A GOLUB-KAHAN DAVIDSON METHOD FOR ACCURATELY COMPUTING A FEW SINGULAR TRIPLETS OF LARGE SPARSE MATRICES [*]

STEVEN GOLDENBERG [†]

In collaboration with: Andreas Stathopoulos, Eloy Romero

Abstract. Obtaining high accuracy singular triplets for large sparse matrices is a significant challenge, especially when searching for the smallest triplets. Due to the difficulty and size of these problems, efficient methods must function iteratively, with preconditioners, and under strict memory constraints. In this research, we present a Golub-Kahan Davidson method (GKD), which satisfies these requirements and includes features such as soft-locking with orthogonality guarantees, an inner correction equation similar to Jacobi-Davidson, locally optimal +k restarting, and the ability to find real zero singular values in both square and rectangular matrices. Additionally, our method achieves full accuracy while avoiding the augmented matrix, which often converges slowly due to the difficulty of interior eigenvalue problems. We describe our method in detail, including implementation issues that may arise. Our experimental results confirm the efficiency and stability of our method over the current implementation of PHSVDS in the PRIMME software package [19].

Key words. Singular Value Decomposition, Iterative Methods

1. Introduction.

Assuming a large sparse matrix, $A \in \Re^{m,n}$ with $m \geq n$, the economy size singular value decomposition (SVD) is given by

$$A = U \Sigma V^T, \tag{1}$$

where $U \in \Re^{m,n}$ and $V \in \Re^{n,n}$ are orthonormal bases and $\Sigma = diag(\sigma_1, \dots, \sigma_n) \in \Re^{n,n}$ with $\sigma_1 \leq \sigma_2 \leq \cdots \leq \sigma_n$ is a diagonal matrix containing the singular values of A. The singular triplets of A are defined as (u_i, σ_i, v_i) given by the SVD. This decomposition has become increasingly important and is frequently used in fields like statistics for principal component analysis [9], computer science for image compression [14] and web search clustering [12], and genomics for expression data processing [1]. More specifically, finding the smallest singular triplets is useful for total least squares problems and the determination of the effective rank of a matrix [6].

When the matrix A is large enough, it can be difficult to compute the SVD with dense methods. Furthermore, applications often require only a few of the largest or smallest singular values and vectors. These observations have lead to iterative algorithms like Golub-Kahan-Lanczos (GKL) also known as Lanczos bidiagonalization. However, when the solution requires many iterations, it may be infeasible to store the previous vectors necessary for GKL with full or partial reorthogonalization. To solve this, restarted versions of GKL that limit the maximum basis size such as IRLBA [2] have been developed. Additionally, other restarted methods have emerged, such as Jacobi-Davidson (JDSVD), the Preconditioned Hybrid SVD method (PHSVDS), and the Preconditioned Locally Minimal Residual method (PLMR_SVD). These methods can also take advantage of preconditioning, which can provide significant speedups for difficult problems.

[*]This work is supported by NSF under grant No. ACI SI2-SSE 1440700, and by DOE under an ECP grant.

[†]College of William and Mary, (sgoldenberg@email.wm.edu).

In general without preconditioning or +k restarting, these methods build Krylov spaces on the normal equations matrix $C = A^TA$ or on the augmented matrix,

$$B = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}. \quad (2)$$

We denote a k-dimensional Krylov space on matrix A with initial vector v_1 by $K_k(A, v_1) = span\{v_1, Av_1, \ldots, A^{k-1}v_1\}$. Additionally, $\|\cdot\|$ denotes the Euclidean norm and $\epsilon_{mach} = 2.2\text{E-}16$ denotes the machine precision. Frequently, methods that build their search space with B, like JDSVD and PLMR_SVD, are able to achieve accuracy of $\|r_B\| < O(\|A\|\epsilon_{mach})$ when searching for the smallest singular triplets, where r_B is the eigenvalue residual on B. This is directly related to the left and right singular value residuals $r_u = A^Tu - \sigma v$ and $r_v = Av - \sigma u$ as $r_B = [r_u; r_v]$. However, this approach mirrors the singular values of A across zero [13]. Therefore, searching for the smallest singular triplets is a highly interior problem which can slow convergence. Worse, when A is rectangular, the spectrum of B contains $m - n$ zero eigenvalues that are not in the spectrum of A. Therefore, methods on B are unable to determine real zero singular values of A when $m \neq n$.

Alternatively, methods that build $K_k(C, v_1)$ explicitly are only able to achieve accuracy $O(\|C\|\epsilon_{mach}) = O(\|A\|^2\epsilon_{mach})$. for the eigenvalue residual on C, r_C. Additionally, r_C can be related to the left singular residual, r_u, by the following equation,

$$r_C = A^TAv - \sigma^2 v = \sigma(A^Tu - \sigma v) = \sigma r_u. \quad (3)$$

Thus, if $\sigma_1 \neq 0$, the norm of the singular value residual when searching for the smallest singular value cannot be better than $O(\|A\|\kappa(A)\epsilon_{mach})$, where $\kappa(A) = \frac{\sigma_n}{\sigma_1}$ is the condition number of A. Despite the squaring of the spectrum, these methods usually converge faster than methods on B, both in theory and in practice, due to the extremal problem they solve. Furthermore, these methods are often able to find real zero singular values of A, as the corresponding eigenproblem on C does not introduce extraneous zero eigenvalues.

In this work, we introduce a Golub-Kahan Davidson method (GKD), which attempts to keep the convergence of methods on C, but attain the full accuracy of methods on B. We define full accuracy to be $\sqrt{\|r_u\|^2 + \|r_v\|^2} < \|A\|\epsilon_{mach}$. First, we discuss related methods such as GKL, JDSVD, PLMR_SVD and PHSVDS, followed by a detailed description of our method including implementation details. Lastly, we provide experimental results that highlight the capabilities of GKD compared to the current implementation of PHSVDS in the PRIMME software package.

1.1. Related Work. GKL [11] builds two spaces including the same space as eigenmethods on C, $K_k(A^TA, v_1)$, but it avoids directly multiplying vectors with A^TA. By doing this, it also avoids the numerical problems associated with working on C. Without any additional matrix vector multiplications (matvecs), it also builds $K_k(AA^T, Av_1)$. This is done by keeping two orthogonal spaces, U and V, where the last vector of V, v_k, is used to expand U as $u_k = Av_k$ and the last vector of U, u_k, is used to expand V as $v_{k+1} = A^Tu_k$. These new vectors are orthonormalized to the previous ones and the coefficients from this process are used to create the bidiagonal projection matrix U^TAV. GKL solves the smaller singular value problem on this projection matrix to approximate the singular triplets. While GKL is considered to be one of the most accurate and effective algorithms for finding small singular triplets, the standard version is unrestarted and cannot be preconditioned. Therefore, GKL tends to be computationally slow for poorly separated triplets of large matrices. Many restarted versions have been developed [3, 2, 8], but they are unable to maintain the convergence of the unrestarted method and they are generally slower than state-of-the-art eigenmethods for the smallest singular triplets. Additionally, restarted versions of GKL use implicit or thick restarting [18], without the locally optimal restarting feature that has been shown to be effective for eigenvalue problems [10] and is currently used in PRIMME as +k restarting.

JDSVD [7] works on B by using two independent subspaces rather than one. Without using preconditioning or solving the correction equation, JDSVD builds subspaces that span the following Krylov spaces:

$$U_k = K_{\frac{k}{2}}(AA^T, u_1) \oplus K_{\frac{k}{2}}(AA^T, Av_1), \quad V_k = K_{\frac{k}{2}}(A^TA, v_1) \oplus K_{\frac{k}{2}}(A^TA, A^Tu_1). \tag{4}$$

These spaces are similar to the ones used in GKL, but crucially, each space is the sum of two different spaces of half dimension. This allows JDSVD to take advantage of initial guesses for both the left and right singular vectors. However, if we choose initial vectors that satisfy $v_1 = A^Tu_1$, the outer iteration of JDSVD becomes wasteful, as it builds exactly the same space of GKL with half the dimension. This is also true of eigensolvers on B as seen below,

$$B^2 \begin{bmatrix} v \\ Av \end{bmatrix} = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}^2 \begin{bmatrix} v \\ Av \end{bmatrix} = \begin{bmatrix} A^TAv \\ AA^T(Av) \end{bmatrix}. \tag{5}$$

The inner correction equation used in JDSVD often allows for faster convergence than standard eigenvalue methods on B. Since JDSVD works on B, it can achieve full accuracy, but suffers from the same issues as other eigenmethods on B.

PHSVDS [20] exploits the different advantages of eigenmethods on B and C by utilizing each in a two-stage method. The first stage can use any state-of-the-art eigensolver on C, which gives it fast convergence until either the user tolerance is met or until switching to a second stage using an eigensolver on B is necessary to reach the remaining user tolerance. Switching to an eigensolver on B after a fully converged first stage can effectively utilize good initial guesses from the first stage on C and thus avoid resolving the entire accuracy on an indefinite problem. Its implementation in PRIMME can use any of the two near-optimal eigensolvers, GD+k or JDQMR. This two-stage approach has been shown to be faster than eigensolvers on B alone, and typically has better performance than other SVD methods.

While PHSVDS has shown significant improvements, it is still limited by the speed of eigensolvers on B when the matrix is ill-conditioned. It converges quite well for problems that do not need to switch stages, but eigensolvers on C cannot converge to high accuracy if the smallest singular value is nearly 0. Once it switches to the second stage on B, a significant slowdown occurs associated with interior problems and methods based on the augmented matrix. Obviously, an improved algorithm would converge with the near-optimal speed of GD+k on C down to $O(\|A\|\epsilon_{mach})$.

Recently, PLMR_SVD [17] was developed, which is based on a stationary iteration that uses two separate four-term recurrences to build the following spaces,

$$span\{v^{(i)}, r_u^{(i)}, P(A^Tr_v^{(i)} - \sigma r_u^{(i)}), v^{(i-1)}\}$$

$$span\{u^{(i)}, r_v^{(i)}, P(Ar_u^{(i)} - \sigma r_v^{(i)}), u^{(i-1)}\},$$

where $v^{(i)}$ and $u^{(i)}$ are the i-th approximations of the right and left singular vectors respectively, and $r_v^{(i)} = P(Av - \sigma u)$ and $r_u^{(i)} = P(A^T u - \sigma v)$ are their preconditioned right and left residuals respectively. Without a preconditioner, these spaces match those of GD+1 on B when we restrict GD to a max basis of 4 vectors. There may be additional benefits to building the spaces separately, but PLMR_SVD lacks the subspace acceleration present in GD and JDSVD, which can provide superlinear convergence.

2. Main Contribution.

We believe that creating a restarted and preconditioned analogue to GKL will improve performance as long as we carefully choose our extraction and restarting methods to avoid losing key directions for convergence. This leads us to the following algorithm for GKD.

2.1. Algorithm.

Our algorithm is designed to mimic the nature of GKL by keeping two orthogonal spaces, V and Q, which are built without multiplying directly with $A^T A$. Instead, we build Q such that $AV = QR$ is the economy QR factorization of AV. Then, we extend V with a left residual based on a Galerkin extraction from R. Without preconditioning or +k restarting, this process builds the spaces $K_q(A^T A, v_1)$ and $K_q(AA^T, v_1)$ after q iterations or $2q$ matvecs like GKL, where both the extraction of approximate triplets and expansion of the spaces avoid a direct multiplication with C. This helps us to avoid the squaring of the norm and condition number that occurs with eigensolvers on C.

Specifically, we extract approximate singular triplets from these spaces using a Rayleigh-Ritz procedure that is adapted for the SVD. Given search spaces $\mathcal{Q} \subset \mathbb{R}^{\mathrm{m}}$ and $\mathcal{V} \subset \mathbb{R}^{\mathrm{n}}$, we can determine approximations (u, σ, v) with the following two Galerkin conditions on the right and left residuals,

$$\begin{aligned} Av - \sigma u &\perp \mathcal{Q}, \\ A^T u - \sigma v &\perp \mathcal{V}. \end{aligned} \tag{6}$$

Since $u \in \mathcal{Q}$ and $v \in \mathcal{V}$, we can write $u = Qx$ and $v = Vy$, where Q and V form k-dimensional orthonormal bases of $\mathcal{Q}$ and $\mathcal{V}$ respectively. Additionally, $AV = QR \Rightarrow Q^T AV = R$, which allows us to rewrite the conditions as follows:

$$\begin{aligned} Q^T AVy = \sigma Q^T Qx &\Rightarrow Ry = \sigma x \\ V^T A^T Qx = \sigma V^T Vy &\Rightarrow R^T x = \sigma y. \end{aligned} \tag{7}$$

Therefore, solving the singular value decomposition on R with singular triplets (x, σ, y) satisfies both constraints and provide us approximations to the singular triplets of A.

As in Generalized Davidson (GD) [5], we take the approximations from this Rayleigh-Ritz extraction and use them to form the left residual $r_u = A^T u - \sigma v$. Then, we can choose to expand V with this residual directly, or with the preconditioned residual Pr_u where P is a suitable preconditioner for $A^T A$. Unlike the JDSVD method, the space Q is expanded with Av_{i+1} rather than a preconditioned right residual. Note that our left residual is exactly $r_u = r_C/\sigma$ and since

$$V^T A^T AVy = \sigma y \Rightarrow R^T Ry = \sigma y,$$

GKD is equivalent to GD solving the eigenproblem on $A^T A$ in exact arithmetic. Moreover, without preconditioning or restarting, it is also equivalent to GKL. However, GKD only shares numerical properties with GKL, whereas the accuracy of GD on

C is limited by the matrix on which it works. Combining this with thick and +k restarting gives us Algorithm 1 for seeking one singular triplet. This algorithm can easily be extended to find multiple singular triplets by using a locking method.

Algorithm 1 GKD Iteration

```
 1: Define target $\tilde{\sigma}$, initial vector $v_1$, max basis size $q$, tolerance $\delta$, preconditioner $P$, and $i = 1$
 2: Build $V = [v_1]$, $Q = [\frac{Av_1}{\|Av_1\|}]$, and $R = \|Av_1\|$
 3: while $\sqrt{\|r_u\|^2 + \|r_v\|^2} > \|A\|\delta$ do
 4:     while $i < q$ do
 5:         Compute SVD of $R$
 6:         Choose the singular triplet $(x, \sigma_r, y)$ nearest to the target $\tilde{\sigma}$
 7:         Save $v_{old} = y$ for +k restarting
 8:         Set $u = Q(:, 1:i)x$, $v = V(:, 1:i)y$
 9:         Compute left residual: $r_u = A^T u - \sigma_r v$
10:         $V(:, i+1) = Pr_u$
11:         Orthogonalize $V(:, i+1)$ against $V(:, 1:i)$
12:         $Q(:, i+1) = AV(:, i+1)$
13:         Orthogonalize $Q(:, i+1)$ against $Q$ and update $R(:, i+1)$
14:         $i = i + 1$
15:     end while
16:     call Algorithm 2 to restart
17: end while
```

Inner-outer solvers like JDSVD and the JDQMR implementation in PRIMME utilize extra matvecs inside of an inner solver as a refinement step to improve the convergence speed of the outer iterations by solving a linear system. These methods can provide a significant speedup in time for problems that have a relatively inexpensive matrix-vector multiplication. GKD can be extended to a Jacobi-Davidson variant, GKJD, that solves the correction equation

$$(I - vv^T)(A^T Ax - \sigma^2 x)(I - vv^T) = -r_u \tag{8}$$

instead of applying a preconditioner at line 10 of Algorithm 1. The inner solver is based on the symmetric Quasi-Minimal Residual method (QMRs) used in PRIMME's JDQMR. Additionally, we include most of the dynamic stopping conditions used in PRIMME to stop QMRs in a near-optimal way [16].

2.2. Restarting and Locking. Our restart procedure takes the current best approximations to the s singular triplets closest to the target, $\tilde{\sigma}$, and uses them together with those from the +k restarting to compress V, Q and R down to dimension $s + k$. The steps for building the restarted V are seen in lines 1-7 of Algorithm 2.

The simplest method to restart Q and R is to set them as $Q\tilde{Q}$ and $\tilde{R}$ respectively, where $Rt = \tilde{Q}\tilde{R}$ is the QR factorization of Rt with $t = [Y_1, v_{new}]$ from line 6 of Algorithm 2. This can introduce numerical error of magnitude $O(\|R\|\epsilon_{mach})$, which can be as large as $O(\|A\|\epsilon_{mach})$. However, this error accumulates over many restarts, eventually causing loss of convergence. It is possible to intelligently compute Q and R to avoid direct multiplications with R through the already available SVD of R as seen below,

$$\begin{aligned} AVt = QRt &= Q \begin{bmatrix} X_1 & X_2 \end{bmatrix} \begin{bmatrix} \Sigma_r^{(1)} & 0 \\ 0 & \Sigma_r^{(2)} \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & Y_2^T v_{old} \end{bmatrix} \\ &= Q \begin{bmatrix} X_1 & X_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_r^{(2)} Y_2^T v_{old} \end{bmatrix}. \end{aligned} \tag{9}$$

Algorithm 2 Restart Procedure

```
1: Define restart size $s$ and target $\tilde{\sigma}$
2: Compute SVD of $R = X\Sigma_r Y^T$
3: Choose $s$ singular triplets of $R$ closest to $\tilde{\sigma}$ (called $(X_1, \Sigma_r^{(1)}, Y_1)$)
4: Save the remaining singular triplets from the SVD of R, $(X_2, \Sigma_r^{(2)}, Y_2)$
5: $v_{new} \leftarrow$ Orthogonalize saved +k vectors $[v_{old}; 0]$ from main iteration against $Y_1$
6: $t = [Y_1, v_{new}]$
7: $V = Vt$
8: if Reset criteria is met then
9:     Reorthogonalize $V$ and build $Q$ and $R$ such that $AV = QR$
10: else
11:     QR factorize $\Sigma_r^{(2)} Y_2^T v_{old} = \tilde{Q}\tilde{R}$
12:     Set $Q = Q[X_1 X_2 \tilde{Q}]$ and $R = \begin{bmatrix} \Sigma_r^{(1)} & 0 \\ 0 & \tilde{R} \end{bmatrix}$.
13: end if
```

From (9), the new Q and R can be obtained with minimal effort by performing a QR factorization $\Sigma_r^{(2)} Y_2^T v_{old} = \tilde{Q}\tilde{R}$. The restarted Q and R are given in Line 12 of Algorithm 2.

To accurately find many singular triplets, we implement two versions of locking. The first, hard-locking, locks singular vectors out of the search space explicitly once the required user tolerance is reached. At every iteration, we orthogonalize the vector added to V against the locked right singular vectors, as well as the previous vectors in V. In practice, the vectors added to Q do not require orthogonalization against the locked left singular vectors. The second, soft-locking, merely flags converged singular triplets while leaving them in the basis.

In some rare cases, we can see stagnation due to hard locking. This is caused by the error still present in the locked vectors, which may contain critical directions for other singular triplets [15]. We have not seen any matrices in this paper that exhibit this behavior. However, soft-locking can provide left and right singular vectors that are orthogonal to machine precision, while hard-locking only obtains left singular vectors orthogonal up to $O(\|A\|\delta)$. Therefore, we present soft-locking results in the following section. We intend to address the issues with hard-locking more thoroughly in the future.

2.3. Resetting. Due to $AV = QR$, the right residual $r_v = Av - \sigma u$ should be zero throughout our procedure,

$$r_v = Av - \sigma u = AVy - Q(\sigma x) = AVy - QRy = (AV - QR)y = 0. \tag{10}$$

Generally, this means we can avoid the extra matrix-vector multiplication (or storage for AV) necessary to compute r_v. In practice though, $\|r_v\|$ cannot be better than $O(\|A\|\epsilon_{mach})$ due to the multiplication with A required to compute it. Worse, $\|r_v\|$ grows as $O(\sqrt{\text{numRestarts}}\|A\|\epsilon_{mach})$, which has also been noticed in [19]. Therefore, our method must calculate $\|r_v\|$ explicitly when $\|r_u\| < \|A\|\delta$, where δ is the user selected tolerance. This ensures we meet the convergence criteria of Algorithm 1.

The errors we observe in r_v may grow large enough to exceed the user tolerance, which would make convergence impossible. These errors come from two main sources. The first source is from the loss of orthogonality of V, and the second is the loss of accuracy of the QR factorization. We have found experimentally that both of these errors can impede or halt convergence as the SVD of R no longer corresponds to the singular triplets in A. We note that this issue is rare and only occurs when

$\delta \approx \epsilon_{mach}\sqrt{\text{numRestarts}}$. To correct these errors, we implement a resetting procedure that reorthogonalizes V, and rebuilds Q and R directly from AV.

It is critical to only reset sparingly, as rebuilding Q and R from scratch takes $s+k$ matvecs to obtain AV and a full QR factorization. Additionally, resetting can cause an increase in the residual norm by a factor of $\kappa(A)$, which may require a few iterations to reduce back to its previous level. In order to track the errors mentioned above, we have devised two inexpensive criteria that help to avoid unnecessary resets. From (10), we can estimate errors in the QR factorization directly from the norm of the right residual. We choose to reset when $\|r_u\| < 1.25\|r_v\|$, as the errors in the QR factorization directly impact the convergence of r_u. Experimentally, we have found a few cases where the small 25% buffer between r_u and r_v is needed to detect potential stagnation.

The error in the orthogonality of V may also cause failures to converge. Therefore, we estimate how large $\|E\| = \|V^TV - I\|$ can be before it begins to affect convergence. Based on the Galerkin conditions, we should have solved the equivalent eigenproblem, $R^TRy = V^TA^TAVy = \sigma^2V^TVy$. In practice, we solve $R^TRy = V^TA^TAVy = \sigma^2 y$ regardless of the orthonormality of V. Therefore, we obtain a Ritz vector and Ritz value that will not converge to a 0 residual for the original problem, since $V^TV \neq I$. However, the Ritz pair produced by our inexact Galerkin can be considered as a Ritz pair of an exact Galerkin condition applied to the nearby generalized eigenproblem $A^TAVy = \sigma^2MVy$ where $M = V(V^TV)^{-2}V^T$ as seen below,

$$V^TA^TAVy = \sigma^2V^TMVy = \sigma^2V^TV(V^TV)^{-2}V^TVy = \sigma^2y. \tag{11}$$

In order to correctly monitor and maintain convergence, the residual we use for expansion, $r_C = \sigma r_u = A^TAv - \sigma^2v$, should not drift too far from this exact residual, $r_E = A^TAv - \sigma^2V(V^TV)^{-2}V^Tv$, where $v = Vy$. Assuming $\|E\| < 1$, we have

$$\begin{aligned}\|r_E - r_C\| &= \sigma^2\|Vy - V(V^TV)^{-1}y\| \\ &\le \sigma^2\|V\|\|I - (V^TV)^{-1}\| = \sigma^2\|V\|\|I - (I+E)^{-1}\| \\ &\le \sigma^2(1+\|E\|)\|(I+E)^{-1}\|\|E\| \\ &\le \sigma^2(1+\|E\|)\left\|I + \sum_{i=1}^{\infty}E^i\right\|\|E\| \\ &= \sigma^2\|E\| + O(\sigma^2\|E\|^2).\end{aligned} \tag{12}$$

Since we want $r_u = r_C/\sigma$ to converge to tolerance $\|A\|\delta$, we limit the distance $\|r_E - r_C\| < \|A\|\delta\sigma$. Thus, from (12), we perform a reset when $\|E\| \ge \|A\|\delta/\sigma$. In practice we have noticed very few situations where this criteria caused a reset.

3. Numerical Results.

To verify our algorithm's performance, we utilized the same matrices given in the original PHSVDS publication [20]. These matrices are publicly available through the University of Florida Sparse Matrix Collection [4] and represent real world applications. These problems are quite difficult for iterative solvers and are used to stress test the capabilities of GKD and PHSVDS. Since these matrices are sparse, we provide their dimensions and the number of non-zero entries of A, $nnz(A)$, as well as the norm of A, $\|A\|$, the condition number of A, $\kappa(A)$, and the gap ratio for σ_1, $\gamma_1 = (\sigma_2 - \sigma_1)/(\sigma_n - \sigma_2)$.

The matrices listed in Table 1 and Table 2 are listed from least to most difficult (left to right) as generally their condition numbers increase, and the gap ratios for

Matrix	pde2961	dw2048	fidap4	jagmesh8	wang3	lshp3025
dimension	2961	2048	1601	1141	26064	3025
nnz(A)	14585	10114	31837	7465	77168	120833
$\kappa(A)$	9.5E+2	5.3E+3	5.2E+3	5.9E+4	1.1E+4	2.2E+5
$\|A\|$	1.0E+1	1.0E+0	1.6E+0	6.8E+0	2.7E-1	7.0E+0
γ_1	8.2E-3	2.6E-3	1.5E-3	1.7E-3	7.4E-5	1.8E-3

TABLE 1
Basic Properties of Square Matrices

Matrix	well1850	lp_ganges	deter4	plddb	ch	lp_bnl2
rows	1850	1309	3235	3049	3700	2324
columns	712	1706	9133	5069	8291	4486
nnz(A)	8755	6937	19231	10839	24102	14996
$\kappa(A)$	1.1E+2	2.1E+4	3.7E+2	1.2E+4	2.8E+3	7.8E+3
$\|A\|$	1.8E+0	4.0E+0	1.0E+1	1.4E+2	7.6E+2	2.1E+2
γ_1	3.0E-3	1.1E-1	1.1E-1	4.2E-3	1.6E-3	7.1E-3

TABLE 2
Basic Properties of Rectangular Matrices

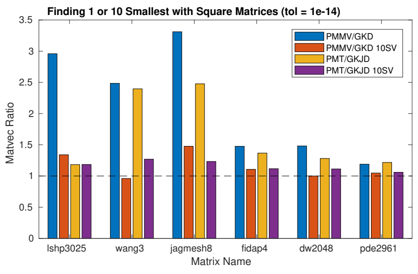
their smallest singular values decrease. It should be noted that none of these matrices are particularly poorly conditioned, and do not require the second stage in PHSVDS to improve the singular vector estimates more than a few orders of magnitude. Therefore, the benefits we would expect to gain on very poorly conditioned problems are significantly larger.

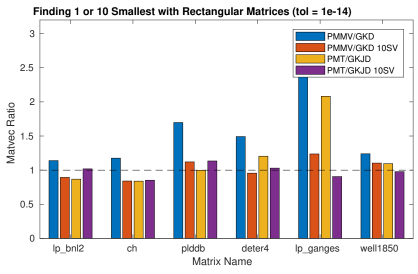
We restrict GKD and PRIMME_SVDS to a maximum basis size of 35 vectors, a minimum restart size of 15 vectors and a user tolerance of δ = 1E-14. We also enforce two retained vectors from the previous iteration (for +2 restarting) and soft-locking. Due to the interior nature of the augmented method in PRIMME_SVDS, we are unable to set soft-locking for the second stage while searching for the smallest singular triplets. It should be noted that hard-locking generally improves performance for our method when searching for more than one singular value, but does not provide the same orthogonality guarantees and is subject to the numerical issues mentioned earlier.

We compare PRIMME_SVDS MIN_MATVECS (GD+k) against our GKD, and PRIMME_SVDS MIN_TIME (JDQMR) against GKJD. As shown in Figure 1, GKD and GKJD require fewer matrix-vector multiplications than their PRIMME_SVDS counterparts for nearly all matrices. Also, the matrices that show the largest benefits are lshp3025, wang3, jagmesh8, and lp_ganges. As expected, these correspond to the matrices that required more significant use of the second stage in PRIMME_SVDS, due to their larger $\kappa(A)$. For most cases, we see a drop off in performance when searching for the 10 smallest singular values, but this is mostly caused by soft-locking. Using soft-locking in the first stage of PRIMME_SVDS can improve the initial guesses to the second stage in some cases, negating the advantage GKD has over the two-stage method.

For rectangular matrices, we also tested whether our method could find a true zero singular value by adding an extra column equal to the first column. GKD is able to find the real zero in all cases. PRIMME_SVDS will not return this numerically zero value, as outlined in its documentation, since its second stage has no way to distinguish real zeros from the null space created by the augmented matrix.

For preconditioning, we provide a preconditioner built using Matlab's ILU with the ilutp factorization, a drop-tolerance of 1E-3, and a pivot threshold of 1.0. Our

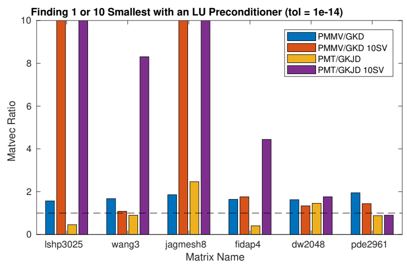




	lshp3025	wang3	jagmesh8	fidap4	dw2048	pde2961
GKD	27635	17651	13137	12066	4474	5661
GKD 10SV	112566	106630	46346	54483	21367	25815
GKJD	35660	17620	16686	12726	5358	6504
GKJD 10SV	133333	100103	55225	55019	27413	29333

	lp_bnl2	ch	plddb	deter4	lp_ganges	well1850
GKD	36087	24665	5670	752	604	1212
GKD 10SV	163167	184704	19515	26404	5519	4683
GKJD	29988	28358	7732	1038	728	1528
GKJD 10SV	95233	142563	24709	20487	7183	6515

FIG. 1. *Unpreconditioned Comparison against Primme MIN_MATVECS (PMMV) and Primme MIN_TIME (PMT) for Square and Rectangular Matrices. The tables provide the matvecs needed by GKD and GKJD.*



	lshp3025	wang3	jagmesh8	fidap4	dw2048	pde2961
GKD	60	136	42	50	48	38
GKD 10SV	469	839	301	413	413	315
GKJD	1038	298	120	456	110	134
GKJD 10SV	2409	1515	943	1537	1195	901

FIG. 2. *Preconditioned Comparison against Primme MIN_MATVECS (PMMV) and Primme MIN_TIME (PMT) for Square Matrices. The table provides the matvecs needed by GKD and GKJD.*

results show the significant benefit of an effective preconditioner, as all problems required less than 150 matvecs when searching for one singular value with GKD. However, these preconditioners sometimes caused significant issues for PRIMME_SVDS, as it was unable to converge for lshp3025 when searching for the 10 smallest singular values, and exhibited significant difficulty converging to 10 singular values for wang3, jagmesh8 and fidap4. These issues are caused by PRIMME_SVDS' first stage trying to achieve full accuracy on C. The two cases where PRIMME_SVDS outperforms our method (lshp3025 and fidap4 searching for 1 SV) are the result of a few extra iterations within the inner method of GKJD. This is due to further optimizations built into the QMRs dynamic stopping criteria of PRIMME_SVDS.

4. Conclusions.

We have presented GKD, a new method for finding the smallest singular triplets of large sparse matrices to full accuracy. Our method works iteratively, under limited memory, with preconditioners, while including features such as soft-locking with orthogonality guarantees, +k restarting, and the ability to find real zero singular values in both square and rectangular matrices. Additionally, GKJD uses an inner solver for the A^TA correction equation into GKD, which can lower execution time when the matrix-vector multiplication operation is inexpensive. Both of these methods have shown to be more reliable and efficient than PHSVDS, and thus over other SVD methods, for nearly all cases.

REFERENCES

[1] O. Alter, P. O. Brown, and D. Botstein, *Singular value decomposition for genome-wide expression data processing and modeling*, Proceedings of the National Academy of Sciences, 97 (2000), pp. 10101–10106.

[2] J. Baglama and L. Reichel, *Augmented implicitly restarted Lanczos bidiagonalization methods*, SIAM J. Sci. Comput., 27 (2005), pp. 19–42.

[3] J. Baglama and L. Reichel, *Restarted block Lanczos bidiagonalization methods*, Numerical Algorithms, 43 (2006), pp. 251–272.

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

[5] J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[6] G. H. Golub and C. F. Van Loan, *Matrix Computations (3rd Ed.)*, Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[7] M. E. Hochstenbach, *A Jacobi–Davidson type SVD method*, SIAM J. Sci. Comput., 23 (2001), pp. 606–628.

[8] Z. Jia and D. Niu, *An implicitly restarted refined bidiagonalization Lanczos method for computing a partial singular value decomposition*, SIAM J. Matrix Anal. Appl., 25 (2003), pp. 246–265.

[9] I. Jolliffe, *Principal component analysis*, Wiley Online Library, 2002.

[10] A. V. Knyazev, *Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method*, SIAM journal on scientific computing, 23 (2001), pp. 517–541.

[11] R. M. Larsen, *Lanczos bidiagonalization with partial reorthogonalization*, DAIMI Report Series, 27 (1998).

[12] S. Osiński, J. Stefanowski, and D. Weiss, *Lingo: Search results clustering algorithm based on singular value decomposition*, in Intelligent information processing and web mining, Springer, 2004, pp. 359–368.

[13] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, 1980.

[14] H. Prasantha, H. Shashidhara, and K. B. Murthy, *Image compression using svd*, in Conference on Computational Intelligence and Multimedia Applications, 2007. International Conference on, vol. 3, IEEE, 2007, pp. 143–145.

[15] A. Stathopoulos, *Locking issues for finding a large number of eigenvectors of hermitian matrices*, tech. report, Tech Report WM-CS-2005-09, Computer Science, The College of William & Mary, 2005.

[16] A. Stathopoulos, *Nearly optimal preconditioned methods for Hermitian eigenproblems under limited memory. part i: Seeking one eigenvalue*, SIAM J. Sci. Comput., 29 (2007), pp. 481–514.

[17] E. Vecharynski, *Preconditioned Iterative Methods for Linear Systems, Eigenvalue and Singular Value Problems*, PhD thesis, University of Colorado at Denver, Denver, CO, USA, 2011. AAI3456056.

[18] K. Wu and H. Simon, *Thick-restart Lanczos method for large symmetric eigenvalue problems*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 602–616.

[19] L. Wu, E. Romero, and A. Stathopoulos, *Primme_svds: A high-performance preconditioned svd solver for accurate large-scale computations*, arXiv preprint arXiv:1607.01404, (2016).

[20] L. Wu and A. Stathopoulos, *A preconditioned hybrid svd method for accurately computing singular triplets of large matrices*, SIAM Journal on Scientific Computing, 37 (2015), pp. S365–S388.