Some remarks on bidiagonalization and its implementation

Post-Graduate Student:
ING. MARTIN PLEŠINGER

Faculty of Mechatronics
Technical University of Liberec
Hálkova 6
461 17 Liberec 1
martin.plesinger@tul.cz

Supervisor:
PROF. ING. ZDENĚK STRAKOŠ, DRSC.

Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
182 07 Prague 8
strakos@cs.cas.cz

Field of Study:
Computational mathematics

This work was supported by the National Program of Research "Information Society" under project 1ET400300415.

Abstract

In this contribution we are interested in orthogonal bidiagonalization of a general rectangular matrix. We describe several bidiagonalization algorithms and focuse particularly on implementation details.

The bidiagonalization is the first step in hybrid methods, which are usually used to solve ill-posed and rank-deficient problems [5] arising for instance from image deblurring. Thus we consider a matrix from ill-posed problem and illustrate the behavior of discussed bidiagonalization methods on this example.

The second example points on the very close relationship between bidiagonalization and the so called core theory in linear approximation problems [4], see also [10]. The relationship with the Lanczos tridiagonalization of symmetric positive semidefinite matrices analyzed in [7], [8] is mentioned.

1. Introduction

Consider a general rectangular matrix $A \in \mathbb{R}^{n\times m}$. The orthogonal transformations

$$P^T A Q = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \alpha_3 & \\ & & \ddots & \ddots \end{bmatrix} = B\,, \quad R^T A S = \begin{bmatrix} \beta_1 & \alpha_1 & & \\ & \beta_2 & \alpha_2 & \\ & & \beta_3 & \alpha_3 \\ & & & \ddots & \ddots \end{bmatrix} = C\,,$$

where P, Q, R, S have mutually orthonormal columns, are called the **lower** and the **upper bidiagonalization**, respectively. Obviously these transformations are closely connected, e. g., the upper bidiagonalization of A can be expressed as the lower bidiagonalization of A^T.

In the rest of this section only the lower bidiagonalization is considered, the results for the upper bidiagonalization are analogous. Define the **full bidiagonal decomposition**

$$P^T A Q = B \in \mathbb{R}^{n\times m}\,, \quad \text{where} \quad P \in \mathbb{R}^{n\times n}\,, \quad Q \in \mathbb{R}^{m\times m} \tag{1}$$

are orthogonal matrices and $P = [\,p_1\,,\,\ldots\,,\,p_n\,]$, $Q = [\,q_1\,,\,\ldots\,,\,q_m\,]$. Consequently $A = P B Q^T$. Depending on the relationship between n and m, the matrix B can contain a zero block. Furthermore, define the k-th **partial bidiagonal reduction**, $k \le \min\{n, m\}$,

$$P_k^T A Q_k = B_k \in \mathbb{R}^{k\times k}\,, \quad \text{where} \quad P_k \in \mathbb{R}^{n\times k}\,, \quad Q_k \in \mathbb{R}^{m\times k}\,, \tag{2}$$

B_k is the upper left principal submatrix of B and P_k, Q_k contains first k columns of P, Q, respectively. Thus

$$B = \left[\begin{array}{c|cc} B_k & & \\ \hline e_k^T \beta_{k+1} & \alpha_{k+1} & \\ & \beta_{k+2} & \alpha_{k+2} \\ & & \ddots \quad \ddots \end{array}\right], \quad P = [P_k\,,\, p_{k+1}\,,\,\ldots\,,\, p_n], \quad Q = [Q_k\,,\, q_{k+1}\,,\,\ldots\,,\, q_m].$$

Note that in general $A \neq P_k\, B_k\, Q_k^T$. Finally, define the $(k+)$-th partial bidiagonal reduction, $k < \min\{n,m\}$,

$$P_{k+1}^T\, A\, Q_k = \left[\frac{B_k}{e_k^T \beta_{k+1}}\right] = B_{k+} \in \mathbb{R}^{(k+1)\times k}, \quad \text{where} \quad P_{k+1} \in \mathbb{R}^{n\times(k+1)}, \quad Q_k \in \mathbb{R}^{m\times k}.$$

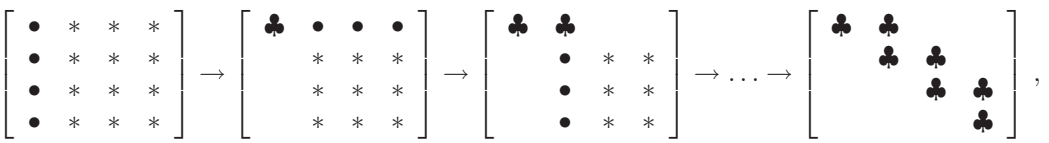
Remark 1 *Similarly, the k-th or $(k+)$-th upper partial bidiagonal reduction produces bidiagonal matrices $C_k \in \mathbb{R}^{k\times k}$, $C_{k+} \in \mathbb{R}^{k\times(k+1)}$, respectively.*

2. Two approaches to bidiagonalization

In this section two basic tools for computation of the bidiagonal transformation are summarized – the Householder method, which leads to the full bidiagonal decomposition, and the Golub-Kahan algorithm, which leads to the partial bidiagonal reduction. At the end of this section the connection between these approaches is explained.

2.1. Householder method

The first technique can also be called the **bidiagonalization using orthogonal transformations**. It is well known, that for any vector x an orthogonal matrix G can be constructed such that $Gx = e_1\|x\|_2$. Using these matrices, the matrix A can be transformed in the following way:



i. e., A is multiplied alternately from the left and right by matrices $G_1, G_2^T, G_3, G_4^T, \ldots$, where the matrices G_j are block-diagonal with two blocks – the first block is an identity matrix of growing dimension, the second block is the above described matrix G. The vectors x are marked by "•" in each step, the length of the vectors is decreasing. Denote $\bar{S}_1 \equiv I_m$,

$$\bar{R}_k \equiv G_1^T\, G_3^T\, G_5^T\, \ldots\, G_{2k-1}^T \in \mathbb{R}^{n\times n}, \quad \bar{S}_{k+1} \equiv G_2\, G_4\, G_6\, \ldots\, G_{2k} \in \mathbb{R}^{m\times m}, \tag{3}$$

the orthogonal matrices. Define the k-th and $(k+)$-th **incomplete bidiagonal decomposition**

$$\bar{R}_k^T\, A\, \bar{S}_k = \left[\frac{C_{(k-1)+}}{e_1\, e_k^T\, \beta_k} \,\middle|\, \begin{array}{c} \\ \hline A^{(k+)} \end{array}\right] = \bar{C}_k, \quad \bar{R}_k^T\, A\, \bar{S}_{k+1} = \left[\begin{array}{c|c} C_k & e_k\, e_1^T\, \alpha_k \\ \hline & A^{(k+1)} \end{array}\right] = \bar{C}_{k+}, \tag{4}$$

respectively, where $A^{(k+)}$ and $A^{(k+1)}$ are the nonbidiagonalized remains of the original matrix, see Section 3. Consequently $A = \bar{R}_k\, \bar{C}_k\, \bar{S}_k^T$, $A = \bar{R}_k\, \bar{C}_{k+}\, \bar{S}_{k+1}^T$. Matrices G may be the Householder reflection matrices, or composed Givens rotation matrices, see, e. g., [3]. Usage of the Householder reflections is discussed in details in Section 3.

The bidiagonalization using orthogonal transformations is useful if the full bidiagonal decomposition of A is required. On the other hand, if only a partial transformation is computed, which is often the case, this approach can be very ineffective. We refer to this technique as **Householder method** (or decomposition).

2.2. Golub-Kahan algorithm

The second technique is based on the **Golub-Kahan bidiagonalization** algorithm [2] (also called the Lanczos bidiagonalization, or the Golub-Kahan-Lanczos bidiagonalization). This approach is different from the previous one. Assuming that the bidiagonal elements are nonnegative, the Householder decomposition is uniquely determined. The outputs of the Golub-Kahan algorithm, i. e. the bidiagonal form of A , depend on the vector $b \in \mathbb{R}^n$. The algorithm follows:

```
00:  β1 := ‖b‖2 ;          u1 := b/β1 ;
01:  wL := A^T u1 ;
02:  α1 := ‖wL‖2 ;         v1 := wL/α1 ;
03:  wR := A v1 ;
04:  β2 := ‖wR‖2 ;         u2 := wR/β2 ;
05:  for   j = 2, 3, 4, ...
06:        wL := A^T uj − v_{j−1} βj ;
07:        αj := ‖wL‖2 ;        vj := wL/αj ;
08:        wR := A vj − uj αj ;
09:        β_{j+1} := ‖wR‖2 ;   u_{j+1} := wR/β_{j+1} ;
10:  end .
```

The algorithm is obviously stopped on the first $\alpha_j = 0$ or $\beta_j = 0$. Consider that $\alpha_1 , \ldots , \alpha_k$ and $\beta_1 , \ldots , \beta_{k+1}$ are nonzero (and thus positive). It is easy to show that $\{u_j\}_{j=1}^{k+1} \subset \mathbb{R}^n$, $\{v_j\}_{j=1}^{k} \subset \mathbb{R}^m$ are two sets of mutually orthonormal vectors. Denote

$$P_j \equiv U_j = [\, u_1 , \ldots , u_j \,] \in \mathbb{R}^{n\times j} , \quad Q_j \equiv V_j = [\, v_1 , \ldots , v_j \,] \in \mathbb{R}^{m\times j} .$$

Then

$$P_k^T A Q_k = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \end{bmatrix} = B_k , \quad P_{k+1}^T A Q_k = \begin{bmatrix} B_k \\ \hline e_k^T \beta_{k+1} \end{bmatrix} = B_{k+} .$$

The Golub-Kahan algorithm returns in k-th iteration, at the line `07` and `09`, the k-th and $(k+)$-th lower partial bidiagonal reduction of A , respectively.

This algorithm is useful if the partial bidiagonal reduction is required. We refer to this algorithm as the **Golub-Kahan algorithm**.

Remark 2 *The Golub-Kahan bidiagonalization of the matrix A starting from the vector b is very closely related to the Lanczos tridiagonalization of the matrices AA^T, and A^TA with starting vectors $b/\|b\|_2$ and $A^Tb/\|A^Tb\|_2$ respectively, see, e. g., [1]; and to the Lanczos tridiagonalization of the augmented matrix*

$$\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} , \quad \textit{with the starting vector} \quad \begin{bmatrix} b/\|b\|_2 \\ 0 \end{bmatrix} .$$

For more information about this relationship and its connection to the so called core problem, see [7], [8].

2.3. Connection between Householder and Golub-Kahan technique

Consider the k-th partial lower bidiagonalization $B_k = P_k^T A Q_k$ of A computed by the Golub-Kahan algorithm starting from the vector b. Remember that $\beta_1 = \|b\|_2$. It can be easily proven that the matrix

$$P_k^T \left[\begin{array}{c|c} b & A \end{array}\right] \left[\begin{array}{c|c} 1 & \\ \hline & Q_k \end{array}\right] = \left[\begin{array}{c|ccc} \beta_1 & \alpha_1 & & \\ & \beta_2 & \alpha_2 & \\ & & \ddots & \ddots \\ & & & \beta_k \quad \alpha_k \end{array}\right] = \left[\begin{array}{c|c} e_1\,\beta_1 & B_k \end{array}\right] \tag{5}$$

is the left upper principal submatrix of $(k+)$-th incomplete upper bidiagonal decomposition and thus the submatrix of matrix $\bar{C}_{k+}$, returned by the Householder method applied on the extended matrix $[\,b\,|\,A\,]$. Here $\bar{R}_k = [\,P_k\,,\,\bar{r}_{k+1}\,,\,\ldots\,,\,\bar{r}_n\,]$, and $\bar{S}_{k+1} = [\,[\,\mathrm{blockdiag}\,(1, Q_k)\,]\,,\,\bar{s}_{k+2}\,,\,\ldots\,,\,\bar{s}_{m+1}\,]$.

3. Implementation

In the previous section, two theoretical approaches to computation of the bidiagonalization were discussed. In this section, some implementation details of bidiagonalization algorithms are analyzed. We briefly describe these procedures, their advantages and disadvantages. The emphasis is put on influence of rounding errors, computational cost, and also computation speed and memory requirements.

3.1. Householder method [HH]

The implementation of the Householder method corresponds to the procedure described above. First, the Householder matrices are defined, then the algorithm follows.

Definition 1 *The Householder (orthogonal) matrix, which transforms the vector $x_1 \in \mathbb{R}^n$ into the vector $x_2 \in \mathbb{R}^n$, $\|x_1\|_2 = \|x_2\|_2 \neq 0$, $x_1 \neq x_2$, has the form*

$$H(x) = I - 2\,\frac{xx^T}{\|x\|_2^2} \in \mathbb{R}^{n\times n}, \quad \textit{where} \quad x = x_1 - x_2\,. \tag{6}$$

Thus $H(x)\,x_1 = x_2$, $H(x)\,x_2 = x_1$ and $H(x) = H(-x) = (H(x))^T$.

Denote $A^{(1)} \equiv A$. Denote a_1 the first column the matrix $A^{(j)}$ and $\tilde{a}_1$ the first row of the matrix $A^{(j+)}$ (matrices $A^{(j)}$ and $A^{(j+)}$ are generated during the computation). The algorithm follows:

00: generate a zero matrix $\bar{C} := 0 \in \mathbb{R}^{n\times m}$;
01: generate two identity matrices $\bar{R} := I_n$ and $\bar{S} := I_m$;
02: **for** $j = 1, 2, 3, \ldots$
03: $\beta_j := \|a_1\|_2$; $\bar{c}_{j,j} := \beta_j$;
04: $H_{2j-1} := H(\,a_1 - e_1\,\beta_j\,) \in \mathbb{R}^{(n-j+1)\times(n-j+1)}$;
05: remove a_1 from $A^{(j)}$; $A^{(j+)} := H_{2j-1}\,A^{(j)} \in \mathbb{R}^{(n-j+1)\times(m-j)}$; *// update of $A^{(\cdot)}$*
06: $\bar{R} := \bar{R}\,[\,\mathrm{blockdiag}\,(\,I_{j-1}\,,\,H_{2j-1}\,)\,]$; *// update of $\bar{R}$*
07: $\alpha_j := \|\tilde{a}_1\|_2$; $\bar{c}_{j,j+1} := \alpha_j$;
08: $H_{2j} := H(\,\tilde{a}_1^T - e_1\,\alpha_j\,) \in \mathbb{R}^{(m-j)\times(m-j)}$;
09: remove $\tilde{a}_1$ from $A^{(j+)}$; $A^{(j+1)} := A^{(j+)}\,H_{2j} \in \mathbb{R}^{(n-j)\times(m-j)}$; *// update of $A^{(\cdot)}$*
10: $\bar{S} := \bar{S}\,[\,\mathrm{blockdiag}\,(\,I_j\,,\,H_{2j}\,)\,]$; *// update of $\bar{S}$*
11: **end**.

The dimensions of matrices $A^{(\cdot)}$ are decreasing during the computation. The algorithm can be stopped for any $j = k$ at the line `06` or `10`, and the small matrix $A^{(k+)}$ or $A^{(k+1)}$, respectively, is written into the right bottom corner of the matrix $\bar{C}$ (this is the nonbidiagonalized part of the original matrix A, see (4)). Then the algorithm yields the k-th or $(k+)$-th incomplete upper bidiagonal decomposition, respectively.

The updates in the lines `05`, `06`, `09`, `10` are not realized by a matrix-matrix products (e. g., $A := A\,H$) which are computationally very expensive, see remark 3. If the knowledge of the structure (6) of the Householder matrix H is used, then the multiplication can be rewritten in the form

$$A\,H \;=\; A\left(\,I \;-\; 2\,\frac{x x^T}{\|x\|_2^2}\,\right) \;=\; A \;-\; \frac{2}{\|x\|_2^2}\,(Ax)\,x^T\,. \tag{7}$$

For detailed description, see [3, paragraphs 5.1.3 and 5.1.4].

Remark 3 *The update realized by matrix-matrix product has computational cost of order $\mathcal{O}(n^3)$ the update (7) has computational cost only of order $\mathcal{O}(n^2)$. The implementation with matrix-matrix multiplication is inapplicable, see Figure 1. Further, update by matrix-matrix product is not as stable as (7) update.*

Advantages. Stable implementation (the most stable implementation mentioned here); very quick implementation; orthogonality in the matrices $\bar{R}$, $\bar{S}$ is perfect; in each step the incomplete decomposition is obtained.

Disadvantages. Dense square matrices $\bar{R}$, $\bar{S}$ must be stored.

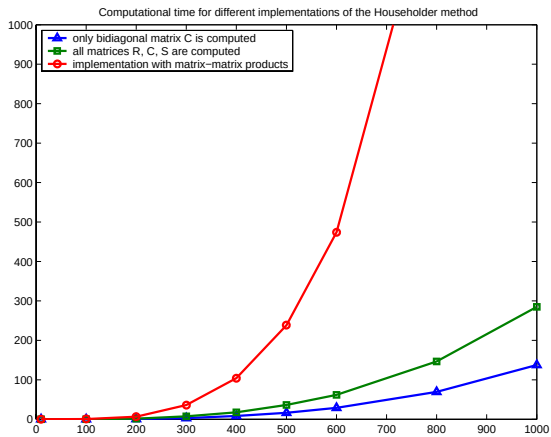


Figure 1: The horizontal axis shows the dimension of a random square matrix, the vertical axis shows the computational times for different implementations of the Householder method. Note that for the implementation with matrix-matrix products and $n = 1000$ the computational time is $t = 3337.9$ seconds.

3.2. Implementation details of Householder method

(*i*) If only the matrix $\bar{C}$ (or its bidiagonal part) has to be computed, the orthogonal matrices $\bar{R}$, $\bar{S}$ need not to be stored (and updated). Consequently, the lines `06` and `10` are leaved out and the algorithm significantly accelerates, see Figure 1.

(*ii*) If the matrix A is considerably rectangular, i. e. $n \gg m$ or $n \ll m$, it is useful to compute at first the QR or LQ decomposition of A, respectively. Assume for example that $n \gg m$ and we want to transform A into the upper bidiagonal form. First, the QR factorization

$$A \;=\; \Pi\,U\,, \quad \Pi \,\in\, \mathbb{R}^{n\times m}\,, \quad U \,\in\, \mathbb{R}^{m\times m}\,,$$

where Π has orthonormal columns and U is upper triangular is computed, then the incomplete (or full) bidiagonal decomposition $U \;=\; \bar{R}\,\bar{C}\,\bar{S}^T$ is found. Summarizing, the corresponding bidiagonal form of A

is the following

$$\bar{\bar{R}}^T A \bar{S}^T = \bar{C} \in \mathbb{R}^{m\times m}, \quad \text{where} \quad \bar{\bar{R}} = \Pi \bar{R} \in \mathbb{R}^{n\times m}, \quad \bar{S} \in \mathbb{R}^{m\times m},$$

and $A = \bar{\bar{R}} \bar{C} \bar{S}^T$. Thus this bidiagonalization is a decomposition.

This QR data preprocessing reduces the computational cost of the bidiagonalization algorithm. The matrix $\bar{\bar{R}}$ is not square and thus we have to be careful while manipulating with it. Obviously, the memory requirements are lower than in the previous algorithms. The QR factorization can be computed by the modified Gram-Schmidt algorithm with iterative refinement. Then the stability of whole procedure is held. For more information, see [3, paragraph 5.4.4].

3.3. Golub-Kahan algorithm without reorthogonalization [GK]

In the rest of this section, we study several implementations of the Golub-Kahan algorithm. All following implementations produce the partial bidiagonal reduction. In general, these algorithms do not get the decomposition of A, because they decompose only the projection of A on a Krylov subspace of growing dimension. The first simple implementation corresponds exactly to the algorithm presented in Section 2.2.

Advantages. The fastest implementation mentioned here; memory requirements are smaller than for the Householder method (if a small part of bidiagonal matrix is computed, we work only with several columns of the matrices P and Q); very simple implementation.

Disadvantages. Total lost of orthogonality in the columns of P and Q ; the bidiagonal elements are computed inexactly, this implementation is in fact inapplicable.

3.4. Golub-Kahan algorithm with reorthogonalization [GK_R]

Better results for the Golub-Kahan bidiagonalization is obtained if the currently computed vectors w_L (line `06`) and w_R (line `08`) are reorthogonalized against previous vectors v_{j-1} , v_{j-2} , $\ldots$, or u_j , u_{j-1} , $\ldots$, respectively.

Really stable implementation is obtained only if the vectors w_L and w_R are reorthogonalized against all previous vectors two times, see [9]. One time reorthogonalization or reorthogonalization against only i previous vectors is not sufficient. More than two times reorthogonalization is not necessary. It has been observed that two full reorthogonalizations is sufficient to achieve results comparable with the Householder method. We refer to this algorithm [GK_R].

Advantages. The orthogonality and stability is as good as in the Householder approach; computational time is comparable with the Householder approach; the memory requirements can be significantly smaller than in the Householder approach (especially if a small part of bidiagonal matrix is computed).

Disadvantages. Computation cost grows with iteration number.

4. Bidiagonalization of ill-posed problem

Consider the matrix

$$A = \text{SHAW}\,(100) \in \mathbb{R}^{100\times 100},$$

and the corresponding right-hand-side vector $b \in \mathbb{R}^n$, from Regularization Toolbox [6]. The system $Ax = b$ is ill-posed, singular values of A decay fast to zero [5] and A is strongly ill-conditioned. Although this matrix is nonsingular, its numerical rank (number of singular values greater than $100\,\|A\|_2\,\epsilon_M$) is equal to 20. We try to bidiagonalize this matrix using the above discussed procedures. All methods are implemented such that they return the same bidiagonal elements in exact arithmetic, i. e. the Householder procedures compute upper bidiagonalization of the extended matrix $[\,b \mid A\,]$, the Golub-Kahan procedures starts from the vector b and compute lower bidiagonalization, see Section 2.3.

Figure 2 shows the elements of the vector

$$[\,\beta_1 \;=\; \|b\|_2\,,\,\alpha_1\,,\,\beta_2\,,\,\alpha_2\,,\,\ldots\,,\,\beta_{100}\,,\,\alpha_{100}\,]\,.$$

obtained by different bidiagonalization algorithms. It shows the large difference between elements computed by [HH] and [GK]. Further, we can see four notable peaks (places of total lost of orthogonality between generated vectors) in data computed by [GK] with one time reorthogonalization against only $i = 20$ previous vectors, but only one notable peak if $i = 50$. This total lost of orthogonality disappears if [GK] with full reorthogonalization is used. It is obvious, that [GK] without full double reorthogonalization is useless here. The results of [GK_R] algorithm (with full double reorthogonalization) are not plotted here

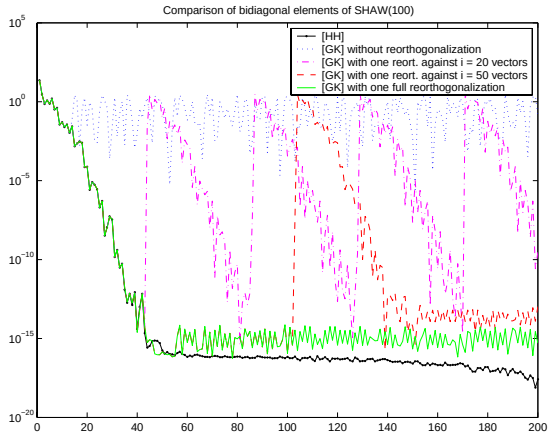


Figure 2: Bidiagonal elements of matrix SHAW(100) computed by different bidiagonalization algorithms.

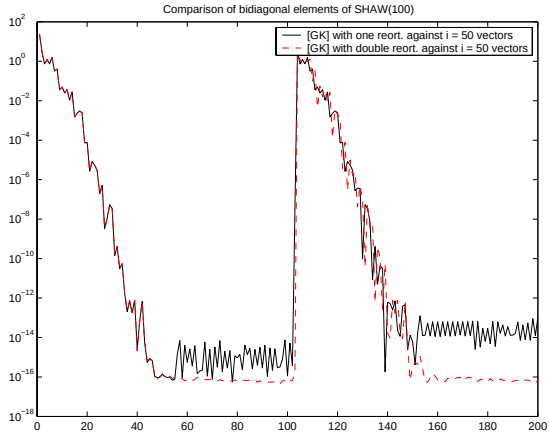


Figure 3: The double reorthogonalization effect used in [GK_R] procedure.

because they are very close to [HH]. In fact, the difference between [HH] and [GK_R] is approximately 5.9494×10^{-13} (note that the difference between [HH] and [GK] with five full reorthogonalizations is approximately 5.4101×10^{-13}). For better illustration of the influence of double reorthogonalization, see Figure 3, where the results of [GK] algorithm with one and two reorthogonalizations against $i = 50$ previous vectors is presented.

The matrix SHAW has very poor numerical properties and thus the differences between the presented algorithms are so large. For matrices having better numerical properties, [GK] with one full reorthogonalization and sometimes also with non-full reorthogonalization can be used. Then the differences between computed bidiagonal forms may not be so significant.

5. The core problem revealing test

Consider an orthogonally invariant linear approximation problem $Ax \approx b$. In [4], it was proved that the upper bidiagonalization of the matrix $[\,b\,|\,A\,]$ determines the so called core approximation problem that contains all necessary and sufficient information for solving the original problem. If the bidiagonalization is stopped at the first $\beta_{q+1} = 0$ or $\alpha_{q+1} = 0$, then it determines the core problem $B_q\, y = \beta_1\, e_1$ (compatible) or $B_{q+}\, y \approx \beta_1\, e_1$ (incompatible), respectively, see (5). In this section, we consider linear systems which contain the compatible core problem of known dimension q and thus we expect that $\beta_{q+1} = 0$. All the following testing problems are solved by [HH].

5.1. Testing problem A

Consider the matrix and the right-hand-side

$$A = \Pi_1 \left[\begin{array}{c|c} \mathrm{diag}\,(\sigma_1,\ldots,\sigma_q) & 0 \\ \hline 0 & \texttt{rand(n-q)} \end{array}\right] \Pi_2^T \in \mathbb{R}^{n\times n}, \quad b = \Pi_1 \left[\begin{array}{c} \texttt{rand(q,1)} \\ \hline 0 \end{array}\right] \in \mathbb{R}^n,$$

where Π_1, Π_2 are "random" orthogonal matrices computed by `qr(rand(n))` command (see MATLAB manual [11] for description of `qr()`, `rand()`).

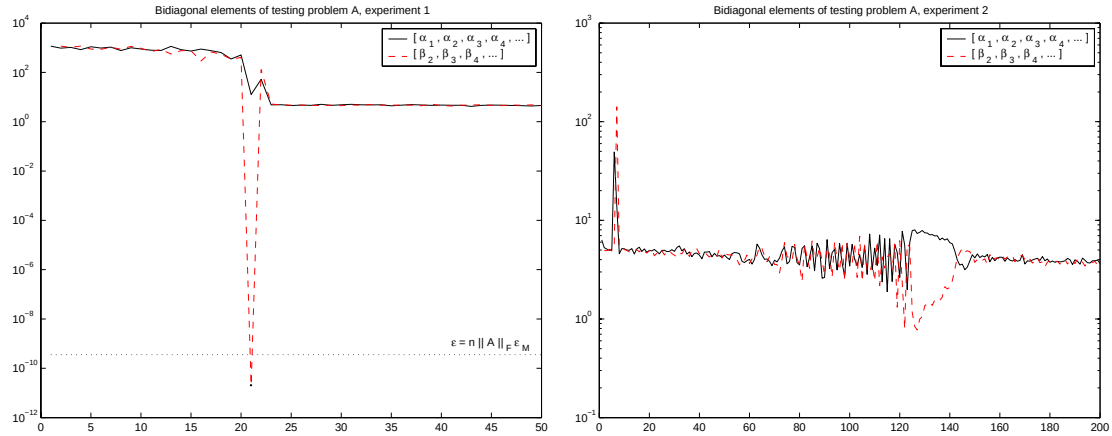


Figure 4: Testing problem A, experiment 1 (*left*): Core problem of dimension $q = 20$ is revealed, $\beta_{21} \approx 10^{-11}$. Experiment 2 (*right*): Core problem of dimension $q = 100$ is not revealed.

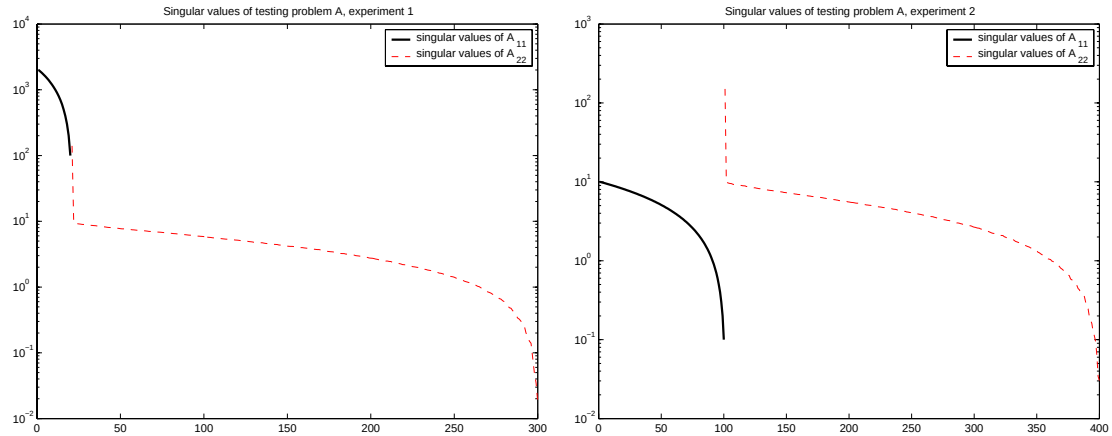


Figure 5: Testing problem A, experiments 1 (*left*) and 2 (*right*): Singular values of $A_{11} = \mathrm{diag}\,(\sigma_1,\ldots,\sigma_q)$ followed by singular values of $A_{22} = \texttt{rand(n-q)}$.

In the first experiment $n = 300$, $q = 20$, $(\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{20}) = (2000, 1900, 1800, \ldots, 100)$. Bidiagonal elements obtained for this example are plotted in the left graph in Figure 4. In the second experiment $n = 400$, $q = 100$, $(\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{100}) = (10, 9.9, 9.8, \ldots, 0.1)$. Obtained results are plotted in the right graph in Figure 4. In the first case, the core problem reveals, β_{21} is evidently smaller than the others bidiagonal elements. In the second case, the results are not satisfactory.

In fact, the wrong results obtained for Experiment 2 are the consequence of an inappropriate choice of testing problem. The random matrix $A_{22} =$ `rand(n-q)` has one very large singular value, approximately $\sigma_{\max}(A_{22}) \approx 0.5\,(n-q)$ and the remaining singular values decay linearly, see Figure 5. If $\sigma_{\max}(A_{22}) \gg \sigma_1$, by influence of rounding errors, the dominant singular value of A_{22} is quickly absorbed into the bidiagonal submatrix. This is the consequence of large sensitivity of this problem, despite the stability of computation is held (this will be obvious from the following testing problem).

Remark 4 *We hope, that it will be possible to explain this effect through the connection between the Golub-Kahan bidiagonalization and the Lanczos tridiagonalization, using the properties of Jacobi matrices, see [7], [8].*

5.2. Testing problem B

Consider the matrix and the right-hand-side

$$A = \Pi_1 \begin{bmatrix} \alpha_1 & & \\ \beta_2 & \alpha_2 & \\ & \ddots & \ddots \end{bmatrix} \Pi_2^T \in \mathbb{R}^{n\times m}, \quad b = \Pi_1 \begin{bmatrix} \beta_1 = \texttt{20*rand} \\ 0 \\ \vdots \end{bmatrix} \in \mathbb{R}^n .$$

For $m < n$, sequences $\{\alpha_j\}_{j=1}^{m}$, $\{\beta_j\}_{j=2}^{m+1}$ are generated by the command

```
-10*sort(-rand(m,1)) + rand(m,1);
```

and similarly for $m \geq n$ (see MATLAB manual [11] for description of `sort()`). Both these sequences contain random numbers from the interval $(0, 10)$ approximately sorted from the largest to the smallest number (all numbers are positive). Matrices Π_1, Π_2 are generated in the same way as in the testing problem A. Then we put $\beta_{q+1} = 0$. Summarizing, the n by m rectangular system with compatible core problem of dimension q is created. Moreover, the bidiagonal form of the core problem is known beforehand.

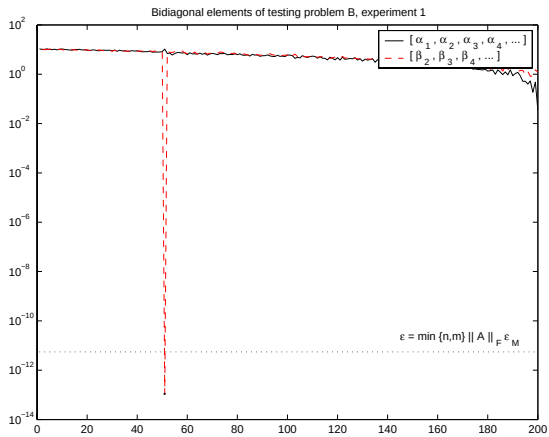


Figure 6: Testing problem B, experiment 1: Core problem is revealed, $\beta_{51} \approx 10^{-13}$.

In the first experiment $n = 1000$, $m = 200$, $q = 50$, in the second experiment $n = m = 1000$, $q = 50$. The core problem reveals without any complication in both cases, for results obtained in the first experiment

see Figure 6. Because the bidiagonal form of these problems was predetermined, the absolute error of the whole process (orthogonal transformation by Π_1, Π_2 and the following bidiagonalization) can be computed. Denote $B_j \in \mathbb{R}^{j\times j}$ the upper left principal part of the original bidiagonal matrix and $\tilde{B}_j$ the same block of the computed matrix, then

$$\|B_q - \tilde{B}_q\|_2 \doteq 8.704253 \times 10^{-14}, \quad \text{in the first experiment and}$$
$$\|B_q - \tilde{B}_q\|_2 \doteq 5.908292 \times 10^{-14}, \quad \text{in the second experiment,} \quad q = 50\,.$$

The bigger absolute error in the first (smaller) problem is probably caused by the inner QR factorization.

Remark 5 *The absolute error must be computed only from the core problem part of bidiagonal matrix. In finite arithmetic $\tilde{\beta}_{q+1} \neq 0$, but it is very close to zero. The computation does not stop and data obtained during the following computation are strongly influenced by errors.*

6. Calling procedures in MATLAB

All presented procedures were implemented in MATLAB such that they return (theoretically) the same bidiagonal form.

6.1. Householder-based procedures

The [HH] algorithm is called by the commands

```
[P,B,Q] = bidiag_hholder(A,b,k,p,tol,qrf);
      B = bidiag_hholder(A,b,k,p,tol,qrf);
```

where A is a rectangular matrix, b is a vector (e. g., right-hand-side of the problem $Ax = b$). The procedure computes the upper incomplete decomposition of $[\,b \,|\, A\,]$ (and returns the lower decomposition of A). Here k is the number of iterations and p equals 0 or 1. If $p = 0$, then the algorithm produces a square bidiagonal matrix of dimension $k \times k$, if $p = 1$, then it produces a rectangular matrix of dimension $(k + 1) \times k$.

Default values for k and p are $k = \min\{n, m\}$, $p = 0$ if $n \leq m$, or $p = 1$ if $n > m$. With this default values the full decomposition is computed. If only k is prescribed then the default value is $p = 0$ and the square bidiagonal form is returned.

The procedure notifies on each bidiagonal element smaller than `tol`, default value is $2^{-52} \doteq 2.22 \times 10^{-16}$. By the parameter `qrf` we can forbid or enforce the QR factorization of $[\,b \,|\, A\,]$, or LQ factorization of A, default (and recommended) is automatic detection. The decision is based on the dimensions of A, and on the demand to compute only the bidiagonal matrix B, or all matrices P, B, Q.

6.2. Golub-Kahan-based procedures

The Golub-Kahan-based methods [GK], [GK_R] are called by the command

```
[P,B,Q] = bidiag_lgk(A,b,k,p,tol,i,t);
```

This procedure computes a lower partial bidiagonal reduction of A starting from the vector b. The parameters k, p and `tol` and their default values are same as in `bidiag_hholder()`. This procedure returns also the k-th or $(k+)$-th lower partial bidiagonal reduction of A where $p = 0$ or $p = 1$, respectively.

The parameter i specifies the number of vectors against which we want to reorthogonalize, the parameter t specifies how many times the reorthogonalization is performed. The choice

$i = 0$	or	$t = 0$	corresponds [GK] without reorthogonalization,
$i > 0$	and	$t = 1$	corresponds [GK] with reort. against i previous vectors,

$i = -1$ and $t = 1$ corresponds [GK] with full reorthogonalization,
$i = -1$ and $t = 2$ corresponds [GK_R] = [GK] with full double reort. (default values).

Another combinations are possible too, but they are not so interesting, e. g., choice

$i = 50$ and $t = 2$ corresponds the method presented in Figure 3,
$i = -1$ and $t = 5$ corresponds the method with five times full reorthogonalization.

All numerical examples were carried out on computer Hewlett-Packard Compaq nx9110, Intel Pentium 4 CPU, 2.80 GHz, 448 MB RAM, in MATLAB 6.5 release 13.

References

[1] Å. Björck, "A bidiagonalization algorithm for solving large and sparse ill-posed systems of linear equations", *BIT*, 28, pp. 659–670, 1988.

[2] G. H. Golub, W. Kahan, "Calculating the singular values and pseudo-inverse of a matrix", *SIAM J. Numer. Anal.*, Ser. B 2, pp. 205–224, 1965.

[3] G. H. Golub, C. F. Van Loan, "Matrix Computations", 3rd edition, *The John Hopkins University Press, Baltimore*, 1996.

[4] C. C. Paige, Z. Strakoš, "Core problems in linear algebraic systems", *SIAM J. Matrix Anal. Appl.*, 27, pp. 861–875, 2006.

[5] P. C. Hansen, "Rank-Deficient and Discrete Ill-Posed Problems", *SIAM, Philadelphia*, 1998.

[6] P. C. Hansen, "Regularization Tools – version 3.2 for MALTAB 6.0", a package for analysis and solution of discrete ill-posed problems, (`http://www2.imm.dtu.dk/~pch/Regutools/index.html`).

[7] I. Hnětynková, Z. Strakoš, "Lanczos tridiagonalization and core problems", to appear in *Lin. Alg Appl*.

[8] I. Hnětynková, M. Plešinger, Z. Strakoš, "Lanczos tridiagonalization, Golub-Kahan bidiagonalization and core problem", to appear in *PAMM*, vol. 6.

[9] G. Meurant, Z. Strakoš, "The Lanczos and conjugate gradients algorithms in finite precision arithmetic", Acta Numerica, pp. 1–70, 2006.

[10] M. Plešinger, Z. Strakoš, "Core problém v úlohách nejmenších čtverců", *In Proceedings of the Doktorandský den ÚI AV ČR*, pp. 102–108, 2005, (`http://www.cs.cas.cz/hakl/doktorandsky-den/files/2005/dk05proc.pdf`).

[11] The MathWorks, Inc. "MATLAB: Documentation", (`http://www.mathworks.com/access/helpdesk/help`).