A KRYLOV-SCHUR APPROACH TO THE TRUNCATED SVD

MARTIN STOLL*

Abstract. Computing a small number of singular values is required in many practical applications and it is therefore desirable to have efficient and robust methods that can generate such truncated singular value decompositions. A new method based on the Lanczos bidiagonalization and the Krylov-Schur method is presented. It is shown how deflation strategies can be easily implemented in this method and possible stopping criteria are discussed. Numerical experiments show that existing methods can be outperformed on a number of real world examples.

AMS subject classifications. Primary 65F30, 65F50, 65F20 Secondary 92B05

Key words. Singular Value Decomposition, Krylov subspaces, Krylov-Schur method

1. Introduction. In [9] Golub and Kahan show how to efficiently compute the Singular Value Decomposition of a matrix $A \in \mathbb{R}^{M\times N}$ $(M > N)$ which is given by $A = U\Sigma V^T$ with $U \in \mathbb{R}^{M\times M}$ and $V \in \mathbb{R}^{N\times N}$ are orthogonal matrices. $\Sigma \in \mathbb{R}^{M\times N}$ has a diagonal N, N block containing the singular values

$$\sigma_1 \geq \sigma_2 \geq \ldots \sigma_N.$$

The computation of the SVD is based on the bidiagonal factorization

$$\begin{array}{rcl} AV_m & = & U_m B_m \\ A^T U_m & = & V_m B_m^T + \beta_{m+1} v_{m+1} e_m^T \end{array} \tag{1.1}$$

with

$$B_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ & \alpha_2 & \beta_3 & & \\ & & \ddots & \ddots & \\ & & & \alpha_{m-1} & \beta_m \\ & & & & \alpha_m \end{bmatrix}$$

introduced by Golub and Kahan in [9]. This decomposition also plays an important role when solving least squares systems as it is the basis for the LSQR method proposed by Paige and Saunders in [23]. A more algorithmic form of (1.1) is given by

$$\begin{array}{rcl} \beta_{j+1} v_{j+1} & = & A^T u_j - \alpha_j v_j \\ \alpha_{j+1} u_{j+1} & = & A v_{j+1} - \beta_{j+1} u_j \end{array} \tag{1.2}$$

which can be straightforwardly used for an implementation.

The singular value decomposition is an important tool in many areas such as signal processing. Some applications such as image analysis or model reduction only require a small number of singular values and singular vectors. Therefore, the computation of the truncated SVD

$$\tilde{A} = U_l \Sigma_l V_l^T = \sum_{i=1}^{l} \sigma_i u_i v_i^T$$

*Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford, OX1 3QD, United Kingdom, (martin.stoll@comlab.ox.ac.uk)

with $l \leq r$ is important. The matrix $\tilde{A}$ represents the best low-rank approximation to A in the Frobenius and 2-norm, see [10, 32].

When one is interested in a small number of eigenvalues, restarted methods such as the implicitly restarted Arnoldi process are the methods of choice, see [27, 28, 33, 3, 29] for more details. Extending these techniques to the case when one is interested in a small number of singular values seems natural and many examples of such techniques can be found in the literature, see [2, 14, 15, 16, 25]. We describe the particular method introduced by Baglama and Reichel (cf. [2]) in Section 2 and present a comparison of numerical results in Section 7.

In this paper we show how the Golub-Kahan bidiagonalization procedure (1.1) can be used to efficiently implement a method that computes the truncated SVD of A as well as allowing an easy implementation of deflation techniques. In more detail, in the case of unwanted but converged eigenvalues/singular values the deflation process is called *purging* and in the case of converged but wanted eigenvalues/singular values the deflation technique that has to be used is called *locking*.

Implementing these techniques for the restarted Arnoldi algorithm is far from trivial, see [27, 26, 20]. A method that allows a relatively easy implementation of these strategies is the so-called *Krylov-Schur algorithm* introduced by Stewart in 2001 (cf. [29, 30]). The Krylov-Schur algorithm is based on the Krylov decomposition instead of the Arnoldi decomposition. An adoption of this strategy for Hamiltonian and skew-Hamiltonian matrices was already successfully demonstrated, see [4, 31, 19]. In this paper we illustrate how the Krylov-Schur strategy can be adopted for the bidiagonal factorization of Golub and Kahan. We discuss purging and locking as well as the implementation issues that might arise. Numerical experiments for real world examples underline the competitiveness of our method.

2. The method of Baglama and Reichel. In this section we quickly review the method introduced by Baglama and Reichel in [2] which uses a thick restart technique [33]. Recently, Hernández et al. analyzed a parallel implementation of this method, see [12].

The basic idea is very similar to that of restarted Lanczos or Arnoldi methods where the full factorization of dimension m

$$\begin{aligned} AV_m &= U_m B_m \\ A^T U_m &= V_m B_m^T + \beta_{m+1} v_{m+1} e_m^T \end{aligned} \tag{2.1}$$

is reduced to a smaller factorization which contains the relevant desired (spectral) information, in this case l singular values and singular vectors. Precisely, we reduce to a factorization of size $l+1$ with $l < m$

$$\begin{aligned} A\breve{V}_{l+1} &= \breve{U}_{l+1} \breve{B}_{l+1} \\ A^T \breve{U}_{l+1} &= \breve{V}_{l+1} \breve{B}_{l+1}^T + \breve{\beta}_{l+1} \breve{v}_{l+2} e_l^T \end{aligned} \tag{2.2}$$

where the matrices $\breve{U}_{l+1}$, $\breve{V}_{l+1}$ and $\breve{B}_{l+1}^T$ represent the information about the desired singular values and vectors. In particular, $\breve{V}_{l+1}$ and $\breve{U}_{l+1}$ represent approximations to the right and left singular vectors respectively. Baglama and Reichel [2] proposed to choose

$$\breve{V}_{l+1} = [q_1, q_2, \ldots, q_l, v_{l+1}]$$

where $q_j = V_m y_i$ are the Ritz vectors with y_i a right singular vector of B_m. This matrix is orthogonal by construction. Furthermore, we define

$$\breve{U}_{l+1} = [p_1, p_2, \ldots, p_l, \breve{u}_{l+1}]$$

where the $p_j = U_m x_i$ are Ritz vectors with x_i left singular vector of B_m and is unit vector $\breve{u}_{l+1} = \tilde{u}_{l+1} / \|\tilde{u}_{l+1}\|$ where

$$\tilde{u}_{l+1} = A v_{k+1} - \sum_{i=1}^{l} \rho_i p_i$$

is a Gram-Schmidt orthogonalization of Av_{k+1} against the vectors p_i. The matrix $\breve{B}_{l+1}$ has no longer diagonal structure; instead an additional spike appears in the last column, i.e.

$$\breve{B}_{l+1} = \begin{bmatrix} \sigma_1 & & & & \rho_1 \\ & \sigma_2 & & & \rho_2 \\ & & \ddots & & \vdots \\ & & & \sigma_l & \rho_l \\ & & & & \alpha_{l+1} \end{bmatrix}$$

where $\alpha_{l+1} = \|\tilde{u}_{l+1}\|$ and σ_i the singular values of B_m. To obtain a factorization of the form (2.2) the parameters $\breve{\beta}_{l+1}$ and $\breve{v}_{l+2}$ have to be determined. This can be done setting $\breve{v}_{l+2} = \tilde{v}_{l+2} / \|\tilde{v}_{l+2}\|$ with $\tilde{v}_{l+2} = A^T \breve{u}_{l+1} - \alpha_{l+1} v_{l+1}$ and $\breve{\beta}_{l+1} = \|\tilde{v}_{l+2}\|$. This setup now yields a factorization

$$\begin{array}{rcl} A\breve{V}_{l+1} & = & \breve{U}_{l+1} \breve{B}_{l+1} \\ A^T \breve{U}_{l+1} & = & \breve{V}_{l+1} \breve{B}_{l+1}^T + \breve{\beta}_{l+1} \breve{v}_{l+2} e_l^T \end{array} \tag{2.3}$$

which can be extended to a factorization of dimension m using the standard Golub-Kahan bidiagonalization to give

$$\begin{array}{rcl} A\breve{V}_{m} & = & \breve{U}_{m} \breve{B}_{m} \\ A^T \breve{U}_{m} & = & \breve{V}_{m} \breve{B}_{m}^T + \breve{\beta}_{m} \breve{v}_{m+1} e_l^T \end{array} \tag{2.4}$$

with

$$\breve{B}_m = \begin{bmatrix} \sigma_1 & & & & \rho_1 & & & & \\ & \sigma_2 & & & \rho_2 & & & & \\ & & \ddots & & \vdots & & & & \\ & & & \sigma_l & \rho_l & & & & \\ & & & & \alpha_{l+1} & \beta_{l+1} & & & \\ & & & & & \ddots & \ddots & & \\ & & & & & & \alpha_{m-1} & \beta_{m-1} & \\ & & & & & & & \alpha_m & \end{bmatrix}.$$

This is the basis for an iterative procedure and we compare this process to the method we introduce in the next section. Here we only reviewed the approach of Baglama and Reichel that uses Ritz values as a basis for the restart. In [2] another approach using harmonic Ritz values is presented. This method is not discussed here but numerical results are mentioned for comparison in Section 7.

3. The Krylov-Schur approach. The implicitly restarted Arnoldi process proposed by Sorensen in [28] is a very powerful tool to compute a few eigenvalues of a large sparse matrix. There are some drawbacks to the method that are mainly concerned with purging and locking eigenvalues during the iteration process. These deflation issues are addressed in [20, 27, 26] where it is illustrated that incorporating these techniques is non-trivial. Stewart was able to address these issues, see [29, 30], by introducing a Krylov-Schur method based on a slightly more general decomposition as the basis of the restarted process. This procedure was adopted in [31, 4] for the case of the implicitly restarted Hamiltonian Lanczos process introduced by Benner and Faßbender in [3]. Based on Stewart's technique we introduce a new factorization that will allow us to deflate certain singular values in the bidiagonalization process proposed by Golub and Kahan (cf. [9].

Let us assume that we are looking for the l largest singular values of the matrix A. We would then typically create a search space of roughly twice the size, ie. $m \approx 2l$. Hence, the starting point of our derivation is a m-dimensional bidiagonal factorization

$$\begin{aligned} AV_m &= U_m B_m \\ A^T U_m &= V_m B_m^T + \beta_{m+1} v_{m+1} e_m^T. \end{aligned} \tag{3.1}$$

We can now cheaply compute the Singular Value Decomposition of the small $m \times m$ matrix $B_m = P_m \Sigma_m Q_m^T$ and substitute this into (3.1) which then gives

$$\begin{aligned} AV_m &= U_m P_m \Sigma_m Q_m^T \\ A^T U_m &= V_m Q_m \Sigma_m P_m^T + \beta_{m+1} v_{m+1} e_m^T. \end{aligned} \tag{3.2}$$

We now multiply the first Equation in (3.2) by Q_m and the second equation by P_m. The result is

$$\begin{aligned} A\tilde{V}_m &= \tilde{U}_m \Sigma_m \\ A^T \tilde{U}_m &= \tilde{V}_m \Sigma_m + \beta_{m+1} v_{m+1} p_m^T \end{aligned} \tag{3.3}$$

where $e_m^T P_m = p_m^T$, $\tilde{U}_m = U_m P_m$ and $\tilde{V}_m = V_m Q_m$.

DEFINITION 3.1. *A factorization of the form*

$$\begin{aligned} AV_m &= U_m \Sigma_m \\ A^T U_m &= V_m \Sigma_m + \beta_{m+1} v_{m+1} p_m^T \end{aligned}$$

with V_m, U_m orthogonal and Σ_m a diagonal matrix coming from the SVD of the bidiagonal matrix is called Krylov-Golub-Kahan (KGK) factorization.

Since Σ_m is a diagonal matrix, we can easily swap the diagonal elements by just using permutation matrices. The swapping of diagonal elements represents the desired deflation techniques, see Section 4 for details. Here, we assume that the l singular values that represent approximation to the desired ones can be moved into the left upper corner. In the next step the $m - l$ elements in the south-east corner of the permuted diagonal matrix can be neglected for further computations. The swapping can be represented by applying the permutation Π_m to (3.3) and get

$$\begin{aligned} A\tilde{V}_m \Pi_m &= \tilde{U}_m \Pi_m \Pi_m^T \Sigma_m \Pi_m \\ A^T \tilde{U}_m \Pi_m &= \tilde{V}_m \Pi_m \Pi_m^T \Sigma_m \Pi_m + \beta_{m+1} v_{m+1} p_m^T \Pi_m. \end{aligned} \tag{3.4}$$

After shrinking the factorization back to size l we get the following

$$\begin{aligned} A\hat{V}_l &= \hat{U}_l \hat{\Sigma}_l \\ A^T \hat{U}_l &= \hat{V}_l \hat{\Sigma}_l + \beta_{l+1} v_{l+1} \hat{p}_l^T \end{aligned} \tag{3.5}$$

with $\hat{V}_l = (\tilde{V_m}\Pi_m)_{1:l}$, $\hat{U}_l = (\tilde{U}_m\Pi_m)_{1:l}$ and $\hat{\Sigma}_l = (\Pi_m^T\Sigma_m\Pi_m)_{1:l}$.

We have now established a factorization in which the swapping of particular subspaces can be easily implemented. Furthermore, we show that this decomposition can be reduced to the original bidiagonal factorization. Hence, we are able to increase the dimension of the search space from l to m using the original method of Golub and Kahan.

To realize the transformation of a KGK factorization to a bidiagonal factorization as given in (3.1), we reduce the residual term in (3.5) using a Householder matrix W_l, i.e. $\beta_{l+1}v_{l+1}\hat{p}_l^TW_l = \hat{\beta}_{l+1}v_{l+1}e_l^T$. Applying W_l to (3.5) yields

$$\begin{array}{rcl} A\hat{V}_lW_l & = & \hat{U}_lW_lW_l\hat{\Sigma}_lW_l^T \\ A^T\hat{U}_lW_l & = & \hat{V}_lW_lW_l\hat{\Sigma}_lW_l^T + \beta_{l+1}v_{l+1}\hat{p}_l^TW_l \end{array} \tag{3.6}$$

which can be further simplified using that $W_l^T = W_l^{-1} = W_l$, ie.

$$\begin{array}{rcl} A\breve{V}_l & = & \breve{U}_l\breve{C}_l \\ A^T\breve{U}_l & = & \breve{V}_l\breve{C}_l^T + \breve{\beta}_{l+1}v_{l+1}e_l^T \end{array} \tag{3.7}$$

with $\breve{U}_l = \hat{U}_lW_l$, $\breve{V}_l = \hat{V}_lW_l$ and $\breve{C}_l = W_l\hat{\Sigma}_lW_l^T$. The factorization given in (3.7) already looks quite similar to a valid bidiagonal factorization but unfortunately the matrix $\breve{C}_l$ is in general a dense matrix. Therefore, we need a transformation that brings $\breve{C}_l$ to bidiagonal form without destroying the residual term $\breve{\beta}_{l+1}v_{l+1}e_l^T$. This can be done in a similar way to the methods used in [29, 31, 4, 19] where a rowwise reduction of the matrix $\breve{C}_l$ is used in order to preserve the form of the residual term $\breve{\beta}_{l+1}v_{l+1}e_l^T$. Here, we propose a complete rowwise reduction to bidiagonal form, ie. $\breve{C}_l = P_lB_lQ_l^T$ with Q_l and P_l being orthogonal matrices. This process is cheap since the matrix $\breve{C}_l$ is relatively small. Substituting this into (3.7) gives

$$\begin{array}{rcl} A\breve{V}_l & = & \breve{U}_lP_lB_lQ_l^T \\ A^T\breve{U}_l & = & \breve{V}_lQ_lB_l^TP_l^T + \breve{\beta}_{l+1}v_{l+1}e_l^T. \end{array} \tag{3.8}$$

In the last step, we multiply the first part of (3.8) by Q_l and the second part by P_l which gives

$$\begin{array}{rcl} A\breve{V}_lQ_l & = & \breve{U}_lP_lB_l \\ A^T\breve{U}_lP_l & = & \breve{V}_lQ_lB_l^T + \breve{\beta}_{l+1}v_{l+1}e_l^T \end{array} \tag{3.9}$$

with $e_l^TP_l = e_l^T$ due to the special structure of P_l from the rowwise algorithm. (3.9) can easily be rewritten such that a valid bidiagonal factorization

$$\begin{array}{rcl} AV_l & = & U_lB_l \\ A^TU_l & = & V_lB_l^T + \beta_{l+1}v_{l+1}e_l^T \end{array} \tag{3.10}$$

as given in (3.1) can be obtained, see (3.10) where $\tilde{U}_l = \breve{U}_lP_l$ and $\tilde{V}_l = \breve{V}_lQ_l$. This derivation results in the following Lemma.

LEMMA 3.2. *Every bidiagonal factorization of order $k \in \mathcal{N}$*

$$\begin{array}{rcl} AV_k & = & U_kB_k \\ A^TU_k & = & V_kB_k^T + \beta_{k+1}v_{k+1}e_k^T \end{array}$$

with U_k, V_k orthogonal matrices and B_k as given in (1.1) can be transformed into a KGK factorization of the form

$$\begin{aligned} A\tilde{V}_k &= \tilde{U}_k\Sigma_k \\ A^T\tilde{U}_k &= \tilde{V}_k\Sigma_k + \beta_{k+1}v_{k+1}p_k^T. \end{aligned}$$

with $\tilde{U}_k$,$\tilde{V}_k$ orthogonal matrices and Σ_k a diagonal matrix as given in Definition 3.1. The converse relation holds as well.

In order to perform the restart and therefore extend the search space via

$$\begin{aligned} \beta_{j+1}v_{j+1} &= A^Tu_j - \alpha_jv_j \\ \alpha_{j+1}u_{j+1} &= Av_{j+1} - \beta_{j+1}u_j, \end{aligned} \tag{3.11}$$

we need a vector v_{l+1} which is given as the residual vector of (3.10) and a vector u_{l+1}. The vector u_{l+1} can be generated using the identity $\alpha_{l+1}u_{l+1} = Av_{l+1} - \beta_{l+1}u_l$ and hence enables the restart process. Note, that an expression of the form $\alpha_{l+1}u_{l+1} = Av_{l+1} - \beta_{l+1}u_l$ has to be evaluated at the end of each iteration in the method of Baglama and Reichel as well and we therefore do not create extra cost compared to their method. It has to be noticed that in comparison to the method presented in [2], the step of reducing the matrix $\breve{C}_l$ to bidiagonal form imposes extra cost. Since we are only interested in a small number of singular values l and thus a small matrix $\breve{C}_l$, the cost of the reduction to bidiagonal form does not have a significant effect on the computation times. Furthermore, we ensure that the matrix B_m is always in bidiagonal form and therefore its SVD can be computed cheaply compared to the method of Baglama and Reichel.

Algorithm 1 gives a description of how the process presented in this section can be implemented.

Algorithm 1 Krylov-Schur SVD algorithm (KSSVD)

Create bidiagonal factorization of dimension l
for $k = 1, 2, \ldots$ **do**
  Expand bidiagonal factorization from dimension l to m.
  Compute SVD of B_m.
  Transform bidiagonal to KGK factorization.
  Sort the singular values according to desired properties.
  Purge unwanted singular values.
  Lock wanted singular values.
  Shrink factorization to order l.
  Transform residual term using a Householder reflection.
  Bidiagonalize small matrix $\breve{C}_l$ and obtain bidiagonal factorization of dimension l.
end for

So far we only focused on how to compute the l largest singular values of A. The standard technique to compute a number of the smallest eigenvalues or eigenvalues around a certain value σ of a given matrix A is the shift-and-invert strategy where the Arnoldi/Lanczos process is applied to the matrix

$$(A - \sigma I)^{-1}$$

where $\sigma \in \mathbb{R}$ represents the area of interest. Note, that for the l smallest singular values σ has to be zero. This technique cannot easily be adopted for the Golub-Kahan bidiagonalization process since we would have to work with the matrix

$$(A^T A - \sigma I)^{-1}$$

which cannot be implicitly done using the Golub-Kahan bidiagonalization. For the case $\sigma = 0$ which is equivalent to computing the smallest singular values of A, a strategy that was proposed in the literature, see [2, 18], is to use the standard Golub-Kahan bidiagonalization as given in (1.1) and then use the smallest singular values of the bidiagonal matrix B_m. This technique can be easily transferred to the method proposed here where at an intermediate step the singular values are sorted using a permutation matrix according to the desired properties. In more detail, we would use a permutation Π_m to move the smallest singular values to the north-west corner of B_m and then shrink the factorization now only containing the information associated with the smallest singular values.

4. Purging and locking. The process of deflation is crucial in achieving efficient methods to compute eigenvalues or in our case singular values of the large and sparse matrix A. Once an approximation to a singular value is computed with the desired tolerance – see Section 5 for stopping criteria – we have to decide whether this singular value is of interest to us and lock it for further computations or in the case it is not of interest purge it.

To illustrate this, we assume that the following KGK decomposition is given after a certain number of iterations

$$\begin{aligned} AV_m &= U_m \Sigma_m \\ A^T U_m &= V_m \Sigma_m + \beta_{m+1} v_{m+1} p_m^T. \end{aligned}$$

with

$$\Sigma_m = \begin{bmatrix} \sigma_1 & & & & & & \\ & \sigma_2 & & & & & \\ & & \ddots & & & & \\ & & & \sigma_l & & & \\ & & & & \ddots & & \\ & & & & & \sigma_m & \end{bmatrix}.$$

Furthermore, we assume that the singular values σ_i and σ_j are converged and with σ_i to be locked and σ_j to be purged during this iteration step. First, we introduce a permutation Π such that

$$\tilde{\Sigma}_m = \Pi^T \Sigma_m \Pi = \begin{bmatrix} \sigma_i & & & & & & \\ & \sigma_j & & & & & \\ & & \sigma_1 & & & & \\ & & & \sigma_2 & & & \\ & & & & \ddots & & \\ & & & & & \sigma_m & \end{bmatrix}$$

and

$$\begin{aligned} A\tilde{V}_m &= \tilde{U}_m \tilde{\Sigma}_m \\ A^T \tilde{U}_m &= \tilde{V}_m \tilde{\Sigma}_m + \beta_{m+1} v_{m+1} \tilde{p}_m^T \end{aligned}$$

with $\tilde{U}_m = \Pi U_m$ and $\tilde{V}_m = \Pi V_m$. The permutation Π also moves the columns in U_m and V_m associated with the singular values σ_i and σ_j to the first columns of $\tilde{U}_m$ and $\tilde{V}_m$. The side-effect of locking converged singular values is that the dimension of the search space decreases which means a reduction of computing time since we can work with smaller matrices. It is easy to see that this technique can be adapted to purge and lock many converged singular values.

5. Stopping criteria. In this section we discuss possible stopping criteria that can be embedded in Algorithm 1. In Section 3 the SVD of the $m \times m$ matrix B_m is computed and we therefore know that

$$B_m q_j = \sigma_j p_j.$$

The singular value of B_m also represents an approximation to the singular values of A in the sense that

$$\begin{aligned} \left\| A^T(U_m p_j) - \sigma_j (V_m q_j) \right\| &= \|\beta_{m+1} p_{mj} v_{m+1}\| \\ &= |\beta_{m+1} p_{mj}| \end{aligned}$$

with p_{mj} being the j-th component of the vector p_m^T. This could be tested as a stopping criteria.

Another way to obtain a stopping criterion is given in the paper by Kahan, Parlett and Jiang [17] where the norm of the backward error is analyzed. We start with an approximation to the eigenvalues and eigenvectors of $A^T A$ which are associated with the singular values and vectors of A, ie.

$$A^T A V_m = V_m \Sigma_m^2 + \beta_{m+1} v_{m+1} \tilde{p}_m^T \tag{5.1}$$

with $\tilde{p}_m^T = p_m^T \Sigma_m$. When analyzing the backward error we compute the norm of a matrix E where

$$(A^T A - E_\lambda)x = \lambda x$$

such that the eigenpair (λ, x) coming from (5.1) is an exact eigenpair of a perturbed matrix. The formulae given in [17] depend on approximations to the left and right eigenvectors of $A^T A$. An approximation to a right eigenvector can be obtained from (5.1) as follows

$$A^T A v_j - \sigma_j^2 v_j = \beta_{m+1} \tilde{p}_{mj} v_{m+1}. \tag{5.2}$$

Since the matrix $A^T A$ is symmetric, (5.2) also represents an approximation to a left eigenvector, ie.

$$v_j^T A^T A - \sigma_j^2 v_j^T = \beta_{m+1} \tilde{p}_{mj} v_{m+1}^T. \tag{5.3}$$

Theorem 2' in [17] states that the norm of the backward error $E_{\sigma_j^2}$ for each eigenvalue σ_j^2 can be computed as

$$\min \left\| E_{\sigma_j^2} \right\| = \max \left\{ \|\beta_{m+1} \tilde{p}_{mj} v_{m+1}\|, \left\|\beta_{m+1} \tilde{p}_{mj} v_{m+1}^T\right\| \right\}. \tag{5.4}$$

(5.4) can be further simplified due to the nature of the left and right eigenvector residual and the result is then given by

$$\min \left\| E_{\sigma_j^2} \right\| = |\beta_{m+1} \tilde{p}_{mj}|. \tag{5.5}$$

6. Implementation details. In this section we want to address certain issues that occur when the method presented in Section 3 is implemented. An important issue in all methods based on the Lanczos process is the orthogonality of the vectors generated by this method. It is well known that a loss of orthogonality can occur when the algorithm progresses, see [24, 22]. A remedy is the so-called *reorthogonalization* where the current Lanczos vector has be orthogonalized against previously created vectors, see [24]. One can choose between a selective reorthogonalization or a full reorthogonalization against all vectors in the current Krylov subspace. In this paper we only discuss the full reorthogonalization since the restarting character of KSSVD guarantees that the number of vectors is relatively small and therefore the full reorthogonalization is not very expensive. The full reorthogonalization can be done as a classical or modified Gram-Schmidt orthogonalization, see [24] for details.

In the context of the bidiagonal factorization there are different reorthogonalization strategies due to the existence of two orthogonal sequences representing the Lanczos method, see [2, 13]. One possibility is the full one-sided reorthogonalization where only one of the sequences v_j or u_j is orthogonalized against the previous Lanczos vectors V_{j-1} or U_{j-1} respectively. The computationally more expensive way would be to do a full reorthogonalization in which both sequences v_j and u_j are orthogonalized against the previous vectors in the Krylov subspace, i.e. V_{j-1} and U_{j-1} respectively. The numerical experiments given in Section 7 only use the one-sided reorthogonalization which is also the default setup of the method implemented by Baglama and Reichel.

Reorthogonalization is also essential when using deflation techniques such as purging and locking. Due to roundoff error we have to do a reorthogonalization against all the vectors associated with already converged and therefore either purged or locked singular values because due to the finite precision arithmetic they might come into the spectrum again.

In Section 3 a number of transformations are accumulated into the two sequences u_j and v_j. For an efficient implementation of Algorithm 1 the number of updates of the sequences has to be reduced to a minimum. One possibility would be to accumulate all transformations of dimension m and apply these transformations just before the shrinking of the factorization as described in (3.5) is done. Afterwards, we accumulate all transformations of dimension l and apply the accumulated transform at the end of each iteration. A second possibility to update U_l and V_l would be to regard all transformations as transformations of dimensions m and accumulate them. With the one accumulated transform we can update the matrices U_m and V_m at the end of the iteration and shrink the factorization then.

7. Numerical Experiments. In this section we want to compare the MATLAB implementation of the method proposed in this paper with the method of Baglama and Reichel (cf. Section 2) and their MATLAB implementation `irlba.m`[1]. We also compare KSSVD to MATLAB's `svds.m` which applies routines from the ARPACK [21] to the matrix

$$\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}.$$

[1] `http://www.math.uri.edu/~jbaglama/software/irlba.m`

Specification	Interpretation	Value
`opts.k`	Number of desired singular values	`10`
`opts.m_b`	Dimension of matrix B_m or dimension of the search space	`20`
`opts.adjust`	Adjusting the number of desired eigenvalues	`3`
`opts.tol`	Tolerance for singular values	`1e-10`
`opts.v0`	Starting vector for the Lanczos bidiagonalization	`rand(M,1)`
`opts.AUG`	Augmentation via Ritz or Harmonic-Ritz vectors	`RITZ` or `HARM`

TABLE 7.1
Setup for `irlba.m`

For a fair comparison we have to start all methods with the same or very similar setup. The function `irlba.m` accepts a MATLAB structure typically called `opts` as an input in which the setup for the method can be specified, see Table 7.1. We explain some of the quantities specified in Table 7.1 in more detail now. The value of `opts.k` corresponds to number l of desired singular values. Baglama and Reichel adjust the number l using the value of `opts.adjust` which gives in the above described case a new number of desired singular values $\hat{l} = l + 3$. Note, that the algorithm stops whenever the original number l of singular values is computed to the desired accuracy. The parameter `opts.m_b` corresponds to the value we have denoted by m, the dimension of the search space. Both KSSVD and Baglama and Reichel's method use a one-sided reorthogonalization process. For MATLAB's `svds.m` we set all parameters according to Table 7.1 apart from the `opts.v0` which is of different dimension due to the formulation of the problem, `opts.adjust` which is a feature not included in the `svds.m` as is the choice of whether to use Ritz or harmonic Ritz values.

Harwell-Boeing Collection. The first example comes from the set LSQ of the Harwell-Boeing Sparse Matrix Collection [8]. In particular we look at the set LSQ which represents least squares problems in surveying. In more detail, we use the examples WELL1850 which is of dimension 1850×712 and WELL1033 which is of dimension 1033×320. We compute the 10 largest singular values of WELL1850 using the setup described in Table 7.1 for KSSVD and `irlba.m` with Ritz value augmentation. The methods are applied 5 times to each problem and we give the best and the worst number of iterations for each method with different starting vectors. The resulting singular values are given in Table 7.2 where the digits all three methods have in common are underlined. The iteration numbers for the three methods are given in Table 7.3.

In the second part, we compute the 10 largest singular values of the matrix WELL1033 with the same setup as for WELL1850. The results for the singular values are given in Table 7.4 and the best-worst iteration numbers are given in Table 7.5.

Table 7.6 shows the results for `kssvd` and `irlba` when computing the 10 smallest singular values of the matrix WELL1033 with the same setup as given above apart from the Harmonic-Ritz augmentation in `irlba`. Iteration numbers are given in Table 7.7. Computing the smallest singular values is important, for example, when solving total least squares problems, see [7].

kssvd	irbla	svds
1.794327990361091	1.794327990361083	1.794327990361090
1.738837164541720	1.738837164541731	1.738837164541731
1.718917469131029	1.718917469131041	1.718917469131042
1.682844584236183	1.682844584236179	1.682844584236186
1.645105027226848	1.645105027226849	1.645105027226848
1.643439827229122	1.643439827229131	1.643439827229122
1.630866615714931	1.630866615714942	1.630866615714934
1.624746040616118	1.624746040616118	1.624746040616121
1.601354004551848	1.601354004551845	1.601354004551850
1.600911179480472	1.600911179480467	1.600911179480464

TABLE 7.2
Results for `kssvd`*,* `irlba` *and* `svds` *for WELL1850*

	kssvd	irbla	svds
best	13	16	29
worst	14	18	32

TABLE 7.3
Iterations for `kssvd`*,* `irlba` *and* `svds` *for WELL1850*

Gene expression data. The next example is taken from [11] as part of the genomwide analysis of the host response to Malaria. Microarray analysis plays an important role in the study of understanding diseases. The amount of data associated with the gene expression analysis poses a number of computational issues since the corresponding matrices [5] can easily go up to tens or hundreds of thousand of data in a single column representing the gene features. A typical example is shown in Figure 7.1; the underlying matrix is of dimension 394×28 which means 394 gene features for the 28 samples, see [11]. The SVD is an increasingly popular tool [6, 1] when analyzing gene expression data. Here, we want to demonstrate that the method introduced in this paper can be employed to calculate a Truncated SVD which can then be used for filtering the noise in the matrix entries. Figure 7 shows rank-1 and rank-2 approximations of the gene expression matrix. We again use `kssvd`, `irlba` and `svds` to compute 10 singular values of the matrix (cf. Figure 7.8) and also the best and worst iteration numbers, see Table 7.9.

8. Conclusions. We presented a method based on the Golub-Kahan bidiagonalization that allows the efficient computation of a small number of singular values. By introducing a KGK factorization we were able to present a method that makes the implementation of deflation techniques such as purging and locking an easy task, i.e. by only using permutations on the KGK factorization.

We showed that the KGK factorization can always be converted to a Golub-Kahan bidiagonal factorization and conversely the Golub-Kahan bidiagonalization can easily be transformed into a KGK decomposition. The costs of the presented transformation are minimal due to the small number of desired singular values. In contrast to the method of Baglama and Reichel, we ensure that the bidiagonal structure is preserved.

Furthermore, we introduced stopping criteria for our method and discussed the numerical issues, such as reorthogonalization, that arise when implementing the algo-

kssvd	irbla	svds
1.806511020006566	1.806511020006559	1.806511020006566
1.767837161647911	1.767837161647911	1.767837161647911
1.728844276588977	1.728844276588977	1.728844276588989
1.586186613455318	1.586186613455318	1.586186613455318
1.567722886046058	1.567722886046059	1.567722886046066
1.538879314136535	1.538879314136525	1.538879314136536
1.474071370285059	1.474071370285055	1.474071370285058
1.443862613306116	1.443862613306119	1.443862613306125
1.432648751585027	1.432648751585035	1.432648751585031
1.430446172662784	1.430446172662785	1.430446172662780

TABLE 7.4
Results for `kssvd` *and* `irlba` *for WELL1033*

	kssvd	irbla	svds
best	17	28	57
worst	18	31	68

TABLE 7.5
Iterations for `kssvd`*,* `irlba` *and* `svds` *for WELL1033*

rithm.

We presented numerical results for a variety of problems and show that the `kssvd` method is able to outperform both `svds` and `irbla` on a number of examples.

Acknowledgment. The author would like to thank Peter Benner, Gene Golub, Jose Román and Andy Wathen for their comments and support. He would also like to thank Andy Wathen for providing the gene expression example used in Section 7.

REFERENCES

[1] O. ALTER, P. BROWN, AND D. BOTSTEIN, *Singular value decomposition for genome-wide expression data processing and modeling*, Proc Natl Acad Sci US A, 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 (electronic).
[3] P. BENNER AND H. FASSBENDER, *An implicitly restarted symplectic Lanczos method for the Hamiltonian eigenvalue problem*, Linear Algebra and its Applications, 263 (1997), pp. 75–111.
[4] P. BENNER, H. FASSBENDER, AND M. STOLL, *Solving Large-Scale Quadratic Eigenvalue Problems with Hamiltonian eigenstructure using a Structure-Preserving Krylov Subspace Method*, Technical Report 07/03, Oxford University Computing Laboratory, 2007.
[5] D. BERRAR, W. DUBITZKY, AND M. GRANZOW, *A Practical Approach to Microarray Data Analysis*, Kluwer Academic Publishers, 2003.
[6] D. BERRAR, W. DUBITZKY, AND M. GRANZOW, *Singular value decomposition and principal component analysis*, A Practical Approach to Microarray Data Analysis, (2003), pp. 91–109.
[7] A. BJORCK, *Numerical methods for least squares problems*, SIAM Philadelphia, 1996.
[8] I. S. DUFF, R. G. GRIMES, AND J. G. LEWIS, *Users' guide for the Harwell-Boeing sparse matrix collection (Release I)*, Tech. Rep. RAL 92-086, Chilton, Oxon, England, 1992.
[9] 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.
[10] G. H. GOLUB AND C. F. VAN LOAN, *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, third ed., 1996.

kssvd	irbla
0.010873862059633	0.010873862059632
0.012176552955142	0.012176552955133
0.012804926526450	0.012804926526449
0.020927626556688	0.020927626556689
0.028964680530784	0.028964680530783
0.030519788654792	0.030519788654792
0.038505437278385	0.038505437278385
0.065571527487315	0.065571527487316
0.089179744376923	0.089179744376925
0.099047944069830	0.099047944069829

TABLE 7.6
Results for `kssvd` *and* `irlba` *for WELL1033*

	kssvd	irbla
best	66	64
worst	83	81

TABLE 7.7
Iterations for `kssvd` *and* `irlba` *for WELL1033*

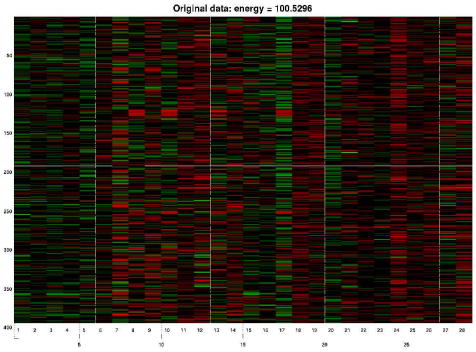

FIG. 7.1. *Original data*

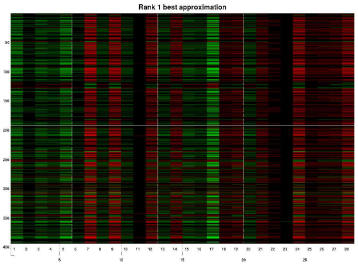

FIG. 7.2. *Rank one approximation.*

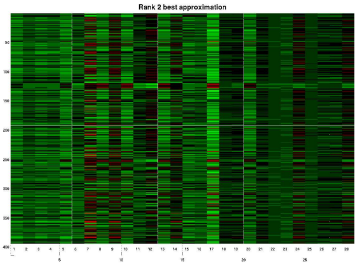

FIG. 7.3. *Rank one approximation.*

kssvd	irbla	svds
51.232909374686542	51.232909374686557	51.232909374686528
44.441215027904583	44.441215027904576	44.441215027904590
28.654180917131207	28.654180917131143	28.654180917131232
23.054650356050875	23.054650356050889	23.054650356050868
22.279173856329631	22.279173856329557	22.279173856329528
19.754193904042893	19.754193904042914	19.754193904042900
18.087482598875489	18.087482598875539	18.087482598875514
16.859822924787466	16.859822924787505	16.859822924787419
15.893070581301755	15.893070581301727	15.893070581301732
15.774495680608581	15.774495680608561	15.774495680608565

TABLE 7.8
Results for `kssvd`, `irlba` and `svds` for microarray data

	kssvd	irbla	svds
best	4	4	14
worst	4	4	17

TABLE 7.9
Iterations for `kssvd`, `irlba` and `svds` for WELL1033

[11] M. GRIFFITHS, M. SHAFI, S. POPPER, C. HEMINGWAY, M. KORTOK, A. WATHEN, K. ROCKETT, R. MOTT, M. LEVIN, C. NEWTON, K. MARSH, D. RELMAN, AND D. KWIATKOWSKI, *Genomewide analysis of the host response to malaria in Kenyan children*, J. Infect. Dis., 191 (2005), pp. 1599–1611.

[12] V. HERNÁNDEZ, J. ROMÁN, AND A. TOMÁS, *A Robust and Efficient Parallel SVD Solver Based on Restarted Lanczos Bidiagonalization*, submitted, (2007).

[13] ———, *Restarted Lanczos Bidiagonalization for the SVD in SLEPc*, tech. rep., Universidad Politecnica de Valencia, 2007.

[14] M. E. HOCHSTENBACH, *A Jacobi-Davidson type SVD method*, SIAM J. Sci. Comput., 23 (2001), pp. 606–628 (electronic). Copper Mountain Conference (2000).

[15] 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.

[16] 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 (electronic).

[17] W. KAHAN, B. PARLETT, AND E. JIANG, *Resiudal bounds on approximate eigensystems of normal matrices*, SIAM J. Numer.Anal., 19 (1982), pp. 470–484.

[18] E. KOKIOPOULOU, C. BEKAS, AND E. GALLOPOULOS, *Computing smallest singular triplets with implicitly restarted Lanczos bidiagonalization*, Appl. Numer. Math., 49 (2004), pp. 39–61.

[19] D. KRESSNER, *Numerical methods for general and structured eigenvalue problems*, vol. 46 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, Berlin, 2005.

[20] R. LEHOUCQ AND D. SORENSEN, *Deflation techniques for an implicitly restarted Arnoldi iteration*, SIAM J. Matrix Anal. Appl, 17 (1996), pp. 789–821.

[21] R. LEHOUCQ, D. SORENSEN, AND C. YANG, *Arpack users' guide: Solution of large scale eigenvalue problems with implicitly restarted arnoldi methods*, 1997.

[22] C. C. PAIGE, *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*, PhD thesis, University of London, 1971.

[23] C. C. PAIGE AND M. A. SAUNDERS, *LSQR: an algorithm for sparse linear equations and sparse least squares*, ACM Trans. Math. Software, 8 (1982), pp. 43–71.

[24] B. N. PARLETT, *The symmetric eigenvalue problem*, vol. 20 of Classics in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998. Corrected reprint of the 1980 original.

[25] H. D. SIMON AND H. ZHA, *Low-rank matrix approximation using the Lanczos bidiagonalization process with applications*, SIAM J. Sci. Comput., 21 (2000), pp. 2257–2274 (electronic).

[26] D. SORENSEN, *Deflation for implicitly restarted Arnoldi methods*, tech. rep., CAAM at Rice

University, 1998.
[27] ———, *Numerical methods for large eigenvalue problems*, Acta Numerica, (2002), pp. 519–584.
[28] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl, 13 (1992), pp. 357–385.
[29] G. Stewart, *A Krylov-Schur algorithm for large eigenproblems*, SIAM J. on Matrix Analysis and Applications, 23 (2001), pp. 601–614.
[30] ———, *Matrix Algorithms, Volume II: Eigensystems*, SIAM, Philadelphia, 2001.
[31] M. Stoll, *Locking und Purging für den Hamiltonischen Lanczos-Prozess*, 2005. Diplomarbeit.
[32] L. N. Trefethen and D. Bau, III, *Numerical linear algebra*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.
[33] K. Wu and H. Simon, *Thick-restart Lanczos method for large symmetric eigenvalue problems*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 602–616 (electronic).