002/nla.743.

[33] J. Leskovec and A. Krevl, *SNAP Datasets: Stanford large network dataset collection*. http://snap.stanford.edu/data, June 2014.

[34] J. Liesen and Z. Strakoš, *Krylov subspace methods*, Numerical Mathematics and Scientific Computation, Oxford University Press, Oxford, 2013. Principles and analysis.

[35] T. A. Manteuffel, *The Tchebychev iteration for nonsymmetric linear systems*, Numer. Math., 28 (1977), pp. 307–327, https://doi.org/10.1007/BF01389971.

[36] P.-G. Martinsson and J. A. Tropp, *Randomized numerical linear algebra: Foundations and algorithms*, Acta Numerica, (2020), pp. 403––572.

[37] 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.

[38] 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.

[39] Y. Nakatsukasa, *Fast and stable randomized low-rank matrix approximation*, arXiv:2009.11392, (2020).

[40] Y. Nakatsukasa, *Sharp error bounds for Ritz vectors and approximate singular vectors*, Math. Comp., 89 (2020), pp. 1843–1866.

[41] Y. Nakatsukasa and J. A. Tropp, *Fast and accurate randomized algorithms for linear systems and eigenvalue problems*, ACM Report 2021-01, California Institute of Technology, 2021, https://doi.org/10.7907/cmyh-va31.

[42] 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.

[43] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, *Graph signal processing: Overview, challenges, and applications*, Proceedings of the IEEE, 106 (2018), pp. 808–828, https://doi.org/10.1109/JPROC.2018.2820126.

[44] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[45] B. N. Parlett, *The Symmetric Eigenvalue Problem*, SIAM, Philadelphia, 1998.

[46] B. Philippe and L. Reichel, *On the generation of Krylov subspace bases*, Appl. Numer. Math., 62 (2012), pp. 1171–1186.

[47] M. Rojas, S. A. Santos, and D. C. Sorensen, *Algorithm 873: LSTRS: MATLAB software for large-scale trust-region subproblems and regularization*, ACM Trans. Math. Soft., 34 (2008), pp. 11:1–11:28, http://doi.acm.org/10.1145/1326548.1326553.

[48] V. Rokhlin, A. Szlam, and M. Tygert, *A randomized algorithm for principal component analysis*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 1100–1124.

[49] V. Rokhlin and M. Tygert, *A fast randomized algorithm for overdetermined linear least-squares regression*, Proc. Natl. Acad. Sci., 105 (2008), pp. 13212–13217.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

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

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

[57] G. L. G. Sleijpen and H. A. VanderVorst, *A Jacobi-Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.

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

[59] G. W. Stewart, *A Krylov-Schur algorithm for large eigenproblems*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 601–614.

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

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

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

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

[64] J. A. Tropp, *Randomized block Krylov methods for approximating extreme eigenvalues*, Numer. Math., 150 (2022), pp. 217–255, https://doi.org/10.1007/s00211-021-01250-3.

[65] 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.

[66] 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), pp. A2430–A2463.

[67] A. van der Sluis, *Condition numbers and equilibration of matrices*, Numer. Math., 14 (1969/70), pp. 14–23, https://doi.org/10.1007/BF02165096.

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

[69] D. P. Woodruff, *15-859: Algorithms for big data*. Course notes, 2020.

[70] 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.

[71] 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.