Accelerating Iterations Involving Eigenvalue or Singular Value Decomposition by Block Lanczos with Warm Start

Siming Wei TOBIAWSM@GMAIL.COM
Zhejiang University

Zhouchen Lin ZHOULIN@MICROSOFT.COM
Microsoft Research Asia

Abstract

Many machine learning problems are solved by algorithms that involve eigenvalue decomposition (EVD) or singular value decomposition (SVD) in each iteration. Therefore, these algorithms suffer from the high computation cost of multiple EVD/SVDs. To relieve this issue, we introduce the block Lanczos method to replace the original exact EVD/SVD in each iteration by solving it approximately, yet still at a high precision. We also propose to utilize the subspace obtained in the previous iteration to start the block Lanczos procedure. Examples of three popular problems are presented to show how our block Lanczos with warm start (BLWS) technique can be adopted to accelerate its host algorithms. Experimental results show that our BLWS technique usually accelerates its host algorithms by at least two times.

1. Introduction

In machine learning community, a large family of problems are formulated as optimization models. Some of them, such as Linear Discriminate Analysis (Wang et al., 2007), Robust Principle Component Analysis (Wright et al., 2009), Matrix Completion (Cai et al., 2008) and other nuclear norm minimization problems, are often solved by iterative methods which involve eigenvalue decomposition (EVD) or singular value decomposition (SVD) in each iteration. The EVD/SVDs are usually the most expensive part of those algorithms. Hence it is valuable to find a way to accelerate the iterations.

Microsoft Technical Report #MSR-TR-2010-162.

To make the acceleration possible, a key observation is that it may be unnecessary to solve the EVD/SVD in each iteration exactly. A naive way is that one computes the EVD/SVD at low precision at the beginning and increase the precision steadily when the iteration goes on. However, this is not the optimal way to acceleration, because EVD and SVD are of $O(p^3)$ complexity, where $p = \min(m, n)$ and $m \times n$ is the size of the matrix. Although sometimes the full EVD/SVD can be replaced by partial EVD/SVD, i.e., only the leading or tailing eigenvalue/vectors or singular value/vectors are computed (e.g., by using `laneig` or `lansvd` (Larsen, 1998)), this strategy still does not fully exploit the properties of iterations. To proceed, we had better introduce the Lanczos method for partial EVD/SVD first.

1.1. Lanczos Method for Partial EVD/SVD

The Lanczos method is to find the optimal leading/tailing eigen-subspace of a matrix A in a Krylov subspace (Golub & Loan, 1996):

$$K(A, q_1, k) = \text{span}\{q_1, Aq_1, \cdots, A^{k-1}q_1\}. \quad (1)$$

The orthonormal basis Q_k of $K(A, q_1, k)$ can be efficiently computed via the Lanczos procedure shown in Algorithm 1. Accordingly, A can be approximated as $A \approx Q_k T_k Q_k^T$, where T_k is a tridiagonal matrix:

$$T_k = \begin{pmatrix} \alpha_1 & \beta_1 & \cdots & 0 \\ \beta_1 & \alpha_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \beta_{k-1} \\ 0 & \cdots & \beta_{k-1} & \alpha_k \end{pmatrix}. \quad (2)$$

The Lanczos procedure is actually derived by comparing the left and right hand sides of $AQ_k \approx Q_k T_k$ (cf. Section 2.1).

An important property of the Lanczos method is that the leading/tailing eigenvalues of T_k (called the Ritz

Algorithm 1 The Lanczos Procedure

```
**Input:** $m \times m$ symmetric matrix $A$, $k$.
1. Initialization: $r_0 = q_1$; $\beta_0 = 1$; $q_0 = 0$; $l = 0$.
2.
for $l = 1 : k-1$ do
    $q_{l+1} = r_l/\beta_l$; $l = l+1$; $\alpha_l = q_l^T A q_l$;
    $r_l = Aq_l - \alpha_l q_l - \beta_{l-1} q_{l-1}$;
    $\beta_l = \|r_l\|_2$;
end for
Output: $Q_k = (q_1, \cdots, q_k)$ and $T_k$ as (2).
```

values of A) converge to those of A quickly when k grows, while $Q_k V$ (called the Ritz vectors of A) also converges to the leading/tailing eigenvectors of A quickly, where V consists of the leading/tailing eigenvectors of T_k. As the Lanczos method only requires solving the EVD of a relatively small tri-diagonal matrix T_k, partial EVD is usually faster than full EVD when the required number of eigenvectors is relatively small.

The Lanczos method can also be applied to compute the partial SVD (Larsen, 1998) of a matrix A. It is based on implicitly applying the Lanczos procedure to the following matrix (cf. Section 2.2)

$$\tilde{A} = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}. \tag{3}$$

Accordingly, A can be approximated as $A \approx U_k B_k V_k^T$, where columns of U_k and V_k are orthonormal and B_k is bi-diagonal. The convergence of Ritz values/vectors, obtained by computing the SVD of a relatively small bi-diagonal matrix B_k, to the singular values/vectors of A is also fast when k increases. For more details, please refer to (Larsen, 1998).

1.2. Ideas to Improve

One can see that the Lanczos procedure starts from an initial vector q_1, which is usually randomly chosen if no apriori information is available, making the size k of T_k unpredictable. If q_1 is not good, k can be relatively large in order for the Ritz values/vectors to achieve prescribed precision, making the partial EVD/SVD inefficient. Actually, during the iterations of optimization, as the matrices A_i and A_{i+1} to compute the EVD/SVDs in successive iterations are close to each other, any of the leading/tailing Ritz vectors of A_i should be good for initializing the Lanczos procedure of A_{i+1}. However, a risk of this way is that the Lanczos procedure may terminate quickly by outputting a small invariant subspace containing the previous Ritz vector because the previous Ritz vector is close to be an eigenvector of A_{i+1} and the Lanczos procedure terminates when an invariant subspace is found (Golub & Loan, 1996). So the current Lanczos procedure may fail and has to restart with another initial vector. Moreover, this strategy neglects the fact that we are seeking an optimal *subspace*. As the computed subspaces in successive iterations should be close to each other, we should fully utilize the information provided by the previous subspace. This motivates us to adopt the block Lanczos method for partial EVD/SVD.

1.3. Our Contributions

The block Lanczos method is a natural generalization of the standard Lanczos method by replacing the Krylov space with

$$\tilde{K}(A, X_1, k) = \text{span}\{X_1, AX_1, \cdots, A^{k-1}X_1\}, \tag{4}$$

where X_1 is an orthonormal basis of an initial subspace. Accordingly, the Lanczos procedure is generalized to the block Lanczos procedure (see Algorithm 2).

We propose the following ways to accelerate the iterations with EVD/SVDs:

1. introduce the block Lanczos method for partial EVD/SVD.
2. initialize X_1 with the subspace obtained in the previous iteration.
3. keep k in the block Lanczos procedure small such that EVD/SVD is only computed at high enough precision.

The intuition behind our method is that the block Lanczos method operates on subspaces. So it is a natural way to update eigen-subspaces. As subspaces in successive iterations should be close to each other, the subspace obtained in the previous iteration should be a good initialization for the current block Lanczos procedure. Although we do not compute EVD/SVD exactly in each iteration, as the initialization is good, a small k can still result in a high-precision eigen/singular subspace, considering that the leading/tailing Ritz vectors converge fast to the true leading/tailing eigen/singular vectors. Note that a small k leads to much smaller computation load in computing EVD/SVD. With such a warm start, compared with the standard Lanczos method, the risk of terminating with a small invariant subspace is gone, and the subspace can be updated more efficiently. As a result, the whole optimization process can be sped up a lot.

As examples, we apply our block Lanczos with warm start (BLWS) technique to two popular problems: Robust PCA (RPCA) (Wright et al., 2009), and matrix completion (MC) problems (Candès & Recht, 2008). For both the RPCA and Matrix Completion problems, using BLWS can result in at least two or three times speedup.

The rest of paper is organized as follows. In Section 2, we introduce our block Lanczos with warm start in more detail. In Section 3, we introduce two exemplar problems, the RPCA and MC problems, that involve SVD in each iteration. In Section 4, we present some implementation details of adopting BLWS in different algorithms for the exemplar problems and show the speed up effect of applying BLWS. In Section 5, we gives more detailed discussions on BLWS. Finally, we conclude the paper in Section 6.

2. Block Lanczos with Warm Start

2.1. Block Lanczos with Warm Start for Partial EVD

The block Lanczos procedure is to find an approximation of A: $A \approx Q_k T_k Q_k^T$, where T_k is a block tridiagonal matrix (Golub & Loan, 1996):

$$T_k = \begin{pmatrix} M_1 & B_1^T & \cdots & 0 \\ B_1 & M_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & B_{k-1}^T \\ 0 & \cdots & B_{k-1} & M_k \end{pmatrix} \tag{5}$$

and columns of $Q_k = (X_1, \cdots, X_k)$ are orthonormal. By comparing the left and right hand sides of $Q_k A = Q_k T_k$, we have

$$AX_l = X_{l-1}B_{l-1}^T + X_l M_l + X_{l+1}B_l, \quad l = 1, \cdots, k-1, \tag{6}$$

where B_0 is defined to be 0. From the orthogonality of Q, we have that

$$M_l = X_l^T A X_l, \quad l = 1, \cdots, k. \tag{7}$$

Moreover, if we define $R_l = AX_l - X_l M_l - X_{l-1}B_{l-1}^T$, then $X_{l+1}B_l$ is the QR decomposition of R_l. This leads to the block Lanczos procedure in Algorithm 2.

After the approximation $A \approx Q_k T_k Q_k^T$ is obtained, one may further compute the EVD of T_k: $T_k = U_k \Lambda_k U_k^T$, where the eigenvalues λ_i are ordered from large to small. Then the leading d eigenvalues and eigenvectors of A is approximated by $\lambda_1, \cdots, \lambda_d$, and $Q_k U_k(:, 1:d)$, respectively. The tailing eigenvalue/vectors are obtained similarly. The whole process is summarized as Algorithm 3.

If we denote the block Lanczos for EVD (Algorithm 3) as $BL\_EVD(A, X_1, k)$, then our BLWS can be written as:

$$\textbf{(BLWS)} \quad (U_j, \Sigma_j) = BL\_EVD(A_j, U_{j-1}, k_j).$$

Algorithm 2 Block Lanczos Procedure

```
Input: $m \times m$ symmetric matrix $A$, $m \times d$ orthogonal matrix $X_1$, $k$.
1. Initialization: $M_1 = X_1^T A X_1$; $B_0 = 0$.
2.
for $l = 1 : k-1$ do
    $R_l = AX_l - X_l M_l - X_{l-1}B_{l-1}^T$;
    $(X_{l+1}, B_l) = qr(R_l)$; (The QR decomposition)
    $M_{l+1} = X_{l+1}^T A X_{l+1}$;
end for
Output: $Q_k = (X_1, \cdots, X_k)$ and $T_k$ as in (5).
```

Algorithm 3 Block Lanczos for EVD

```
Input: $m \times m$ symmetric matrix $A$, $m \times d$ orthogonal matrix $X_1$, $k$.
1. Compute $Q_k$ and $T_k$ by Algorithm 2.
2. Compute EVD of $T_k$: $T_k = U_k \Lambda_k U_k^T$, where the eigenvalues on the diagonal of $\Lambda$ are in a decreasing order.
Output: $X = Q_k U_k(:, 1:d)$, $\Sigma = \Lambda(1:d, 1:d)$.
```

2.2. Block Lanczos with Warm Start for Partial SVD

The block Lanczos can also be applied to $\tilde{A}$ in (3) to compute the partial SVD of A. This is based on the following theorem (Larsen, 1998):

Theorem 2.1. *If the SVD of an $m \times n$ $(m \le n)$ matrix A is $A = U\Sigma V^T$, then the EVD of $\tilde{A}$ is*

$$\tilde{A} = \tilde{U} \begin{pmatrix} \Sigma & 0 & 0 \\ 0 & -\Sigma & 0 \\ 0 & 0 & 0 \end{pmatrix} \tilde{U}^T, \tag{8}$$

where

$$\tilde{U} = \frac{1}{\sqrt{2}} \begin{pmatrix} U_1 & U_1 & \sqrt{2}U_2 \\ V & -V & 0 \end{pmatrix} \quad \textit{and} \quad (U_1, U_2) = U. \tag{9}$$

So by computing the EVD of $\tilde{A}$, the SVD of A can be obtained. Thus the BLWS for EVD can be trivially extended to BLWS for SVD, by using $\tilde{X}_1 = \frac{1}{\sqrt{2}}(U^T, V^T)^T$ as the initial subspace, where U and V are the estimated left and right singular spaces obtained in the previous iteration. Note that $\tilde{A}$ is of special structure. So the block Lanczos procedure can be done efficiently by identifying the zero sub-matrices of A.

If one starts the block Lanczos procedure for $\tilde{A}$ from $\tilde{X}_1 = (U^T, 0)^T$ or $\tilde{X}_1 = (0, V^T)^T$, s/he can obtain a more compact recursive relationship that resembles that in `lansvd` (Larsen, 1998). However, such an initialization only utilizes half of the information provided by the previous iteration. So the precision of obtained subspace is not as high as the one obtained by initializing with $\tilde{X}_1 = \frac{1}{\sqrt{2}}(U^T, V^T)^T$.

3. Exemplar Problems that Can Use BLWS

In this section, we introduce two popular problems in machine learning: the robust PCA and matrix completion problems, which need multiple SVDs.

3.1. The Robust PCA Problem

The Robust PCA is a variant of traditional PCA. RPCA is modeled as the following convex optimization problem (Wright et al., 2009):

$$\min_{A,E} \|A\|_* + \lambda \|E\|_1, \quad \text{subject to } D = A + E, \quad (10)$$

where $\|\cdot\|_*$ denotes the nuclear norm of a matrix (i.e., the sum of its singular values), $\|\cdot\|_1$ denotes the l_1-norm of a matrix computed as the sum of the absolute values of matrix entries, and λ is a positive weighting parameter. RPCA is to decompose a matrix D into a low-rank matrix A and a sparse matrix E. It has found wide applications in computer vision, such as background extraction and face shadow and highlight removal (Wright et al., 2009).

Many algorithms (Wright et al., 2009; Yuan & Yang, 2009; Tao & Yuan, 2009; Lin et al., 2009) have been proposed to solve (10). One of the fastest algorithms is Lin et al.'s inexact augmented Lagrangian multiplier (IALM) method (Lin et al., 2009). The method works on the augmented Lagrangian function of (10):

$$\begin{aligned} L(A, E, Y, \mu) &= \|A\|_* + \lambda \|E\|_1 \\ &+ \langle Y, D - A - E\rangle + \frac{\mu}{2}\|D - A - E\|_F^2. \end{aligned} \quad (11)$$

The iteration goes as follows:

1. $A_{k+1} = \arg\min_A L(A, E_k, Y_k, \mu_k)$;
2. $E_{k+1} = \arg\min_E L(A_{k+1}, E, Y_k, \mu_k)$;
3. $Y_{k+1} = Y_k + \mu_k(D - A_{k+1} - E_{k+1})$;
4. $\mu_{k+1} = \rho\mu_k$.

where $\rho \in [1, 1.5]$ is a parameter. Step 1 requires an SVD to update A:

$$A_{k+1} = U\Theta_{\mu_k^{-1}}(\Sigma)V^T, \quad (12)$$

where $U\Sigma V^T$ is the SVD of $D - E_k + \mu_k^{-1}Y_k$ and Θ is a shrinkage operator:

$$\Theta_\varepsilon(x) \doteq \begin{cases} x - \varepsilon, & \text{if } x > \varepsilon, \\ x + \varepsilon, & \text{if } x < -\varepsilon, \\ 0, & \text{otherwise,} \end{cases} \quad (13)$$

We will apply BLWS to accelerate step 1.

3.2. The Matrix Completion Problem

The matrix completion problem is to recover a low rank matrix when only a few entries of the matrix is observed. It is ubiquitous in a lot of areas such as machine learning, control and computer vision (Abernethy et al., 2006). A famous example is the Netflix problem[1]. Recently, Candès and Recht (Candès & Recht, 2008) proved that the low rank matrix can be exactly recovered with high probability under some probability model. Its optimization model is as follows:

$$\min_A \|A\|_*, \quad \text{subject to} \quad A_{ij} = D_{ij}, \quad \forall(i,j) \in \Omega. \quad (14)$$

There are many algorithms in this line of work. Among them, singular value thresholding (SVT) by Cai et al. (Cai et al., 2008) is a famous one. Although the IALM for the MC problem is faster than SVT (Lin et al., 2009), we have presented the IALM for the RPCA problem in Section 3.1. So here we choose the SVT algorithm. The iterations of SVT goes as follows:

$$\begin{aligned} A_k &= U\Theta_\varepsilon(\Sigma)V^T; \\ Y_k &= Y_{k-1} + \delta_k P_\Omega(D - A_k); \end{aligned} \quad (15)$$

where $U\Sigma V^T$ is the SVD of Y_{k-1}, P_Ω is an operator that keeps the values in Ω and fills those outside Ω zeros, and δ_k is the step size. We will apply BLWS to accelerate the first step of the iterations.

4. Implementation and Experimental Results

In this section, we show details of implementing BLWS on the RPCA and MC problems. For each problem, we compare the original algorithms introduced in Section 3 and their BLWS improved versions in the aspect of computation time. Accuracy of obtained solutions is also shown in order to ensure that the correct solutions are approached. All experiments are run on the same workstation with Intel Xeon E5540 2.53GHz with 4 cores, running Windows Server 2008 and Matlab (Version 7.7).

[1]http://www.netflixprize.com/

4.1. BLWS for RPCA

The implementation of BLWS for the IALM method (Lin et al., 2009) for the RPCA problem (10) is not straightforward, for the intrinsic rank r of A is unknown. However, the rank prediction mechanism in (Lin et al., 2009) can provide a dynamic estimate on r and it turns out that the estimated rank stablizes quickly. When the predicted rank is larger or smaller than the current block size d, we simply use more or less Ritz vectors. As the estimated rank converges fast (usually less than 7 iterations), such adaption does not affect the effectiveness of BLWS.

For the RPCA problem, we generate the synthetic data in the same way as that in (Lin et al., 2009). A is generated according to the independent random orthogonal model, E is a sparse matrix whose support is independent and uniformly distributed in $[-500, 500]$, and $D = A + E$. For simplicity, we only focus on $m \times m$ square matrices and the parameter λ is fixed at $1/\sqrt{m}$. The m is chosen in $\{500, 1000, 2000, 3000\}$. The rank of A is chosen as $10\%m$, and the number of corrupted entries (i.e., $\|E\|_0$) is $10\%m^2$. In the IALM method (see Section 3.1), ρ is set to be 1.3. It terminates when $\frac{\|D-A_k-E_k\|_F}{\|D\|_F} \leq 10^{-6}$, where $\|\cdot\|_F$ denotes the Frobenius norm.

Table 4.1 shows detailed comparison between IALM and BLWS accelerated IALM. We can see that BLWS-IALM roughly costs less than 1/3 time of IALM, achieving the same accuracy, and the total number of iterations only increases slightly.

4.2. BLWS for MC

The implementation of BLWS for the SVT algorithm for the matrix completion problem is similar to that for RPCA, where a rank prediction mechanism for matrix completion (Lin et al., 2009) is used. The SVT algorithm can be efficiently implemented by using low-rank representation of A_k and sparse representation for Y_k. Adopting BLWS does not affect such efficiency.

The data for MC is generated in the same way in (Cai et al., 2008). The $m \times m$ matrices of rank r is generated by sampling two $m \times r$ factors M_L and M_R independently, each having i.i.d. Gaussian entries, and then setting $M = M_L M_R^*$. Finally, the set of observed entries is sampled uniformly at random among all sets of cardinality m.

The comparison results are shown in Table 4.2. We can see that the time of BLWS accelerated SVT is usually much less than half of that of SVT and the total number of iterations hardly changes.

5. Discussions

In the last two sections, we have presented three examples to show how BLWS works. These three problems are typical representatives for those algorithms who may take advantage of BLWS. They can involve EVD or SVD. The block size d can be known or unknown. The matrices can be sparse or non-sparse. And BLWS can be used all the way or only partly. Nonetheless, they all have some properties in common. Actually, BLWS is suitable for algorithms having the following properties:

1. The algorithm is iterative;
2. Each iteration involves an EVD or SVD that uses information from the previous iteration;
3. Let Z be the limit of the solutions $\{Z_1, Z_2, \cdots\}$ of iterations. Only the d largest eigenspace/singular space of Z is used to determine the optimal solution of the optimization problem.

In the BLWS scheme, we usually set a relatively small integer k of the blocks in T_k (see (5)). Though a larger k gives more precise estimation of the eigen/singular space, it is more expensive to compute the EVD of T_k. Besides, when k is large there will be loss of orthogonality among Lanczos blocks and reorthogonalization, which is expensive, is needed to guarantee enough precision (Golub & Loan, 1996).

The BLWS can be further sped up. In our current implementation, the EVD of matrix T_k is computed directly by treating T_k as a general dense matrix. Actually, it is block tri-diagonal and there are fast methods to compute the EVD of such matrices (Schwartz, 1968).

6. Conclusions

In this paper, we introduce the block Lanczos technique to accelerate iterative algorithms involving EVD/SVD in each iteration. Since the block Lanczos method is much cheaper than EVD/SVD when the size of matrix is large, and the warm start takes full advantage of the information from last iteration, the total time for optimization can be greatly reduced. A large family of algorithms may benefit from our BLWS technique. Three typical examples are selected to show how BLWS works. The extensive experimental results indicate that our BLWS technique usually makes its host algorithm at least twice faster.

Although the convergence of BLWS accelerated algorithms to the true solution is testified by our experiments, we still have to investigate its convergence

Table 1. BLWS-IALM vs. IALM on different synthetic data. $\hat{A}$ and $\hat{E}$ are the computed low rank and sparse matrices and A is the ground truth.

m	$method$	$\frac{\|\hat{A}-A\|_F}{\|A\|_F}$	$rank(\hat{A})$	$\|\hat{E}\|_0$	$\#iter$	$time(s)$
500	IALM	5.27e-006	50	25009	30	4.07
500	BLWS-IALM	9.64e-006	50	25008	30	2.07
1000	IALM	3.99e-006	100	100021	29	23.09
1000	BLWS-IALM	6.05e-006	100	100015	30	9.25
2000	IALM	2.80e-006	200	400064	28	154.80
2000	BLWS-IALM	4.30e-006	200	400008	30	53.37
3000	IALM	2.52e-006	300	900075	28	477.13
3000	BLWS-IALM	3.90e-006	300	900006	30	149.19

Table 2. BLWS-SVT vs. SVT on different synthetic data. $\hat{A}$ is the recovered low rank matrix and A is the ground truth. m is the size of matrix and n is the number of sampled entries. $d_r = r(2m - r)$ is the number of degrees of freedom in an $m \times m$ matrix of rank r.

m	r	n/d_r	n/m^2	$algorithm$	$time(s)$	$\#iter$	$\frac{\|\hat{A}-A\|_F}{\|A\|_F}$
5000	10	6	0.024	SVT	72.57	123	1.73e-004
5000	10	6	0.024	BLWS-SVT	20.02	123	1.74e-004
5000	50	5	0.1	SVT	438.81	107	1.63e-004
5000	50	5	0.1	BLWS-SVT	279.08	108	1.55e-004
5000	100	4	0.158	SVT	1783.26	122	1.73e-004
5000	100	4	0.158	BLWS-SVT	1175.91	122	1.74e-004
10000	10	6	0.012	SVT	135.90	123	1.68e-004
10000	10	6	0.012	BLWS-SVT	42.80	123	1.70e-004
10000	50	5	0.050	SVT	1156.25	110	1.58e-004
10000	50	5	0.050	BLWS-SVT	736.01	110	1.60e-004
20000	10	6	0.006	SVT	251.13	123	1.74e-004
20000	10	6	0.006	BLWS-SVT	101.47	124	1.68e-004
30000	10	6	0.004	SVT	449.34	124	1.75e-004
30000	10	6	0.004	BLWS-SVT	171.40	125	1.69e-004

properties and carry out a rigorous theoretical analysis. Moreover, we will also implement the further acceleration technique mentioned in Section 5 to make BLWS even faster.

References

Abernethy, J., Bach, F., Evgeniou, T., and Vert, J.-P. Low-rank matrix factorization with attributes. Technical report, 2006.

Cai, J., Candès, E., and Shen, Z. A singular value thresholding algorithm for matrix completion. Preprint, 2008.

Candès, E.J. and Recht, B. Exact low-rank matrix completion via convex optimization. In *Communication, Control, and Computing, 46th Annual Allerton Conference on*, pp. 806–812, Sept. 2008.

Golub, G. and Loan, C. *Matrix computations*. The Johns Hopkins University Press, 1996.

Larsen, R.M. Lanczos bidiagonalization with partial reorthogonalization. Department of Computer Science, Aarhus University, Technical report, DAIMI PB-357, code available at http://soi.stanford.edu/~rmunk/PROPACK/, Sep 1998.

Lin, Z., Chen, M., Wu, L., and Ma, Y. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *UIUC Technical Report UILU-ENG-09-2215*, 2009.

Schwartz, H.R. Tridiagonalization of a symetric band matrix. *Numerical Mathematics*, 12:231–241, 1968.

Tao, M. and Yuan, X. Recovering low-rank and sparse components of matrices from incomplete and noisy observations. Preprint, 2009.

Wang, H., Yan, S., Xu, D., Tang, X., and Huang, T. Trace ratio vs. ratio trace for dimensionality reduction. In *Computer Vision and Pattern Recognition*, 2007.

Wright, J., Ganesh, A., Rao, S., Peng, Y., and Ma, Y. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Advances in Neural Information Processing Systems 22*, pp. 2080–2088, 2009.

Yuan, X. and Yang, J. Sparse and low-rank matrix decomposition via alternating direction methods. Preprint, 2009.