Enhancing the PRIMME Eigensolver for Computing Accurately Singular Triplets of Large Matrices

Lingfei Wu[1], Andreas Stathopoulos[1]

Abstract

The computation of a few singular triplets of large, sparse matrices is a challenging task, especially when the smallest magnitude singular values are needed in high accuracy. Most recent efforts try to address this problem through variations of the Lanczos bidiagonalization method, but algorithmic research is ongoing and without production level software. We show that a more efficient, robust, and production-ready method can be developed on top of the state-of-the-art eigensolver PRIMME. In a first stage, our new method, primme_svds, solves the normal equations problem up to the best achievable accuracy. If further accuracy is required, the method switches automatically to an eigenvalue problem with the augmented matrix. Thus it combines the advantages of the two stages, faster convergence and accuracy, respectively. For the augmented matrix, solving the interior eigenvalue is facilitated by a proper use of the good initial guesses from the first stage and an efficient implementation of the refined projection method. The method can be used with or without preconditioning, on large problems, and can be called with its full functionality from MATLAB through our MEX interface. Numerical experiments illustrate the efficiency and effectiveness of the presented method.

1 Introduction

The Singular Value Decomposition (SVD) is a ubiquitous computational kernel in science and engineering. Many applications demand a few of the largest or smallest singular values of a large sparse matrix A and the associated left and right singular vectors. It is well known that the computation of the smallest singular triplets presents challenges both to the speed of convergence and the accuracy that iterative methods can accomplish. In this paper we mainly focus on this problem of finding the smallest singular triplets.

Assume $A \in \Re^{m\times n}$ is a large sparse matrix with full column rank and $m \geq n$. The singular value decomposition of A can be written as: $A = U\Sigma V^T$, where $U = [u_1,\ldots,u_m] \in \Re^{m\times m}$ and $V = [v_1,\ldots,v_m] \in \Re^{n\times n}$ are unitary matrices, and $\Sigma = diag(\sigma_1,\ldots,\sigma_n) \in \Re^{m\times n}$ contains the singular values of A in increasing order $\sigma_1 \leq \ldots \leq \sigma_n$. We will be looking for the smallest $k \ll n$ singular triplets $\{\sigma_i, u_i, v_i\}, i = 1,\ldots,k$.

There are two approaches to compute the singular triplets $\{\sigma_i, u_i, v_i\}$ by using a Hermitian eigensolver. The first approach seeks a few eigenpairs of the augmented matrix $B = [0\ A^T; A\ 0] \in \Re^{(m+n)\times(m+n)}$, which has eigenvalues $\pm\sigma_1,\ldots,\pm\sigma_n$ and eigenvectors whose upper and lower part are the right and left singular vectors respectively [2, 3, 4]. The main advantage of this approach is that iterative methods can compute the smallest singular values accurately, i.e., with residual norm close to $O(\|A\|e_{mach})$. However, they must solve a highly interior eigenvalue problem, and even the use of iterative refinement or inverse iteration involves a maximally indefinite matrix. This results in very slow convergence [1]. When restarting is used, convergence is even slower, irregular, and often the required eigenvalues are missed since the Rayleigh-Ritz projection method does not effectively extract the appropriate information for interior eigenvectors.

The second approach computes eigenpairs of the normal equations matrix $C = A^TA \in \Re^{n\times n}$ which has eigenvalues $\sigma_1^2, \leq \ldots, \leq \sigma_n^2$ and the associated eigenvectors are the right singular vectors of A. If $\sigma_i \neq 0$, the corresponding left singular vectors are obtained as $u_i = \frac{1}{\sigma_i}A^Tv_i$. This approach is preferred for largest singular values because its gap ratios are much larger than the augmented approach. For smallest singular values the gaps in C reduce but convergence is still faster because of the indefiniteness of B. However, squaring the matrix limits the accuracy we can obtain for the smallest singular triplets. Therefore, when eigenvalue methods are applied on C, they are typically followed by a second stage of iterative refinement

[1]Department of Computer Science, College of William and Mary, Williamsburg, Virginia 23187, (lfwu,andreas@cs.wm.edu)

for each needed singular triplet which resolves the numerical problems of the first stage [8, 9]. However, this one-by-one iterative refinement does not exploit information from other singular vectors and thus is it not as efficient as an eigensolver applied on B with the estimates of the first stage.

Over the last decade, the Lanczos bidiagonalization method [2, 10] has gained acceptance as an accurate and more efficient method for seeking singular triplets (especially smallest), and numerous variants have been proposed [17, 14, 16, 11, 12, 13, 15, 18]. The Lanczos bidiagonalzation method computes the same information as Lanczos on matrix C, but since it works on A directly, it avoids the numerical problems of squaring. Despite the remarkable algorithmic progress, however, current Lanczos bidiagonalization methods are still in development, with only few existing MATLAB implementations that serve mainly as a testbed for mathematical research. Moreover, as we show in this paper, a two stage approach based on a well designed eigenvalue code (such as PRIMME) can be more robust and efficient for a few singular triplets. Most importantly, our approach can use preconditioners to accelerate convergence, something that is not directly possible with Lanczos bidiagonalization but becomes crucial because of the difficulty of the problem even for medium matrix sizes.

The Jacobi-Davidson type SVD method, JDSVD [5, 6], which can also use preconditioning, obtains the left and right singular vectors directly from a projection of B on two subspaces. To cope with the irregular convergence of the Rayleigh-Ritz method, JDSVD must be coupled with a harmonic projection method [19, 22, 21] or a refined projection method [23, 24]. JDSVD is only available in a MATLAB research implementation which is not robust enough for real world problems. But most importantly, because it works on B, JDSVD may be less efficient than a two stage method that first works on C for efficiency and then on B for final accuracy.

In this paper we develop a hybrid, two-stage singular value method that address the algorithmic problems of the above methods but also provides a production-ready implementation based on the state-of-the-art eigensolver PRIMME [27]. In the first stage, the proposed method primme_svds solves an extreme eigenvalue problem on C up to the user required accuracy or up to the accuracy achievable by the normal equations. If further accuracy is required, primme_svds switches to a second stage where it utilizes the eigenvectors and eigenvalues from C as initial guesses to a Jacobi-Davidson method on B, which has been enhanced by a refined projection method. The appropriate choices for tolerances, transitions, selection of target shifts, and initial guesses are handled automatically by the method. Our extensive numerical experiments show that primme_svds can be considerably more efficient than all other methods when computing a few of the smallest singular triplets, even without a preconditioner.

In the section 2 we describe how to adjust the tolerance in the first stage and how to address irregular convergence of the standard projection method in the second stage. Other algorithmic enhancements and an outline of the implementation are also discussed. In Section 3, we present several experiments that corroborate our conclusions.

2 Developing the two stage strategy

To solve the eigenvalue problem on C and then on B, we use the suite of eigenvalue methods provided in PRIMME [25, 26, 27]. PRIMME implements a wide variety of preconditioned eigenvalue algorithms, including the nearly optimal methods GD+k and JDQMR, and a host of practical techniques for improving efficiency and robustness, including block-methods, locking and a variety of restarting techniques and heuristics. Also given a set of user provided shifts, PRIMME can find interior eigenvalues closest to in absolute value or on the left or right side of each of these shifts. This functionality is central in our stage two where very good eigenvalue approximations are available. PRIMME has proved faster and more robust than almost any other eigensolver when seeking a small number of extreme eigenvalues of large sparse hermitian matrices. PRIMME also chooses the optimal method for a problem dynamically, based on runtime

measurements. Our meta-method lets PRIMME make most of these choices, but needs to address certain issues. We discuss what accuracy can be expected from the normal equations in the first stage, and how to use the approximate solutions as an initial guess to the second stage. In the second stage we show how to overcome convergence problems with interior eigenvalues by using a refined projection method.

2.1 The first stage of primme_svds

A straightforward theoretical argument on gap ratios explains the experimental observations ([9]) that Krylov methods on the normal equations are faster than on the augmented matrix for finding a few largest singular triplets. In our experiments, the benefits were even larger due to the nearly optimal convergence of the two eigenmethods GD+k and JDQMR in PRIMME. As we show below, the first stage of primme_svds is sufficient to compute a few of largest singular triplets without compromising the accuracy.

When seeking the smallest eigenvalues of C, convergence of Krylov methods is slow but still faster than seeking interior eigenvalues of B. In fact, the more clustered the singular values, the bigger the convergence benefits of working on C. Accuracy, however, is limited by the squared matrix norm.

Let (σ, u, v) be a targeted singular triplet of A and $(\tilde{\sigma}^2, \tilde{v})$ the approximating Ritz pair from a method working on C. Using the approximation $\tilde{u} = A\tilde{v}/\tilde{\sigma}$ we can write the following four residuals:

$$r_v = A\tilde{v} - \tilde{\sigma}\tilde{u}, \quad r_u = A^T\tilde{u} - \tilde{\sigma}\tilde{v}, \quad r_C = C\tilde{v} - \tilde{\sigma}^2\tilde{v}, \quad r_B = B\begin{bmatrix} \tilde{v} \\ \tilde{u} \end{bmatrix} - \tilde{\sigma}\begin{bmatrix} \tilde{v} \\ \tilde{u} \end{bmatrix}. \tag{1}$$

Typically a singular triplet is considered converged when $\|r_v\|$ and $\|r_u\|$ are less than a given tolerance. Since our eigenvalue methods work on C and B we need to relate the above quantities. First, it is easy to see that $r_C = A^T(A\tilde{v}) - \tilde{\sigma}^2\tilde{v} = \tilde{\sigma}A^T\tilde{u} - \tilde{\sigma}^2\tilde{v} = \tilde{\sigma}r_u$. To relate to the norm of the residual of the second stage note that $\|r_B\|^2 = (\|r_v\|^2 + \|r_u\|^2)/(\|\tilde{v}\|^2 + \|\tilde{u}\|^2)$. By choosing a Ritz vector in the normal equations with $\|\tilde{v}\| = 1$, we also obtain $\|\tilde{u}\| = \|A\tilde{v}/\tilde{\sigma}\| = 1$ and $r_v = 0$. Bringing it all together:

$$\|r_u\| = \frac{\|r_C\|}{\tilde{\sigma}} = \|r_B\|\sqrt{2}. \tag{2}$$

Therefore, given a user requirement $r_u < \delta\,\|A\|$, the normal equations method and the augmented method are stopped when $r_C < \delta\,\tilde{\sigma}\,\|A\|$ and $r_B < \delta\,\|A\|/\sqrt{2}$ respectively. Since the stopping criterion for PRIMME is $\|r_C\| < \delta_C\|C\|$, we must provide $\delta_C = \delta\,\tilde{\sigma}/\|A\|$. In floating point arithmetic this may not be achievable since $\|r_C\|$ can only reach down to $\|A\|^2\varepsilon_{mach}$ [1]. Thus, the criterion for the normal equations becomes,

$$\delta_C = \max\left(\delta\,\tilde{\sigma}/\|A\|, \varepsilon_{mach}\right) \tag{3}$$

while any remaining accuracy will have to be obtained by running the second stage.

First, note that for the largest $\sigma = \|A\|$, full accuracy is achievable with the normal equations, as $\delta_C = \delta$. Then, based on the Bauer-Fike bound and since $\sigma \approx \tilde{\sigma}$, the singular values are as accurate as can be expected: $|\sigma^2 - \tilde{\sigma}^2| = |\sigma - \tilde{\sigma}|(2\tilde{\sigma}) \leq \|r_C\| < \delta_C\|A\|^2 = \tilde{\sigma}\delta\|A\|$, and thus $|\sigma - \tilde{\sigma}| \leq \delta\|A\|/2$. This does not hold for smaller, and in particular the smallest few, eigenvalues. Thus, if the user requires $\delta < \|A\|\varepsilon_{mach}/\tilde{\sigma}$, primme_svds first makes full use of the first stage and then switches to the second stage to continue computing the smallest singular triplets accurately on B.

Any preconditioner for C can be used directly in PRIMME in this stage. Most often a preconditioner for $M \approx A$ and $M^T \approx A^T$ would instead be available (e.g., incomplete LU factorization). The M^TM operator could then be provided to PRIMME to speed up the calculation. Regarding the choice of eigenvalue methods, we let PRIMME's dynamic scheme decide whether a GD+k or JDQMR would be preferable. This depends on runtime issues including the cost of the matrix vector and preconditioning operators.

2.2 The second stage of primme_svds

Typically, singular triplets obtained through normal equations are subsequently improved using an iterative refinement procedure [8, 9]. We argue that solving an eigenvalue problem on matrix B with the approximate eigenspace as initial guess is a better approach for the following three reasons. First, with iterative refinement, eigenvectors are improved one by one without any cross utilization of the nearby subspace information. Instead, an eigensolver improves not only the targeted but also the nearby eigenvectors, accelerating global convergence. Second, solving the linear system in iterative refinement is not as efficient as in JDQMR, which stops the linear solver dynamically to avoid exiting early (which increases the number of outer iterations) or iterating too long (which increases the number of inner iterations). Third, iterative refinement for clustered interior eigenvalues may not be able to converge to the desired accuracy due to the lack of proper deflation strategies, both at the linear solver and at the outer iteration. Naturally, a well designed eigensolver that employs locking and blocking techniques is more robust to address these problems. Finally, we point out that the correction equation of the Jacobi-Davidson method applied on B^T:

$$(I - ww^T)(B^T - \mu I)(I - ww^T)\tilde{t} = \tilde{\sigma}w - B^T w \tag{4}$$

where $w^T = \begin{bmatrix} u & v \end{bmatrix}$ and $\tilde{t}^T = \begin{bmatrix} s & t \end{bmatrix}$ is equivalent to the iterative refinement proposed in [8] (see Lemma 7.1 in [5]). Therefore, JDQMR enjoys the benefits of both eigensolvers and iterative refinement.

PRIMME provides remarkable flexibility for seeking interior eigenvalues. Unlike the Lanczos method, it accepts initial guesses from the first stage for all required eigenvectors. Then it builds the initial search space by running a few Lanczos steps on the initial vector associated with the first targeted eigenvalue. For this, we modified slightly the original PRIMME implementation which included a small Lanczos space from a random vector to guard against extremely bad guesses, which is not the case here. Thus the initial search space is rich in eigenvector components for all nearby eigenvectors. Once an eigenpair converges it is locked out of the search space, and the initial guess associated with the next targeted shift is introduced into the search space. In a slight departure from the original PRIMME implementation, we reintroduce initial guesses even if these were part of the initial search space. The reason is that convergence for interior eigenvalues is plagued by spurious eigenvalues which may even displace nearly converged eigenvectors from the search space. This resulted in significant improvement in robustness and often in convergence.

PRIMME accepts multiple shifts and provides three different ways to select an interior eigenvalue close to each shift. The most common is to select Ritz values which are closest in absolute value to each shift (`primme_closest_abs`). In certain cases, it is beneficial to look only at those on the left or on the right of the shift. Since the accuracy of eigenvalues of C is $O(\|r_C\|^2)$, the shifts we provide to the second stage are very close to the eigenvalues of B. Because of this, the `primme_closest_abs` option is more effective at selecting the proper Ritz value during the outer iterations. More importantly, these shifts are ideal for the Jacobi-Davidson correction equation, which typically returns the exact correction to the eigenvector after the solution of only one linear system.

For interior eigenvalues, the Rayleigh-Ritz procedure does not have the same optimality as for extreme eigenvalues. This can cause convergence to be irregular which makes it difficult to select appropriate Ritz pairs or can produce spurious ones. We have observed that in a well implemented GD+k method with sufficiently large search space, such selection problems are transient and do not affect the convergence and overall speed of the method. This is why PRIMME only implemented the Rayleigh-Ritz method originally. In the second stage of primme_svds, the availability of accurate initial guesses and shifts calls for using the correction equation, i.e., the JDQMR method. For this method, spurious Ritz values can cause Ritz vectors to fail to converge [24]. The effect is detrimental not only during the correction equation, but also during restart where major eigenvector components would be discarded and need to be recovered. [19, 20, 21]. The problem is accentuated in the maximally indefinite case of SVD problems.

We addressed this problem by extending PRIMME's functionality to include the refined projection that minimizes the residual $\|BVy - \tilde{\sigma}Vy\|$ over the search space V and for a given $\tilde{\sigma}$ [23, 24]. Because the shifts $\tilde{\sigma}$ are very accurate, a harmonic Ritz procedure is not necessary, and the refined one is expected to give the best approximation for the targeted eigenpair. Our refined projection method is similar to the one in [6] and [19] which produces refined Ritz vectors for all the required eigenvectors (not just the closest to $\tilde{\sigma}$). Since $\tilde{\sigma}$ remains constant, there is no need to perform a QR factorization of the $BV - \tilde{\sigma}V$ matrix at every step. Instead, as part of a Gram-Schmidt, we add one more column to the orthonormal matrix Q and to the matrix R. A full QR factorization is only needed at restart. Then, following [24], we compute the refined Ritz vectors by solving the small SVD problem with R, and replace the targeted Ritz value with the Rayleigh quotient based on the first refined Ritz vector.

Solving the small SVD problem for only one shift per iteration reduces the cost of the refined procedure considerably, making it similar to the cost of computing the Ritz vectors. Although the quality of other refined Ritz vectors reduces with the distance of their Rayleigh quotient from $\tilde{\sigma}$, these approximations have the desirable property of monotonic convergence as claimed in [6, 19] and also observed in our experiments. This added robustness for JDQMR more than justifies the small additional cost.

2.3 Outline of the implementation

We first developed PRIMME MEX, a MATLAB interface for PRIMME. This exposes the full functionality of PRIMME to a broader class of users, who can now take advantage of MATLAB's built-in blocked matrix-vector operators and preconditioners. Its user interface is similar to eigs() allowing it to be called not only by non-expert users but also by experts that can adjust more than 30 parameters. The meta-method primme_svds was then implemented as a MATLAB function on top of PRIMME MEX. This allowed flexibility for algorithmically tuning the two stages. Enhancements such as the refined projection method were implemented directly in PRIMME and will be part of its next release, which will also include a native C implementation of primme_svds.

3 Numerical experiments

We conduct three sets of numerical experiments to compare primme_svds against three other state of the art SVD methods: IRRHLB [15], JDSVD [5, 6], and MATLAB's svds (based on the ARPACK software [30]). In the first set of experiments, we compare with IRRHLB and JDSVD, and show primme_svds to be much more efficient in determining a few of the smallest singular triplets, even without a preconditioner. In the second set of experiments, we demonstrate that primme_svds provides a much faster convergence with a good preconditioner, which becomes necessary for larger problems. In the last set of experiments, primme_svds is compared with svds. Both primme_svds and svds leverage the shift-and-invert technique but our method achieves better performance. All computations are carried out on a DELL server with sixteen 2.93GHz Intel Xeon processors and 50 GB of memory running Linux operating system and using MATLAB 2013a with machine precision $\varepsilon = 2.2 \times 10^{-16}$.

All methods start with the same initial guess, ones(min(m,n),1), except for matrix lshp3025 for which a random guess is necessary. We set maxBasisSize=35, minRestartSize=21 and experiment with two δ tolerances, 1e-8 and 1e-14. For δ =1e-8, primme_svds does not need to enter the second stage. Since the numbers of matrix-vector products with A and A^T are the same, the tables report as "MV" the number of products with A only. "Sec" is the run time in seconds, and "*" means the method cannot converge to all desired singular values. For the first stage of primme_svds we have used the GD_Olsen_PlusK method. For the second stage, we run experiments with both GD_PlusK and JDQMR.

We select seven test matrices from [15, 29]. Table 1 lists these matrices along with some basic properties.

Table 1: Properties of the test matrices. $gap_{min}(k) = \min_{i=1}^{k}(gap(\sigma_i))$, where $gap(\sigma_i) = min_{j\neq i}|\sigma_i - \sigma_j|$.

Matrix	well1850	pde2961	dw2048	fidap4	jagmesh8	lshp3025	wang3
order	1850	2961	2048	1601	1141	3025	26064
nnz(A)	8755	14585	10114	31837	7465	20833	177168
$\kappa(A)$	1.1e2	9.5e2	5.3e3	5.2e3	5.9e4	2.2e5	1.1e4
$\|A\|_2$	1.8e0	1.0e1	1.0e0	1.6e0	6.8e0	7.0e0	2.7e-1
$gap_{min}(1)$	3.0e-3	8.2e-3	2.6e-3	1.5e-3	1.7e-3	1.8e-3	7.4e-5
$gap_{min}(3)$	3.0e-3	2.4e-3	2.9e-4	2.5e-4	1.6e-3	9.1e-4	1.9e-5
$gap_{min}(5)$	3.0e-3	2.4e-3	2.9e-4	2.5e-4	4.8e-5	1.8e-4	1.9e-5
$gap_{min}(10)$	2.6e-3	7.0e-4	1.6e-4	2.5e-4	4.8e-5	2.2e-5	6.6e-6

Among them, the matrices well1850, pde2961 and dw2048 are easy ones, the matrices fidap4, jagmesh8 and wang3 are hard cases, and the matrix lshp3025 is a very hard one.

3.1 Without preconditioning

We compute the k smallest singular triplets for $k = 1, 3, 5, 10$ using the default parameters in all methods. In order to speed up convergence of the eigenvalue problems, $k+3$ eigenvalues are computed when k desired eigenvalues are required in ARPACK [30]. A similar strategy is applied in [15, 11]. For primme_svds, we found this is not necessary except when $k = 1$, when it is better to find 2 eigenpairs in the first stage, and pass their space as initial guesses to stage two. Stage two only calculates the k needed eigenpairs.

Tables 2 and 3 show that the primme_svds variants converge faster and more robustly than other methods. Specifically, Table 2 shows that for moderate accuracy the normal equations solved with a PRIMME method are unbeatable. This is expected from the near-optimal properties of our methods [25]. However, when looking for ten or more smallest singular triplets, the Lanczos type methods begin to show their effectiveness [26].

Table 3 shows smaller differences between the methods, reflecting the slower convergence of the augmented method in stage two. For computing 10 eigenpairs, IRRHLB shows a small edge in convergence for three cases. However, primme_svds method never misses eigenvalues, and is still much faster than JDSVD and IRRHLB in most cases, and significantly better than IRRHLB in the hard cases. JDSVD is significantly slower than primme_svds because it relies on the augmented matrix to produce all the required accuracy. Taking advantage of the normal equations, primme_svds can then use more effectively the refined Ritz vectors to improve the interior eigenvalue problem.

We point out that because of PRIMME's high quality implementation, not only does primme_svds enjoy better robustness but it is also ten times faster than IRRHLB for the cases where IRRHLB takes fewer MVs!

3.2 With preconditioning

The most notable fact from Tables 2 and 3, however, may be the difficulty of solving this SVD problem, even for small matrices. Preconditioning is a prerequisite for addressing the larger problems in practice, which limits our choice to primme_svds and JDSVD. We compare the two methods, using as a preconditioner M^TM or $[0\ M^T; M\ 0]$, where $M = LU$, the factors obtained from MATLAB's ILU function on A with parameters `'type=ilutp','droptol'=1e-3,'thresh'=1.0`. We focus only on the hard cases of the previous experiments. Except for preconditioning, all other parameters remain the same. Table 4 shows that a good preconditioner makes the problems tractable, with both primme_svds and JDSVD solving the problems effectively, but primme_svds method still provides much faster convergence and execution time.

Table 2: Seeking 1, 3, 5, and 10 smallest singular triplets with user tolerance 1e-8

	$\delta = 1e\text{-}8$ Matrix:	well1850		pde2961		dw2048		fidap4	
k	Method	MV	Sec	MV	Sec	MV	Sec	MV	Sec
1	primme_svds(1st stage only)	499	0.3	2175	1.8	1743	1.2	4741	3.2
1	JDSVD	1563	1.3	4269	4.8	3840	3.1	4379	3.8
1	IRRHLB	872	8.6	3755	44.6	3228	35.1	8839	97.9
3	primme_svds(1st stage only)	539	0.3	2643	2.2	2135	1.5	5661	3.8
3	JDSVD	2773	2.1	7195	8.3	5776	4.8	14334	12.7
3	IRRHLB	847	8.6	3718	44.6	3225	36.1	14303	156.2
5	primme_svds(1st stage only)	607	0.4	3118	2.6	2431	1.7	6890	5.4
5	JDSVD	4203	3.6	9076	12.2	7514	8.1	16270	16.3
5	IRRHLB	872	9.3	4301	53.8	2978	34.6	13184	152.5
10	primme_svds(1st stage only)	898	0.5	4894	4.2	3912	2.8	10087	6.8
10	JDSVD	85053	74.5	14906	20.8	11683	10.4	20934	19.7
10	IRRHLB	827	9.9	4809	63.5	3445	43.8	12025	147.6

	$\delta = 1e\text{-}8$ Matrix:	jagmesh8		lshp3025		wang3	
k	Method	MV	Sec	MV	Sec	MV	Sec
1	primme_svds(1st stage only)	5444	2.9	11403	9.5	6535	58.3
1	JDSVD	12343	8.2	37225	40.9	11353	83.8
1	IRRHLB	30105	317.7	46845	565.8	19689	632.1
3	primme_svds(1st stage only)	5915	3.2	12250	10.9	7589	58.3
3	JDSVD	13861	8.7	50282	57.5	17865	127.3
3	IRRHLB	25497	271	42201	518.4	19001	619.5
5	primme_svds(1st stage only)	6679	3.7	14126	12.5	8673	68.7
5	JDSVD	18173	12.8	66034	80.9	22441	157.3
5	IRRHLB	18314	197.8	93239	1186.6	17963	570
10	primme_svds(1st stage only)	8861	4.8	19755	18.7	19445	154.2
10	JDSVD	21209	14.7	86780	115.6	110000*	854*
10	IRRHLB	52043	608.1	110013*	1511*	16975	537

3.3 With the shift and invert technique

When the matrix can be factorized, the shift and invert (SI) Lanczos/Arnoldi method is an effective alternative. This is also the default for seeking smallest singular values in MATLAB's svds. Shift and invert is also applicable to PRIMME and primme_svds. Because the normal equations can accurately and efficiently compute the largest singular triplets of A^{-1}, there is no need for two stages or to work on the augmented B. Still, we report results with SI on both C and B.

Table 5 compares MATLAB's svds (SI on B) against primme_svds(C) (SI on C by inverting A), and against primme_svds(B) (SI on B by inverting B). We have instrumented the native svds code to return the number of iterations. To facilitate comparisons, we include the LU factorization times in the running times of all methods, but also report them separately. The tolerance is δ =1e-10, and primme_svds uses the DYNAMIC method that switches between GD+k and JDQMR to optimize performance [27].

Overall, primme_svds(C) is much faster than svds both in convergence and execution time. This is because working on C improves the separations of the required eigenvalues, and it involves computations with vectors of half the size of those in svds. Although there is no reason to work on B, we report that primme_svds(B) takes more iterations than svds because 10 triplets are computed, and it takes far less iterations when computing 1, 3, or 5 ones; similarly to our first experiment.

Table 3: Seeking 1, 3, 5, and 10 smallest singular triplets with user tolerance 1e-14

	$\delta =$ 1e-14 Matrix:	well1850		pde2961		dw2048		fidap4	
k	Method	MV	Sec	MV	Sec	MV	Sec	MV	Sec
1	primme_svds(GD+k)	730	0.5	3491	3.2	2906	2.2	6950	5.0
1	primme_svds(JDQMR)	786	0.5	3707	3.4	2899	1.9	6918	4.5
1	JDSVD	1838	1.4	6106	7.6	5061	4.5	6436	5.9
1	IRRHLB	1368	14.2	6328	75.5	4561	50.9	14078	155.4
3	primme_svds(GD+k)	882	0.6	4648	4.8	3679	3.7	8846	8.8
3	primme_svds(JDQMR)	974	0.5	5143	3.8	3746	3.1	8936	5.7
3	JDSVD	71259	59.2	10517	11.6	8911	7.1	10781	9.0
3	IRRHLB	1137	12.2	6241	77.0	4443	50.0	19059	215.4
5	primme_svds(GD+k)	1134	0.7	5879	5.7	4776	3.7	11976	9.1
5	primme_svds(JDQMR)	1252	0.9	6506	5.0	4932	4.1	12179	9.1
5	JDSVD	110000*	81.9*	14554	19.1	12266	11.7	14906	14.6
5	IRRHLB	1196	13.1	6218	79.2	4193	49.1	18098	211.2
10	primme_svds(GD+k)	1833	1.2	9463	9.6	8069	7.9	19805	16.2
10	primme_svds(JDQMR)	1961	1.3	9856	11.8	8381	7.7	20261	15.0
10	JDSVD	110000*	79.9*	24498	29.6	20351	20.5	25125	24.5
10	IRRHLB	1069	11.7	6371	86.8	4589	58.6	17393	214.2

	$\delta =$ 1e-14 Matrix:	jagmesh8		lshp3025		wang3	
k	Method	MV	Sec	MV	Sec	MV	Sec
1	primme_svds(GD+k)	7177	4.2	15159	14.6	19473	192
1	primme_svds(JDQMR)	7228	3.7	14701	11.2	18887	71.9
1	JDSVD	13608	9.6	42835	53.1	16457	105.4
1	IRRHLB	43869	466.5	57912	693	27470	1003
3	primme_svds(GD+k)	8859	5.7	18910	20.8	21560	334
3	primme_svds(JDQMR)	9128	4.5	19167	19	31365	148
3	JDSVD	17029	11.0	48731	59.4	41900	387
3	IRRHLB	31210	330.1	63806	799.5	29035	1094
5	primme_svds(GD+k)	11569	7.5	24743	27.5	27497	307.8
5	primme_svds(JDQMR)	11756	5.9	25745	20	25558	145.5
5	JDSVD	22573	15.2	62646	75.9	57454	441.4
5	IRRHLB	23498	331.4	99395	1258.9	22985	730.1
10	primme_svds(GD+k)	17344	10.8	39852	42.4	36695	551.5
10	primme_svds(JDQMR)	18022	11.2	41749	31.6	35134	237.5
10	JDSVD	29613	21.0	110000*	150.1*	91290	871.9
10	IRRHLB	55673	879.4	110013*	1511*	43309	1679.4

Table 4: Seeking ten smallest singular triplets with a good preconditioner

	$k = 10$ Matrix:	jagmesh8		lshp3025		wang3	
δ	Method	MV	Sec	MV	Sec	MV	Sec
1e-8	primme_svds(1st stage only)	92	0.2	125	2.3	181	4.0
1e-8	JDSVD	287	1.4	331	11.1	320	17.9
1e-14	primme_svds(GD+k)	177	0.6	223	3.5	320	7.5
1e-14	primme_svds(JDQMR)	308	2.1	398	5.9	525	6.8
1e-14	JDSVD	408	1.9	518	17.1	606	31.8

Table 5: primme_svds and svds using shift and invert technique for 10 smallest singular triplets

δ = 1e-10, Matrix:	pde2961		dw2048		fidap4		jagmesh8		lshp3025		wang3	
Method	MV	Sec	MV	Sec	MV	Sec	MV	Sec	MV	Sec	MV	Sec
LU(A) time	–	0.01	–	0.01	–	0.02	–	0.01	–	0.06	–	1.1
primme_svds(C)	35	0.13	33	0.10	31	0.12	26	0.09	35	0.26	57	4.4
LU(B) time	–	0.03	–	0.02	–	0.03	–	0.01	–	0.05	–	4.6
primme_svds(B)	103	0.26	100	0.18	134	0.31	69	0.11	199	0.45	137	18.7
svds	73	0.33	73	0.28	73	0.34	61	0.24	63	0.37	71	12.3

4 Conclusion

We have developed a meta-method, primme_svds, that computes a few singular triplets of large matrices by using the state-of-the-art eigensolver PRIMME in a two-stage approach. The first stage solves the normal equations as a fast way to get sufficiently accurate approximations. If futher accuracy is needed, a second stage solves the interior eigenvalue problem from the augmented matrix. We have presented several enhancements to the PRIMME eigensolver that allow for an efficient computation of the interior eigenproblem. primme_svds improves on convergence and robustness over other state-of-the-art singular value methods, but most importantly it is based on a highly optimized production software that allows its use, with or without preconditioning, in large, real world problems.

References

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

[2] G. Golub, and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, J. Soc. Indust. Appl. Math. Ser.B Numer. Anal., 2 (1965), pp. 205-224.

[3] G. Golub, F. T. Luk and M. L. Overton, *A block Lanczos method for computing the singular values and corresponding singular vectors of a matrix*, ACM Trans. Math. Software, 7 (1981), pp. 149-169.

[4] J. Cullum, R. A. Willoughby, and M. Lake, *A Lanczos algorithm for computing singular values and vectors of large matrices*, SIAM J. Sci. Statist. Comput., 4 (1983), pp. 197-215.

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

[6] M. E. Hochstenbach, *Harmonic and refined extraction methods for the singular value problem, with applications in least squares problems*, BIT, 44 (2004), pp. 721-754.

[7] B. Philippe and M. Sadkane, *Computation of the fundamental singular subspace of a large matrix*, Linear Algebra Appl., 257 (1997), pp. 77-104.

[8] J. J. Dongarra, *Improving the accuracy of computed singular values*, SIAM J. Sci. Statist. Comput., 4 (1983), pp. 712-719.

[9] Michael W. Berry, Dani Mezher, B. Philippe and Ahmed Sameh, Parallel algorithms for the singular value decomposition (Chapter 4), in Erricos John Kontoghiorghes, editor, *Handbook on parallel computing and statistics*, pages 117-164, Chapman and Hall/CRC, 2005.

[10] G. H. Golub, and C. F. Van Loan, *Matrix Computations*, 3rd ed., The John Hopkins University Press, Baltimore, London, 1996.

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

[12] J. Baglama and L. Reichel, *Restarted block Lanczos bidiagonalization methods*, Numer. Algorithms, 43 (2006), pp. 251-272.

[13] J. Baglama and L. Reichel, *An implicitly restarted block Lanczos bidiagonalization method using leja shifts*, BIT, 53 (2013), pp. 285-310.

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

[15] Z. Jia and D. Niu, *An refined harmonic Lanczos bidiagonalization method and implicitly restarted algorithm for computing the smallest singular triplets of large matrices*, SIAM J. Sci. Comput., 32 (2010), pp. 714-744.

[16] E. Kokiopoulou, C. Bekas, and E. Gallopoulos, *computing the smallest singular triplets with implicitly restarted Lanczos bidiagonalization*, Appl. Numer. Math., 49 (2004), pp. 39-61.

[17] R. M. Larsen, *Combining implicit restarts and partial reorthogonalization in Lanczos bidiagonalization*, http://soi.stanford.edu/ rmunk/PROPACK/

[18] D. Niu and X. Yuan, *A harmonic Lanczos bidiagonalization method for computing interior singular triplets of large matrices*, Appl. Math. Comput., 218 (2012), pp. 7459-7467.

[19] R.B. Morgan, *Computing interior eigenvalues for large matrices*, Linear Algebra Appl. 154/156 (1991), pp. 289-309.

[20] R.B. Morgan, M. Zeng, *Harmonic projection methods for large non-symmetric eigenvalue problems*, Numer. Linear Algebra Appl.5(1) (1998), pp. 33-55.

[21] G.L.G. Sleijpen, H.A. van der Vorst, *A Jacobi-Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl, 17(2) (1996), pp. 401-425.

[22] C.C.Paige, B.N.Parlett, H.A. van der Vorst, *Approximate solutions and eigenvalue bounds from Krylov subspaces*, Numer. Linear Algebra Appl. 2 (1995), pp. 115-134.

[23] Z.Jia, *Refined iterative algorithms based on Arnoldi's process for large unsymmetric eigenproblems*, Linear Algebra Appl, 259 (1997), pp. 1-23.

[24] G.W.Stewart, *Matrix Algorithms; Vol. II Eigensystems*, SIAM, Philadelphia, PA, 2001.

[25] A. Stathopoulos, *Nearly optimal preconditioned methods for Hermitian eigenproblems under limited memory. Part I: Seeking one eigenvalue*, SIAM J. Sci. Comput., 29(2) (2007), pp. 481-514.

[26] A. Stathopoulos and J. R. McCombs, *Nearly optimal preconditioned methods for Hermitian eigenproblems under limited memory. Part II: Seeking Many eigenvalue*, SIAM J. Sci. Comput., 29(5) (2007), pp. 2162-2188.

[27] A. Stathopoulos and James R. McCombs, *PRIMME: PReconditioned Iterative MultiMethod Eigensolver: Methods and software description*, ACM Trans. Math. Software, 37 (2010), pp. 21:1-21.30.

[28] H.A. van der Vorst, *Computational Methods for Large Eigenvalue Problems*, In P.G. Ciarlet and J.L. Lions(eds), Handbook of Numerical Analysis, Volume VIII, North-Holand, 2002, pp. 3-179.

[29] T. A. Davis, and Y. Hu, *The University of Florida Sparse Matrix Collection*, ACM Trans. Math. Software, 38 (2011), pp. 1:1-1:25.

[30] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK User's Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998.