Chapter 9

Arnoldi and Lanczos algorithms

9.1 An orthonormal basis for the Krylov space $\mathcal{K}^j(\mathbf{x})$

The natural basis of the Krylov subspace $\mathcal{K}^j(\mathbf{x}) = \mathcal{K}^j(\mathbf{x}, A)$ is evidently $\{\mathbf{x}, A\mathbf{x}, \ldots, A^{j-1}\mathbf{x}\}$. Remember that the vectors $A^k\mathbf{x}$ converge to the direction of the eigenvector corresponding to the largest eigenvalue (in modulus) of A. Thus, this basis tends to be badly conditioned with increasing dimension j. Therefore, the straightforward procedure, the **Gram–Schmidt orthogonalization process**, is applied to the basis vectors in their natural order.

Suppose that $\{\mathbf{q}_1, \ldots, \mathbf{q}_i\}$ is the orthonormal basis for $\mathcal{K}^i(\mathbf{x})$, where $i \leq j$. We construct the vector $\mathbf{q}_{j+1}$ by first orthogonalizing $A^j\mathbf{x}$ against $\mathbf{q}_1, \ldots, \mathbf{q}_j$,

$$\mathbf{y}_j := A^j\mathbf{x} - \sum_{i=1}^{j} \mathbf{q}_i\mathbf{q}_i^* A^j\mathbf{x}, \tag{9.1}$$

and then normalizing the resulting vector,

$$\mathbf{q}_{j+1} = \mathbf{y}_j/\|\mathbf{y}_j\|. \tag{9.2}$$

Then $\{\mathbf{q}_1, \ldots, \mathbf{q}_{j+1}\}$ is an orthonormal basis of $\mathcal{K}^{j+1}(\mathbf{x})$, called in general the **Arnoldi basis** or, if the matrix A is real symmetric or Hermitian, the **Lanczos basis**. The vectors $\mathbf{q}_i$ are called **Arnoldi vectors** or **Lanczos vectors**, respectively, see [6, 1].

The vector $\mathbf{q}_{j+1}$ can be computed in a more economical way since

$$\begin{aligned}
\mathcal{K}^{j+1}(\mathbf{x}, A) &= \mathcal{R}\left([\mathbf{x}, A\mathbf{x}, \ldots, A^j\mathbf{x}]\right), && (\mathbf{q}_1 = \mathbf{x}/\|\mathbf{x}\|), \\
&= \mathcal{R}\left([\mathbf{q}_1, A\mathbf{q}_1, \ldots, A^j\mathbf{q}_1]\right) && (A\mathbf{q}_1 = \alpha\mathbf{q}_1 + \beta\mathbf{q}_2,\ \beta \neq 0), \\
&= \mathcal{R}\left([\mathbf{q}_1, \alpha\mathbf{q}_1 + \beta\mathbf{q}_2, A(\alpha\mathbf{q}_1 + \beta\mathbf{q}_2), \ldots, A^{j-1}(\alpha\mathbf{q}_1 + \beta\mathbf{q}_2)]\right), \\
&= \mathcal{R}\left([\mathbf{q}_1, \mathbf{q}_2, A\mathbf{q}_2, \ldots, A^{j-1}\mathbf{q}_2]\right), \\
&\ \ \vdots \\
&= \mathcal{R}\left([\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_{j-1}, A\mathbf{q}_j]\right).
\end{aligned}$$

So, instead of orthogonalizing $A^j\mathbf{q}_1$ against $\mathbf{q}_1, \ldots, \mathbf{q}_j$, we can orthogonalize $A\mathbf{q}_j$ against $\mathbf{q}_1, \ldots, \mathbf{q}_j$ to obtain $\mathbf{q}_{j+1}$. The component $\mathbf{r}_j$ of $A\mathbf{q}_j$ orthogonal to $\mathbf{q}_1, \ldots, \mathbf{q}_j$ is given by

$$\mathbf{r}_j = A\mathbf{q}_j - \sum_{i=1}^{j} \mathbf{q}_i(\mathbf{q}_i{}^* A\mathbf{q}_j). \tag{9.3}$$

If $\mathbf{r}_j = \mathbf{0}$ then the procedure stops which means that we have found an invariant subspace, namely $\text{span}\{\mathbf{q}_1, \ldots, \mathbf{q}_j\}$. If $\|\mathbf{r}_j\| > 0$ we obtain $\mathbf{q}_{j+1}$ by normalizing,

$$\mathbf{q}_{j+1} = \frac{\mathbf{r}_j}{\|\mathbf{r}_j\|}. \tag{9.4}$$

Since, $\mathbf{q}_{j+1}$ and $\mathbf{r}_j$ are aligned, we have

$$\mathbf{q}_{j+1}^* \mathbf{r}_j = \|\mathbf{r}_j\| \overset{(9.3)}{=} \mathbf{q}_{j+1}^* A \mathbf{q}_j. \tag{9.5}$$

The last equation holds since $\mathbf{q}_{j+1}$ (by construction) is orthogonal to all the previous Arnoldi vectors. Let

$$h_{ij} = \mathbf{q}_i^* A \mathbf{q}_j.$$

Then, (9.3)–(9.5) can be written as

$$A\mathbf{q}_j = \sum_{i=1}^{j+1} \mathbf{q}_i h_{ij}. \tag{9.6}$$

We collect the procedure in Algorithm 9.1

Algorithm 9.1 The Arnoldi algorithm for the computation of an orthonormal basis of a Krylov space

1: Let $A \in \mathbb{F}^{n\times n}$. This algorithm computes an orthonormal basis for $\mathcal{K}^k(\mathbf{x})$.
2: $\mathbf{q}_1 = \mathbf{x}/\|\mathbf{x}\|_2$;
3: **for** $j = 1, \ldots, k$ **do**
4: $\quad \mathbf{r} := A\mathbf{q}_j$;
5: $\quad$ **for** $i = 1, \ldots, j$ **do** /* Gram-Schmidt orthogonalization */
6: $\quad\quad h_{ij} := \mathbf{q}_i^* \mathbf{r}, \quad \mathbf{r} := \mathbf{r} - \mathbf{q}_i h_{ij}$;
7: $\quad$ **end for**
8: $\quad h_{j+1,j} := \|\mathbf{r}\|$;
9: $\quad$ **if** $h_{j+1,j} = 0$ **then** /* Found an invariant subspace */
10: $\quad\quad$ **return** $(\mathbf{q}_1, \ldots, \mathbf{q}_j, H \in \mathbb{F}^{j\times j})$
11: $\quad$ **end if**
12: $\quad \mathbf{q}_{j+1} = \mathbf{r}/h_{j+1,j}$;
13: **end for**
14: **return** $(\mathbf{q}_1, \ldots, \mathbf{q}_{k+1}, H \in \mathbb{F}^{k+1\times k})$

The Arnoldi algorithm returns if $h_{j+1,j} = 0$, in which case j is the degree of the minimal polynomial of A relative to $\mathbf{x}$, cf. (8.5). This algorithm costs k matrix-vector multiplications, $n^2/2 + \mathcal{O}(n)$ inner products, and the same number of _axpy's.

Defining $Q_k = [\mathbf{q}_1, \ldots, \mathbf{q}_k]$, equation (9.6) can be collected for $j = 1, \ldots, k$,

$$AQ_k = Q_k H_k + [\underbrace{\mathbf{0}, \ldots, \mathbf{0}}_{k-1 \text{ times}}, \mathbf{q}_{k+1} h_{k+1,k}] \tag{9.7}$$

Equation (9.7) is called **Arnoldi relation**. The construction of the Arnoldi vectors is expensive. Most of all, each iteration step becomes more costly as the number of vectors against which $\mathbf{r}$ has to be orthogonalized increases. Therefore, algorithms based on the Arnoldi relation like GMRES or the Arnoldi algorithm itself are restarted. This in general means that the algorithm is repeated with a initial vector that is extracted from previous invocation of the algorithm.

9.2 Arnoldi algorithm with explicit restarts

Algorithm 9.1 stops if $h_{m+1,m} = 0$, i.e., if it has found an invariant subspace. The vectors $\{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ then form an invariant subspace of A,

$$AQ_m = Q_m H_m, \qquad Q_m = [\mathbf{q}_1, \ldots, \mathbf{q}_m].$$

The eigenvalues of H_m are eigenvalues of A as well and the Ritz vectors are eigenvectors of A.

In general, we cannot afford to store the vectors $\mathbf{q}_1, \ldots, \mathbf{q}_m$ because of limited memory space. Furthermore, the algorithmic complexity increases linearly in the iteration number j. The orthogonalization would cost $2nm^2$ floating point operations.

Often it is possible to extract good approximate eigenvectors from a Krylov space of small dimension. We have seen, that in particular the extremal eigenvalues and corresponding eigenvectors are very well approximated after a few iteration steps. So, if only a small number of eigenpairs is desired, it is usually sufficient to get away with Krylov space of much smaller dimension than m.

Exploiting the Arnoldi relation (9.7) we can get cheap estimates for the eigenvalue/eigenvector residuals. Let $\mathbf{u}_i^{(k)} = Q_k \mathbf{s}_i^{(k)}$ be a Ritz vector with Ritz value $\vartheta_i^{(k)}$. Then

$$A\mathbf{u}_i^{(k)} - \vartheta_i^{(k)}\mathbf{u}_i^{(k)} = AQ_k\mathbf{s}_i^{(k)} - \vartheta_i^{(k)}Q_k\mathbf{s}_i^{(k)} = (AQ_k - Q_kH_k)\mathbf{s}_i^{(k)} = h_{k+1,k}\mathbf{q}_{k+1}\mathbf{e}_k^*\mathbf{s}_i^{(k)}.$$

Therefore,

$$\|(A - \vartheta_i^{(k)}I)\mathbf{u}_i^{(k)}\|_2 = h_{k+1,k}|\mathbf{e}_k^*\mathbf{s}_i^{(k)}|. \tag{9.8}$$

The residual norm is equal to the last component of $\mathbf{s}_i^{(k)}$ multiplied by $h_{k+1,k}$ (which is positive by construction). These residual norms are not always indicative of actual errors in $\lambda_i^{(k)}$, but they can be helpful in deriving stopping procedures.

We now consider an algorithm for computing some of the extremal eigenvalues of a non-Hermitian matrix. The algorithm proceeds by computing one eigenvector or rather Schur vector at the time. For each of them an individual Arnoldi procedure is employed. Let us assume that we have already computed $k-1$ Schur vectors $\mathbf{u}_1, \ldots \mathbf{u}_{k-1}$. To compute $\mathbf{u}_k$ we force the iterates in the Arnoldi process (the Arnoldi vectors) to be orthogonal to U_{k-1} where $U_{k-1} = [\mathbf{u}_1, \ldots \mathbf{u}_{k-1}]$. So, we work essentially with the matrix

$$(I - U_{k-1}U_{k-1}^*)A$$

that has $k-1$ eigenvalues zero which we of course neglect.

The procedure is given in Algorithm 9.2. The Schur vectors $\mathbf{u}_1, \ldots \mathbf{u}_{k-1}$ are kept in the search space, while the Krylov space is formed with the next approximate Schur vector. The search space thus is

$$\text{span}\{\mathbf{u}_1, \ldots \mathbf{u}_{k-1}, \mathbf{u}_k, A\mathbf{u}_k, \ldots A^{m-k}\mathbf{u}_k\}.$$

In Algorithm 9.2 the basis vectors are denoted $\mathbf{v}_j$ with $\mathbf{v}_j = \mathbf{u}_j$ for $j < k$. The vectors $\mathbf{v}_k, \ldots, \mathbf{v}_m$ form an orthonormal basis of $\text{span}\{\mathbf{u}_k, A\mathbf{u}_k, \ldots A^{m-k}\mathbf{u}_k\}$.

The matrix H_m for $k = 2$ has the structure

$$H_m = \left[\begin{array}{cc|cccc} \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ \hline & & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \end{array}\right]$$

Algorithm 9.2 Explicitly restarted Arnoldi algorithm

Let $A \in \mathbb{F}^{n\times n}$. This algorithm computes the n_{ev} largest eigenvalues of A together with the corresponding Schur vectors.
Set $k = 1$.
loop
  for $j = k, \ldots, m$ **do** /* Execute $m - k$ steps of Arnoldi */
    $\mathbf{r} := A\mathbf{q}_j$;
    for $i = 1, \ldots, j$ **do**
      $h_{ij} := \mathbf{q}_i^*\mathbf{r}, \quad \mathbf{r} := \mathbf{r} - \mathbf{q}_i h_{ij}$;
    end for
    $h_{j+1,j} := \|\mathbf{r}\|$;
    $\mathbf{q}_{j+1} = \mathbf{r}/h_{j+1,j}$;
  end for
  Compute approximate eigenvector of A associated with λ_k and the corresponding residual norm estimate ρ_k according to (9.8).
  Orthogonalize this eigenvector (Ritz vector) against all previous $\mathbf{v}_j$ to get the approximate Schur vector $\mathbf{u}_k$. Set $\mathbf{v}_k := \mathbf{u}_k$.
  if ρ_k is small enough **then** /* accept eigenvalue */
    for $i = 1, \ldots, k$ **do**
      $h_{ik} := \mathbf{v}_i^* A\mathbf{v}_k$;
    end for
    Set $k := k + 1$.
    if $k \geq n_{\text{ev}}$ **then**
      return $(\mathbf{v}_1, \ldots, \mathbf{v}_k, H \in \mathbb{F}^{k\times k})$
    end if
  end if
end loop

where the block in the lower right corresponds to the Arnoldi process for the Krylov space $\mathcal{K}_{m-k}(\mathbf{u}_k, (I - U_{k-1}U_{k-1}^*)A)$.

This algorithm needs at most m basis vectors. As soon as the dimension of the search space reaches m the Arnoldi iteration is **restarted** with the best approximation as the initial vector. The Schur vectors that have already converged are **locked** or **deflated**.

9.3 The Lanczos basis

We have seen that the Lanczos basis is formally constructed in the same way as the Arnoldi basis, however with a Hermitian matrix. It deserves a special name for the simplifications that the symmetry entails.

By multiplying (9.7) with Q_k^* from the left we get

$$Q_k^* A Q_k = Q_k^* Q_k H_k = H_k. \tag{9.9}$$

If A is Hermitian, then so is H_k. This means that H_k is *tridiagonal*. To emphasize this matrix structure, we call this tridiagonal matrix T_k. Due to symmetry, equation (9.3) simplifies considerably,

$$\mathbf{r}_j = A\mathbf{q}_j - \mathbf{q}_i \underbrace{(\mathbf{q}_j^* A\mathbf{q}_j)}_{\alpha_j\in\mathbb{R}} - \mathbf{q}_{j-1} \underbrace{(\mathbf{q}_{j-1}^* A\mathbf{q}_j)}_{\beta_{j-1}\in\mathbb{F}} = A\mathbf{q}_j - \alpha_j\mathbf{q}_j - \beta_{j-1}\mathbf{q}_{j-1}. \tag{9.10}$$

Similarly as earlier, we premultiply (9.10) by $\mathbf{q}_{j+1}$ to get

$$\begin{aligned}\|\mathbf{r}_j\| = \mathbf{q}_{j+1}^*\mathbf{r}_j &= \mathbf{q}_{j+1}^*(A\mathbf{q}_j - \alpha_j\mathbf{q}_j - \beta_{j-1}\mathbf{q}_{j-1})\\ &= \mathbf{q}_{j+1}^*A\mathbf{q}_j = \bar{\beta}_j.\end{aligned}$$

From this it follows that $\beta_j \in \mathbb{R}$. Therefore,

$$\beta_j\mathbf{q}_{j+1} = \mathbf{r}_j, \qquad \beta_j = \|\mathbf{r}_j\|. \tag{9.11}$$

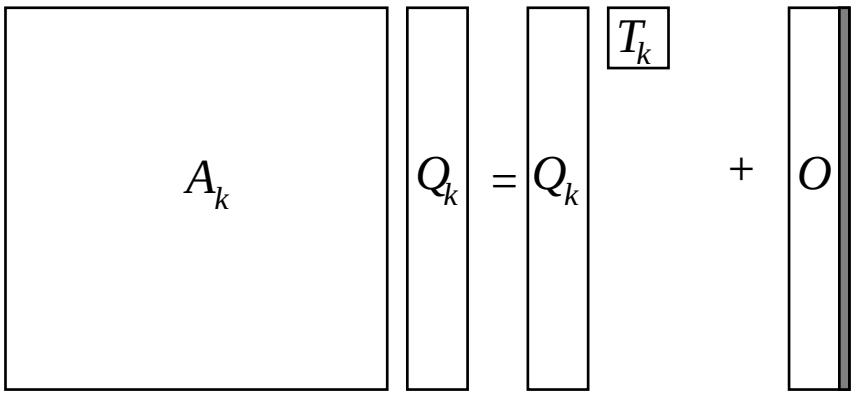
Collecting (9.10)–(9.11) yields

$$A\mathbf{q}_j = \beta_{j-1}\mathbf{q}_{j-1} + \alpha_j\mathbf{q}_j + \beta_j\mathbf{q}_{j+1}. \tag{9.12}$$

Gathering these equations for $j = 1, \ldots, k$ we get

$$AQ_k = Q_k \underbrace{\begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \beta_{k-1} \\ & & & \beta_{k-1} & \alpha_k \end{pmatrix}}_{\mathbf{T}_k} + \beta_k[\mathbf{0}, \ldots, \mathbf{0}, \mathbf{q}_{k+1}]. \tag{9.13}$$

$T_k \in \mathbb{R}^{k\times k}$ is *real symmetric*. Equation (9.13) is called **Lanczos relation**. Pictorially, this is



The Lanczos algorithm is summarized in Algorithm 9.3. In this algorithm just the three vectors $\mathbf{q}$, $\mathbf{r}$, and $\mathbf{v}$ are employed. In the j-th iteration step (line 8) $\mathbf{q}$ is assigned $\mathbf{q}_j$ and $\mathbf{v}$ stores $\mathbf{q}_{j-1}$. $\mathbf{r}$ stores first (line 9) $A\mathbf{q}_j - \beta_{j-1}\mathbf{q}_{j-1}$. Later (step 11), when α_j is available, it stores $\mathbf{r}_j = A\mathbf{q}_j - \beta_{j-1}\mathbf{q}_{j-1} - \alpha_j\mathbf{q}_j$. In the computation of α_j the fact is exploited that $\mathbf{q}_j^*\mathbf{q}_{j-1} = 0$ whence

$$\alpha_j = \mathbf{q}_j^*A\mathbf{q}_j = \mathbf{q}_j^*(A\mathbf{q}_j - \beta_{j-1}\mathbf{q}_{j-1}).$$

In each traversal of the j-loop a column is appended to the matrix Q_{j-1} to become Q_j. If the Lanczos vectors are not desired this statement can be omitted. The Lanczos vectors are required to compute the eigenvectors of A. Algorithm 9.3 returns when $j = m$, where m is the degree of the minimal polynomial of A relative to $\mathbf{x}$. $b_m = 0$ implies

$$AQ_m = Q_mT_m. \tag{9.14}$$

Algorithm 9.3 Basic Lanczos algorithm for the computation of an orthonormal basis for of the Krylov space $\mathcal{K}^m(\mathbf{x})$

Let $A \in \mathbb{F}^{n\times n}$ be Hermitian. This algorithm computes the Lanczos relation (9.13), i.e., an orthonormal basis $Q_m = [\mathbf{q}_1, \ldots, \mathbf{q}_m]$ for $\mathcal{K}^m(\mathbf{x})$ where m is the smallest index such that $\mathcal{K}^m(\mathbf{x}) = \mathcal{K}^{m+1}(\mathbf{x})$, and (the nontrivial elements of) the tridiagonal matrix T_m.
$\mathbf{q} := \mathbf{x}/\|\mathbf{x}\|; \quad Q_1 = [\mathbf{q}];$
$\mathbf{r} := A\mathbf{q};$
$\alpha_1 := \mathbf{q}^*\mathbf{r};$
$\mathbf{r} := \mathbf{r} - \alpha_1\mathbf{q};$
$\beta_1 := \|\mathbf{r}\|;$
for $j = 2, 3, \ldots$ **do**
  $\mathbf{v} = \mathbf{q}; \quad \mathbf{q} := \mathbf{r}/\beta_{j-1}; \quad Q_j := [Q_{j-1}, \mathbf{q}];$
  $\mathbf{r} := A\mathbf{q} - \beta_{j-1}\mathbf{v};$
  $\alpha_j := \mathbf{q}^*\mathbf{r};$
  $\mathbf{r} := \mathbf{r} - \alpha_j\mathbf{q};$
  $\beta_j := \|\mathbf{r}\|;$
  if $\beta_j = 0$ **then**
    return $(Q \in \mathbb{F}^{n\times j};\ \alpha_1, \ldots, \alpha_j;\ \beta_1, \ldots, \beta_{j-1})$
  end if
end for

Let $(\lambda_i, \mathbf{s}_i)$ be an eigenpair of T_m,

$$T_m\mathbf{s}_i^{(m)} = \vartheta_i^{(m)}\mathbf{s}_i^{(m)}. \tag{9.15}$$

Then,

$$AQ_m\mathbf{s}_i^{(m)} = Q_mT_m\mathbf{s}_i^{(m)} = \vartheta_i^{(m)}Q_m\mathbf{s}_i^{(m)}. \tag{9.16}$$

So, the eigenvalues of T_m are also eigenvalues of A. The eigenvector of A corresponding to the eigenvalue ϑ_i is

$$\mathbf{y}_i = Q_m\mathbf{s}_i^{(m)} = [\mathbf{q}_1, \ldots, \mathbf{q}_m]\,\mathbf{s}_i^{(m)} = \sum_{j=1}^{m} \mathbf{q}_j s_{ji}^{(m)}. \tag{9.17}$$

The cost of a single iteration step of Algorithm 9.3 does not depend on the index of the iteration! In a single iteration step we have to execute a matrix-vector multiplication and $7n$ further floating point operations.

Remark 9.1. In certain very big applications the Lanczos vectors cannot be stored for reasons of limited memory. In this situation, the Lanczos algorithm is executed *without* building the matrix Q. When the desired eigenvalues and Ritz vectors have been determined from (9.15) the Lanczos algorithm is repeated and the desired eigenvectors are accumulated on the fly using (9.17). □

9.4 The Lanczos process as an iterative method

The Lanczos Algorithm 9.3 essentially determines an invariant Krylov subspace $\mathcal{K}^m(\mathbf{x})$ of $\mathbb{F}^n$. More precisely, it constructs an orthonormal basis $\{\mathbf{q}_1, \ldots, \mathbf{q}_m\}$ of $\mathcal{K}^m(\mathbf{x})$. The

projection of A onto this space is a Hessenberg or even a real tridiagonal matrix if A is Hermitian.

We have seen in section 8.4 that the eigenvalues at the end of the spectrum are approximated very quickly in Krylov spaces. Therefore, only a very few iteration steps may be required to get those eigenvalues (and corresponding eigenvectors) within the desired accuracy, i.e., $|\vartheta_i^{(j)} - \lambda_i|$ may be tiny for $j \ll m$.

The Ritz values $\vartheta_i^{(j)}$ are the eigenvalues of the tridiagonal matrices T_j that are generated element by element in the course of the Lanczos algorithm. They can be computed efficiently by, e.g., the tridiagonal QR algorithm in $\mathcal{O}(j^2)$ flops. The cost for computing the eigenvalues of T_j are in general negligible compared with the cost for forming $A\mathbf{q}_j$.

But how can the error $|\vartheta_i^{(j)} - \lambda_i|$ be estimated? We will adapt the following more general lemma to this end.

Lemma 9.1 (**Eigenvalue inclusion of Krylov–Bogoliubov** [5] [7, p.69]) *Let $A \in \mathbb{F}^{n\times n}$ be Hermitian. Let $\vartheta \in \mathbb{R}$ and $\mathbf{x} \in \mathbb{F}^n$ with $\mathbf{x} \neq \mathbf{0}$ be arbitrary. Set $\tau := \|(A - \vartheta I)\mathbf{x}\|/\|\mathbf{x}\|$. Then there is an eigenvalue of A in the interval $[\vartheta - \tau, \vartheta + \tau]$.*

Proof. Let

$$A = U\Lambda U = \sum_{i=1}^{n} \lambda_i \mathbf{u}_i \mathbf{u}_i^*$$

be the spectral decomposition of A. Then,

$$(A - \vartheta I)\mathbf{x} = \sum_{i=1}^{n} (\lambda_i \mathbf{u}_i \mathbf{u}_i^* - \vartheta \mathbf{u}_i \mathbf{u}_i^*)\mathbf{x} = \sum_{i=1}^{n} (\lambda_i - \vartheta)(\mathbf{u}_i^* \mathbf{x})\mathbf{u}_i.$$

Taking norms, we obtain

$$\|(A - \vartheta I)\mathbf{x}\|^2 = \sum_{i=1}^{n} |\lambda_i - \vartheta|^2 |\mathbf{u}_i^* \mathbf{x}|^2 \geq |\lambda_k - \vartheta|^2 \sum_{i=1}^{n} |\mathbf{u}_i^* \mathbf{x}|^2 = |\lambda_k - \vartheta|^2 \|\mathbf{x}\|,$$

where λ_k is the eigenvalue closest to τ, i.e., $|\lambda_k - \vartheta| \leq |\lambda_i - \vartheta|$ for all i. ∎

We want to apply this Lemma to the case where the vector is a Ritz vector $\mathbf{y}_i^{(j)}$ corresponding to the Ritz value $\tau = \vartheta_i^{(j)}$ as obtained in the j-th step of the Lanczos algorithm. Then,

$$\mathbf{y}_i^{(j)} = Q_j \mathbf{s}_i^{(j)}, \qquad T_j \mathbf{s}_i^{(j)} = \vartheta_i^{(j)} \mathbf{s}_i^{(j)}.$$

Thus, by employing the Lanczos relation (9.13),

$$\begin{aligned} \|A\mathbf{y}_i^{(j)} - \vartheta_i^{(j)} \mathbf{y}_i^{(j)}\| &= \|AQ_j \mathbf{s}_i^{(j)} - \vartheta_i^{(j)} Q_j \mathbf{s}_i^{(j)}\| \\ &= \|(AQ_j - Q_j T_j)\mathbf{s}_i^{(j)}\| \\ &= \|\beta_j \mathbf{q}_{j+1} \mathbf{e}_j^* \mathbf{s}_i^{(j)}\| = |\beta_j| |\mathbf{e}_j^* \mathbf{s}_i^{(j)}| = |\beta_j| |s_{ji}^{(j)}|. \end{aligned}$$

$s_{ji}^{(j)}$ is the j-th, i.e., the last element of the eigenvector matrix S_j of T_j,

$$T_j S_j = S_j \Theta_j, \qquad \Theta_j = \operatorname{diag}(\vartheta_1^{(j)}, \cdots, \vartheta_j^{(j)}).$$

According to Lemma 9.1 there is an eigenvalue λ of A such that

$$|\lambda - \vartheta_i^{(j)}| \leq \beta_j |s_{ji}|. \tag{9.18}$$

Thus, it is possible to get good eigenvalue approximations even if β_j is not small! Further, we know that [7, §11.7]

$$\sin \angle(\mathbf{y}_i^{(j)}, \mathbf{z}) \leq \beta_j \frac{|s_{ji}|}{\gamma}, \tag{9.19}$$

where $\mathbf{z}$ is the eigenvector corresponding to λ in (9.18) and γ is the gap between λ and the next eigenvalue $\neq \lambda$ of A. In an actual computation, γ is not known. Parlett suggests to replace γ by the distance of $\vartheta_i^{(j)}$ to the next $\vartheta_k^{(j)}$, $k \neq i$. Because the $\vartheta_i^{(j)}$ converge to eigenvalues of A this substitution will give a reasonable number, at least in the limit.

In order to use the estimate (9.18) we need to compute all eigenvalues of T_j and the last row of S_j. It is possible and in fact straightforward to compute this row without the rest of S_j. The algorithm, a simple modification of the tridiagonal QR algorithm, has been introduced by Golub and Welsch [3] in connection with the computation of interpolation points and weights in Gaussian integration.

A numerical example

This numerical example is intended to show that the implementation of the Lanczos algorithm is not as simple as it seems from the previous. Let

$$A = \operatorname{diag}(0, 1, 2, 3, 4, 100000)$$

and

$$\mathbf{x} = (1, 1, 1, 1, 1, 1)^T.$$

The diagonal matrix A has six simple eigenvalues and $\mathbf{x}$ has a non-vanishing component in the direction of each eigenspace. Thus, the Lanczos algorithm should stop after $m = n = 6$ iteration steps with the complete Lanczos relation. Up to rounding error, we expect that $\beta_6 = 0$ and that the eigenvalues of T_6 are identical with those of A. Let's see what happens if Algorithm 9.3 is applied with these input data. in the sequel we present the numbers that we obtained with a MATLAB implementation of this algorithm.

$\boxed{j = 1}$

$$\alpha_1 = 16668.33333333334, \quad \beta_1 = 37267.05429136513.$$

$\boxed{j = 2}$

$$\alpha_2 = 83333.66652666384, \quad \beta_2 = 3.464101610531258.$$

The diagonal of the eigenvalue matrix Θ_2 is:

$$\operatorname{diag}(\Theta_2) = (1.999959999195565, 99999.99989999799)^T.$$

The last row of $\beta_2 S_2$ is

$$\beta_2 S_{2,:} = (1.4142135626139063.162277655014521).$$

The matrix of Ritz vectors $Y_2 = Q_2 S_2$ is

$$\begin{pmatrix} -0.44722 & -2.0000 \cdot 10^{-05} \\ -0.44722 & -9.9998 \cdot 10^{-06} \\ -0.44721 & 4.0002 \cdot 10^{-10} \\ -0.44721 & 1.0001 \cdot 10^{-05} \\ -0.44720 & 2.0001 \cdot 10^{-05} \\ 4.4723 \cdot 10^{-10} & 1.0000 \end{pmatrix}$$

$\boxed{j=3}$

$$\alpha_3 = 2.000112002245340 \qquad \beta_3 = 1.183215957295906.$$

The diagonal of the eigenvalue matrix is

$$\operatorname{diag}(\Theta_3) = (0.5857724375775532, 3.414199561869119, 99999.99999999999)^T .$$

The largest eigenvalue has converged already. This is not surprising as $\lambda_2/\lambda_1 = 4\cdot 10^{-5}$. With simple vector iteration the eigenvalues would converge with the factor $\lambda_2/\lambda_1 = 4\cdot 10^{-5}$.

The last row of $\beta_3 S_3$ is

$$\beta_3 S_{3,:} = \left(0.8366523355001995, 0.8366677176165411, 3.741732220526109\cdot 10^{-05}\right).$$

The matrix of Ritz vectors $Y_3 = Q_3 S_3$ is

$$\begin{pmatrix} 0.76345 & 0.13099 & 2.0000\cdot 10^{-10} \\ 0.53983 & -0.09263 & -1.0001\cdot 10^{-10} \\ 0.31622 & -0.31623 & -2.0001\cdot 10^{-10} \\ 0.09262 & -0.53984 & -1.0000\cdot 10^{-10} \\ -0.13098 & -0.76344 & 2.0001\cdot 10^{-10} \\ -1.5864\cdot 10^{-13} & -1.5851\cdot 10^{-13} & 1.00000 \end{pmatrix}$$

The largest element (in modulus) of $Y_3^T Y_3$ is $\approx 3\cdot 10^{-12}$.

The Ritz vectors (and thus the Lanczos vectors $\mathbf{q}_i$) are mutually orthogonal up to rounding error.

$\boxed{j=4}$

$$\alpha_4 = 2.000007428756856 \qquad \beta_4 = 1.014186947306611.$$

The diagonal of the eigenvalue matrix is

$$\operatorname{diag}(\Theta_4) = \begin{pmatrix} 0.1560868732577987 \\ 1.999987898940119 \\ 3.843904656006355 \\ 99999.99999999999 \end{pmatrix}.$$

The last row of $\beta_4 S_4$ is

$$\beta_4 S_{4,:} = \left(0.46017, -0.77785, -0.46018, 3.7949\cdot 10^{-10}\right).$$

The matrix of Ritz vectors $Y_4 = Q_4 S_4$ is

$$\begin{pmatrix} -0.82515 & 0.069476 & -0.40834 & -0.18249 \\ -0.034415 & 0.41262 & -0.40834 & -0.18243 \\ 0.37812 & 0.37781 & -0.40834 & -0.18236 \\ 0.41256 & -0.034834 & -0.40834 & -0.18230 \\ 0.069022 & -0.82520 & -0.40834 & -0.18223 \\ -1.3202\cdot 10^{-04} & 1.3211\cdot 10^{-04} & -0.40777 & 0.91308 \end{pmatrix}.$$

The largest element (in modulus) of $Y_4^T Y_4$ is $\approx 2\cdot 10^{-8}$.

We have $\beta_4 s_{4,4} \doteq 4\cdot 10^{-10}$. So, according to our previous estimates $(\vartheta_4, \mathbf{y}_4)$, $\mathbf{y}_4 = Y_4\mathbf{e}_4$ is a very good approximation for an eigenpair of A. This is in fact the case.

Notice that $Y_4^T Y_4$ has off diagonal elements of the order 10^{-8}. These elements are in the last row/column of $Y_4^T Y_4$. This means that all Ritz vectors have a small but not negligible component in the direction of the 'largest' Ritz vector.

$\boxed{j = 5}$

$$\alpha_5 = 2.363169101109444 \quad \beta_5 = 190.5668098726485.$$

The diagonal of the eigenvalue matrix is

$$\mathrm{diag}(\Theta_5) = \begin{pmatrix} 0.04749223464478182 \\ 1.413262891598485 \\ 2.894172742223630 \\ 4.008220660846780 \\ 9.999999999999999 \cdot 10^4 \end{pmatrix}.$$

The last row of $\beta_5 S_5$ is

$$\beta_5 S_{5,:} = \left(-43.570 - 111.38134.0963.4957.2320 \cdot 10^{-13}\right).$$

The matrix of Ritz vectors Y_5 is

$$\begin{pmatrix} -0.98779 & -0.084856 & 0.049886 & 0.017056 & -1.1424 \cdot 10^{-17} \\ -0.14188 & 0.83594 & -0.21957 & -0.065468 & -7.2361 \cdot 10^{-18} \\ 0.063480 & 0.54001 & 0.42660 & 0.089943 & -8.0207 \cdot 10^{-18} \\ -0.010200 & -0.048519 & 0.87582 & -0.043531 & -5.1980 \cdot 10^{-18} \\ -0.0014168 & -0.0055339 & 0.015585 & -0.99269 & -1.6128 \cdot 10^{-17} \\ 4.3570 \cdot 10^{-4} & 0.0011138 & -0.0013409 & -6.3497 \cdot 10^{-4} & 1.0000 \end{pmatrix}$$

Evidently, the last column of Y_5 is an excellent eigenvector approximation. Notice, however, that all Ritz vectors have a relatively large ($\sim 10^{-4}$) last component. This, gives rise to quite large off-diagonal elements of $Y_5^{\mathrm{T}} Y_5 - I_5 =$

$$\begin{pmatrix} 2.220 \cdot 10^{-16} & -1.587 \cdot 10^{-16} & -3.430 \cdot 10^{-12} & -7.890 \cdot 10^{-9} & -7.780 \cdot 10^{-4} \\ -1.587 \cdot 10^{-16} & -1.110 \cdot 10^{-16} & 1.283 \cdot 10^{-12} & -1.764 \cdot 10^{-8} & -1.740 \cdot 10^{-3} \\ -3.430 \cdot 10^{-12} & 1.283 \cdot 10^{-12} & 0 & 5.6800 \cdot 10^{-17} & -6.027 \cdot 10^{-8} \\ -7.890 \cdot 10^{-9} & -1.764 \cdot 10^{-8} & 5.6800 \cdot 10^{-17} & -2.220 \cdot 10^{-16} & 4.187 \cdot 10^{-16} \\ -7.780 \cdot 10^{-4} & -1.740 \cdot 10^{-3} & -6.027 \cdot 10^{-8} & 4.187 \cdot 10^{-16} & -1.110 \cdot 10^{-16} \end{pmatrix}.$$

Similarly as with $j = 4$, the first four Ritz vectors satisfy the orthogonality condition very well. But they are not perpendicular to the last Ritz vector.

$\boxed{j = 6}$

$$\alpha_6 = 99998.06336906151 \qquad \beta_6 = 396.6622037049789$$

The diagonal of the eigenvalue matrix is

$$\mathrm{diag}(\Theta_6) = \begin{pmatrix} 0.02483483859326367 \\ 1.273835519171372 \\ 2.726145019098232 \\ 3.975161765440400 \\ 9.999842654044850 \cdot 10^{+4} \\ 1.000000000000000 \cdot 10^{+5} \end{pmatrix}.$$

The eigenvalues are not the exact ones, as was to be expected. We even have *two* copies of the largest eigenvalue of A in Θ_6! The last row of $\beta_6 S_6$ is

$$\beta_6 S_{6,:} = \left(-0.20603, 0.49322, 0.49323, 0.20604, 396.66, -8.6152 \cdot 10^{-15}\right)$$

although theory predicts that $\beta_6 = 0$. The sixth entry of $\beta_6 S_6$ is very small, which means that the sixth Ritz value and the corresponding Ritz vector are good approximations to an eigenpair of A. In fact, eigenvalue and eigenvector are accurate to machine precision.

$\beta_5 s_{6,5}$ does not predict the fifth column of Y_6 to be a good eigenvector approximation, although the angle between the fifth and sixth column of Y_6 is less than 10^{-3}. The last two columns of Y_6 are

$$\begin{pmatrix} -4.7409 \cdot 10^{-4} & -3.3578 \cdot 10^{-17} \\ 1.8964 \cdot 10^{-3} & -5.3735 \cdot 10^{-17} \\ -2.8447 \cdot 10^{-3} & -7.0931 \cdot 10^{-17} \\ 1.8965 \cdot 10^{-3} & -6.7074 \cdot 10^{-17} \\ -4.7414 \cdot 10^{-4} & -4.9289 \cdot 10^{-17} \\ -0.99999 & 1.0000 \end{pmatrix}.$$

As $\beta_6 \neq 0$ one could continue the Lanczos process and compute ever larger tridiagonal matrices. If one proceeds in this way one obtains multiple copies of certain eigenvalues [2, 2]. The corresponding values $\beta_j s_{ji}^{(j)}$ will be tiny. The corresponding Ritz vectors will be 'almost' linearly dependent.

From this numerical example we see that the problem of the Lanczos algorithm consists in the loss of orthogonality among Ritz vectors which is a consequence of the loss of orthogonality among Lanczos vectors, since $Y_j = Q_j S_j$ and S_j is unitary (up to roundoff).

To verify this diagnosis, we rerun the Lanczos algorithm with *complete reorthogonalization*. This procedure amounts to the Arnoldi algorithm 9.1. Ir can be accomplished by modifying line 11 in the Lanczos algorithm 9.3, see Algorithm 9.4.

Algorithm 9.4 Lanczos algorithm with full reorthogonalization

11: $\mathbf{r} := \mathbf{r} - \alpha_j \mathbf{q}$; $\mathbf{r} := \mathbf{r} - Q(Q^* \mathbf{r})$;

Of course, the cost of the algorithm increases considerably. The j-th step of the algorithm requires now a matrix-vector multiplication and $(2j + \mathcal{O}(1))n$ floating point operations.

A numerical example [continued]

With matrix and initial vector as before Algorithm 9.4 gives the following numbers.

$\boxed{j = 1}$

$$\alpha_1 = 16668.33333333334, \quad \beta_1 = 37267.05429136513.$$

$\boxed{j = 2}$

$$\alpha_2 = 83333.66652666384, \quad \beta_2 = 3.464101610531258.$$

The diagonal of the eigenvalue matrix Θ_2 is:

$$\operatorname{diag}(\Theta_2) = (1.999959999195565, 99999.99989999799)^T.$$

$\boxed{j = 3}$

$$\alpha_3 = 2.000112002240894 \quad \beta_3 = 1.183215957295905$$

The diagonal of the eigenvalue matrix is

$$\operatorname{diag}(\Theta_3) = (0.5857724375677908, 3.414199561859357, 100000.0000000000)^T.$$

$\boxed{j=4}$

$$\alpha_4 = 2.000007428719501 \qquad \beta_4 = 1.014185105707661$$

$$\mathrm{diag}(\Theta_4) = \begin{pmatrix} 0.1560868732475296 \\ 1.999987898917647 \\ 3.843904655996084 \\ 99999.99999999999 \end{pmatrix}$$

The matrix of Ritz vectors $Y_4 = Q_4 S_4$ is

$$\begin{pmatrix} -0.93229 & 0.12299 & 0.03786 & -1.1767 \cdot 10^{-15} \\ -0.34487 & -0.49196 & -0.10234 & 2.4391 \cdot 10^{-15} \\ 2.7058 \cdot 10^{-6} & -0.69693 & 2.7059 \cdot 10^{-6} & 4.9558 \cdot 10^{-17} \\ 0.10233 & -0.49195 & 0.34488 & -2.3616 \cdot 10^{-15} \\ -0.03786 & 0.12299 & 0.93228 & 1.2391 \cdot 10^{-15} \\ 2.7086 \cdot 10^{-17} & 6.6451 \cdot 10^{-17} & -5.1206 \cdot 10^{-17} & 1.00000 \end{pmatrix}$$

The largest off-diagonal element of $|Y_4^T Y_4|$ is about $2 \cdot 10^{-16}$

$\boxed{j=5}$

$$\alpha_5 = 2.000009143040107 \qquad \beta_5 = 0.7559289460488005$$

$$\mathrm{diag}(\Theta_5) = \begin{pmatrix} 0.02483568754088384 \\ 1.273840384543175 \\ 2.726149884630423 \\ 3.975162614480485 \\ 10000.000000000000 \end{pmatrix}$$

The Ritz vectors are $Y_5 =$

$$\begin{pmatrix} -9.91 \cdot 10^{-01} & -4.62 \cdot 10^{-02} & 2.16 \cdot 10^{-02} & -6.19 \cdot 10^{-03} & -4.41 \cdot 10^{-18} \\ -1.01 \cdot 10^{-01} & 8.61 \cdot 10^{-01} & -1.36 \cdot 10^{-01} & -3.31 \cdot 10^{-02} & 1.12 \cdot 10^{-17} \\ 7.48 \cdot 10^{-02} & 4.87 \cdot 10^{-01} & 4.87 \cdot 10^{-01} & -7.48 \cdot 10^{-02} & -5.89 \cdot 10^{-18} \\ -3.31 \cdot 10^{-02} & -1.36 \cdot 10^{-01} & 8.61 \cdot 10^{-01} & -1.01 \cdot 10^{-01} & 1.07 \cdot 10^{-17} \\ 6.19 \cdot 10^{-03} & 2.16 \cdot 10^{-02} & -4.62 \cdot 10^{-02} & -9.91 \cdot 10^{-01} & 1.13 \cdot 10^{-17} \\ 5.98 \cdot 10^{-18} & 1.58 \cdot 10^{-17} & -3.39 \cdot 10^{-17} & -5.96 \cdot 10^{-17} & 1.000000000000000 \end{pmatrix}$$

Largest off-diagonal element of $|Y_5^T Y_5|$ is about 10^{-16} The last row of $\beta_5 S_5$ is

$$\beta_5 S_{5,:} = \left(-0.20603, -0.49322, 0.49322, 0.20603, 2.8687 \cdot 10^{-15}\right).$$

$\boxed{j=6}$

$$\alpha_6 = 2.000011428799386 \qquad \beta_6 = 4.178550866749342 \cdot 10^{-28}$$

$$\mathrm{diag}(\Theta_6) = \begin{pmatrix} 7.950307079340746 \cdot 10^{-13} \\ 1.000000000000402 \\ 2.000000000000210 \\ 3.000000000000886 \\ 4.000000000001099 \\ 9.999999999999999 \cdot 10^{4} \end{pmatrix}$$

The Ritz vectors are very accurate. Y_6 is almost the identity matrix are 1.0. The largest off diagonal element of $Y_6^T Y_6$ is about 10^{-16}. Finally,

$$\beta_6 S_{6,:} = \left(4.99 \cdot 10^{-29}, -2.00 \cdot 10^{-28}, 3.00 \cdot 10^{-28}, -2.00 \cdot 10^{-28}, 5.00 \cdot 10^{-29}, 1.20 \cdot 10^{-47}\right).$$

With a much enlarged effort we have obtained the desired result. Thus, the loss of orthogonality among the Lanczos vectors can be prevented by the explicit reorthogonalization against *all* previous Lanczos vectors. This amounts to applying the Arnoldi algorithm. In the sequel we want to better understand when the loss of orthogonality actually happens.

9.5 An error analysis of the unmodified Lanczos algorithm

When the quantities $Q_j, T_j, \mathbf{r}_j$, etc., are computed numerically by using the Lanczos algorithm, they can deviate greatly from their theoretical counterparts. However, despite this gross deviation from the exact model, it nevertheless delivers fully accurate Ritz value and Ritz vector approximations.

In this section $Q_j, T_j, \mathbf{r}_j$ etc. denote the *numerically computed* values and not their theoretical counterparts. So, instead of the Lanczos relation (9.13) we write

$$AQ_j - Q_j T_j = \mathbf{r}_j \mathbf{e}_j^* + F_j \tag{9.20}$$

where the matrix F_j accounts for errors due to roundoff. Similarly, we write

$$I_j - Q_j^* Q_j = C_j^* + \Delta_j + C_j, \tag{9.21}$$

where Δ_j is a diagonal matrix and C_j is a *strictly* upper triangular matrix (with zero diagonal). Thus, $C_j^* + \Delta_j + C_j$ indicates the deviation of the Lanczos vectors from orthogonality.

We make the following **assumptions**

1. The tridiagonal eigenvalue problem can be solved exactly, i.e.,

$$T_j = S_j \Theta_j S_j^*, \quad S_j^* = S_j^{-1}, \quad \Theta_j = \operatorname{diag}(\vartheta_1, \ldots, \vartheta_j). \tag{9.22}$$

2. The orthogonality of the Lanczos vectors holds *locally*, i.e.,

$$\mathbf{q}_{i+1}^* \mathbf{q}_i = 0, \quad i = 1, \ldots, j-1, \quad \text{and} \quad \mathbf{r}_j^* \mathbf{q}_i = 0. \tag{9.23}$$

3. Furthermore,

$$\|\mathbf{q}_i\| = 1. \tag{9.24}$$

So, we assume that the computations that we actually perform (like orthogonalizations or solving the eigenvalue problem) are accurate. These assumptions imply that $\Delta_j = O$ and $c_{i,i+1}^{(j)} = 0$ for $i = 1, \ldots, j-1$.

We premultiply (9.20) by Q_j^* and obtain

$$Q_j^* A Q_j - Q_j^* Q_j T_j = Q_j^* \mathbf{r}_j \mathbf{e}_j^* + Q_j^* F_j \tag{9.25}$$

In order to eliminate A we subtract from this equation its transposed,

$$\begin{aligned} Q_j^* \mathbf{r}_j \mathbf{e}_j^* - \mathbf{e}_j \mathbf{r}_j^* Q_j &= -Q_j^* Q_j T_j + T_j Q_j^* Q_j + Q_j^* F_j - F_j^* Q_j, \\ &= (I - Q_j^* Q_j) T_j - T_j (I - Q_j^* Q_j) + Q_j^* F_j - F_j^* Q_j, \\ &\overset{(9.21)}{=} (C_j + C_j^*) T_j - T_j (C_j + C_j^*) + Q_j^* F_j - F_j^* Q_j, \\ &= \underbrace{(C_j T_j - T_j C_j)}_{\text{upper triangular}} + \underbrace{(C_j^* T_j - T_j C_j^*)}_{\text{lower triangular}} - F_j^* Q_j + Q_j^* F_j. \end{aligned} \tag{9.26}$$

$F_j^*Q_j - Q_j^*F_j$ is skew symmetric. Therefore we have

$$F_j^*Q_j - Q_j^*F_j = -K_j^* + K_j,$$

where K_j is an upper triangular matrix with zero diagonal. Thus, (9.25) has the form

$$\begin{pmatrix} & & \times \\ & \mathrm{O} & \vdots \\ & & \times \\ \underbrace{\times \cdots \times}_{\text{First } j-1 \text{ components of } \mathbf{r}_j^*Q_j.} & & 0 \end{pmatrix} = \begin{pmatrix} 0 & & C_jT_j - T_jC_j \\ & \ddots & \\ C_j^*T_j - T_jC_j^* & & 0 \end{pmatrix} + \begin{pmatrix} 0 & & K_j \\ & \ddots & \\ -K_j^* & & 0 \end{pmatrix}.$$

As the last component of $Q_j^*\mathbf{r}_j$ vanishes, we can treat these triangular matrices separately. For the upper triangular matrices we have

$$Q_j^*\mathbf{r}_j\mathbf{e}_j^* = C_jT_j - T_jC_j + K_j.$$

Multiplication by $\mathbf{s}_i^*$ and $\mathbf{s}_i$, respectively, from left and right gives

$$\underbrace{\mathbf{s}_i^*Q_j^*}_{\mathbf{y}_i^*}\ \underbrace{\mathbf{r}_j}_{\beta_j\mathbf{q}_{j+1}}\ \underbrace{\mathbf{e}_j^*\mathbf{s}_i}_{s_{ji}} = \mathbf{s}_i^*(C_jT_j - T_jC_j)\mathbf{s}_i + \mathbf{s}_i^*K_j\mathbf{s}_i.$$

Let $G_j := S_i^*K_jS_i$. Then we have

$$\beta_j s_{ji}\mathbf{y}_i^*\mathbf{q}_{j+1} = s_{ji}\mathbf{y}_i^*\mathbf{r}_j = \mathbf{s}_i^*C_j\mathbf{s}_i\vartheta_i - \vartheta_i\mathbf{s}_i^*C_j\mathbf{s}_i + g_{ii}^{(j)} = g_{ii}^{(j)}. \tag{9.27}$$

We now multiply (9.25) with $\mathbf{s}_i^*$ from the left and with $\mathbf{s}_k$ from the right. As $Q_j\mathbf{s}_i = \mathbf{y}_i$, we have

$$\mathbf{y}_i^*A\mathbf{y}_k - \mathbf{y}_i^*\mathbf{y}_k\vartheta_k = \mathbf{y}_i^*\mathbf{r}_j\mathbf{e}_j^*\mathbf{s}_k + \mathbf{s}_i^*Q_j^*F_j\mathbf{s}_k.$$

Now, from this equation we subtract again its transposed, such that A is eliminated,

$$\begin{aligned}\mathbf{y}_i^*\mathbf{y}_k(\vartheta_i - \vartheta_k) &= \mathbf{y}_i^*\mathbf{r}_j\mathbf{e}_j^*\mathbf{s}_k - \mathbf{y}_k^*\mathbf{r}_j\mathbf{e}_j^*\mathbf{s}_i + \mathbf{s}_i^*Q_j^*F_j\mathbf{s}_k - \mathbf{s}_k^*Q_j^*F_j\mathbf{s}_i \\ &\overset{(9.27)}{=} \left(\frac{g_{ii}^{(j)}}{s_{ji}^{(j)}}\right)s_{jk} - \left(\frac{g_{kk}^{(j)}}{s_{jk}^{(j)}}\right)s_{ji} \\ &\quad + \frac{1}{2}(\mathbf{s}_i^*Q_j^*F_j\mathbf{s}_k + \mathbf{s}_k^*F_j^*Q_j\mathbf{s}_i) - \frac{1}{2}(\mathbf{s}_k^*Q_j^*F_j\mathbf{s}_i + \mathbf{s}_i^*F_j^*Q_j\mathbf{s}_k) \\ &= g_{ii}^{(j)}\frac{s_{jk}^{(j)}}{s_{ji}^{(j)}} - g_{kk}^{(j)}\frac{s_{ji}^{(j)}}{s_{jk}^{(j)}} - (g_{ik}^{(j)} - g_{ki}^{(j)}).\end{aligned}$$

Thus we have proved

Theorem 9.2 (Paige, see [7, p.266]) *With the above notations we have*

$$\mathbf{y}_i^{(j)*}\mathbf{q}_{j+1} = \frac{g_{ii}^{(j)}}{\beta_j s_{ji}^{(j)}} \tag{9.28}$$

$$(\vartheta_i^{(j)} - \vartheta_k^{(j)})\mathbf{y}_i^{(j)*}\mathbf{y}_k^{(j)} = g_{ii}^{(j)}\frac{s_{jk}^{(j)}}{s_{ji}^{(j)}} - g_{kk}^{(j)}\frac{s_{ji}^{(j)}}{s_{jk}^{(j)}} - (g_{ik}^{(j)} - g_{ki}^{(j)}). \tag{9.29}$$

■

We can **interpret** these equations in the following way.

- From numerical experiments it is known that equation (9.20) is always satisfied to machine precision. Thus, $\|F_j\| \approx \varepsilon \|A\|$. Therefore, $\|G_j\| \approx \varepsilon \|A\|$, and, in particular, $|g_{ik}^{(j)}| \approx \varepsilon \|A\|$.

- We see from (9.28) that $|\mathbf{y}_i^{(j)*}\mathbf{q}_{j+1}|$ becomes large if $\beta_j|s_{ji}^{(j)}|$ becomes small, i.e., if the Ritz vector $\mathbf{y}_i^{(j)}$ is a good approximation of the corresponding eigenvector. Thus, each *new* Lanczos vector has a significant component in the direction of converged ('good') Ritz vectors.

  As a consequence: $\boxed{\textbf{convergence} \Longleftrightarrow \textbf{loss of orthogonality}}$.

- Let $|s_{ji}^{(j)}| \ll |s_{jk}^{(j)}|$, i.e., $\mathbf{y}_i^{(j)}$ is a 'good' Ritz vector in contrast to $\mathbf{y}_k^{(j)}$ that is a 'bad' Ritz vector. Then in the first term on the right of (9.29) two small ($\mathcal{O}(\varepsilon)$) quantities counteract each other such that the right hand side in (9.29) becomes large, $\mathcal{O}(1)$. If the corresponding Ritz values are well separated, $|\vartheta_i - \vartheta_k| = \mathcal{O}(1)$, then $|\mathbf{y}_i^*\mathbf{y}_k| \gg \varepsilon$. So, in this case also 'bad' Ritz vectors have a significant component in the direction of the 'good' Ritz vectors.

- If $|\vartheta_i - \vartheta_k| = \mathcal{O}(\varepsilon)$ and both $s_{ji}^{(j)}$ and $s_{jk}^{(j)}$ are of $\mathcal{O}(\varepsilon)$ the $s_{ji}^{(j)}/s_{jk}^{(j)} = \mathcal{O}(1)$ such that the right hand side of (9.29) as well as $|\vartheta_i - \vartheta_k|$ is $\mathcal{O}(\varepsilon)$. Therefore, we must have $\mathbf{y}_i^{(j)*}\mathbf{y}_k^{(j)} = \mathcal{O}(1)$. So, these two vectors are almost parallel.

9.6 Partial reorthogonalization

In Section 9.4 we have learned that the Lanczos algorithm does not yield orthogonal Lanczos vectors as it should in theory due to floating point arithmetic. In the previous section we learned that the loss of orthogonality happens as soon as Ritz vectors have converged accurately enough to eigenvectors. In this section we review an approach how to counteract the loss of orthogonality without executing *full* reorthogonalization [8, 9].

In [7] it is shown that if the Lanczos basis is **semiorthogonal**, i.e., if

$$W_j = Q_j^*Q_j = I_j + E, \qquad \|E\| < \sqrt{\varepsilon_M},$$

then the tridiagonal matrix T_j is the projection of A onto the subspace $\mathcal{R}(V_j)$,

$$T_j = N_j^*AN_j + G, \qquad \|G\| = O((\varepsilon_M)\|A\|),$$

where N_j is an orthonormal basis of $\mathcal{R}(Q_j)$. Therefore, it suffices to have semiorthogonal Lanczos vectors for computing accurate eigenvalues. Our goal is now to enforce semiorthogonality by monitoring the loss of orthogonality and to reorthogonalize if needed.

The computed Lanczos vectors satisfy

$$\beta_j\mathbf{q}_{j+1} = A\mathbf{q}_j - \alpha_j\mathbf{q}_j - \beta_{j-1}\mathbf{q}_{j-1} + \mathbf{f}_j, \tag{9.30}$$

where $\mathbf{f}_j$ accounts for the roundoff errors committed in the j-th iteration step. Let $W_j = ((\omega_{ik}))_{1\le i,\,k\le j}$. Premultiplying equation (9.30) by $\mathbf{q}_k^*$ gives

$$\beta_j\omega_{j+1,k} = \mathbf{q}_k^*A\mathbf{q}_j - \alpha_j\omega_{jk} - \beta_{j-1}\omega_{j-1,k} + \mathbf{q}_k^*\mathbf{f}_j. \tag{9.31}$$

Exchanging indices j and k in the last equation (9.31) gives

$$\beta_k \omega_{j,k+1} = \mathbf{q}_j^* A \mathbf{q}_k - \alpha_k \omega_{jk} - \beta_{k-1} \omega_{j,k-1} + \mathbf{q}_j^* \mathbf{f}_k. \tag{9.32}$$

By subtracting (9.32) from (9.31) we get

$$\beta_j \omega_{j+1,k} = \beta_k \omega_{j,k+1} + (\alpha_k - \alpha_j)\omega_{jk} - \beta_{k-1}\omega_{j,k-1} - \beta_{j-1}\omega_{j-1,k} - \mathbf{q}_j^* \mathbf{f}_k + \mathbf{q}_k^* \mathbf{f}_j. \tag{9.33}$$

Given W_j we employ equation (9.33) to compute the $j+1$-th row of W_{j+1}. However, elements $\omega_{j+1,j}$ and $\omega_{j+1,j+1}$ are not defined by (9.33). We can assign values to these two matrix entries by reasoning as follows.

- We set $\omega_{j+1,j+1} = 1$ because we explicitly normalize $\mathbf{q}_{j+1}$.
- We set $\omega_{j+1,j} = O(\varepsilon_M)$ because we explicitly orthogonalize $\mathbf{q}_{j+1}$ and $\mathbf{q}_j$.

For computational purposes, equation (9.33) is now replaced by

$$\begin{aligned} \tilde{\omega} &= \beta_k \omega_{j,k+1} + (\alpha_k - \alpha_j)\omega_{jk} - \beta_{k-1}\omega_{j,k-1} - \beta_{j-1}\omega_{j-1,k}, \\ \omega_{j+1,k} &= (\tilde{\omega} + \operatorname{sign}(\tilde{\omega}) \underbrace{2\varepsilon\|A\|}_{\substack{\text{the estimate of} \\ \mathbf{q}_j^*\mathbf{f}_k + \mathbf{q}_k^*\mathbf{f}_j}})/\beta_j. \end{aligned} \tag{9.34}$$

As soon as $\omega_{j+1,k} > \sqrt{\varepsilon_M}$ the vectors $\mathbf{q}_j$ and $\mathbf{q}_{j+1}$ are orthogonalized against *all* previous Lanczos vectors $\mathbf{q}_1, \ldots, \mathbf{q}_{j-1}$. Then the elements of last two lines of W_j are set equal to a number of size $\mathcal{O}(\varepsilon_M)$. Notice that only the last two rows of W_j have to be stored.

Numerical example

We perform the Lanczos algorithm with matrix

$$A = \operatorname{diag}(1, 2, \ldots, 50)$$

and initial vector

$$\mathbf{x} = [1, \ldots, 1]^*.$$

In the first experiment we execute 50 iteration steps. In Table 9.1 the base-10 logarithms of the values $|w_{i,j}|/\text{macheps}$ are listed where $|w_{i,j}| = |\mathbf{q}_i^* \mathbf{q}_j|$, $1 \le j \le i \le 50$ and macheps $\approx 2.2 \cdot 10^{-16}$. One sees how the $|w_{i,j}|$ steadily grow with increasing i and with increasing $|i-j|$.

In the second experiment we execute 50 iteration steps with partial reorthogonalization turned on. The estimators $\omega_{j,k}$ are computed according to (9.33),

$$\begin{aligned} \omega_{k,k} &= 1, \qquad k = 1, \ldots, j \\ \omega_{k,k-1} &= \psi_k, \qquad k = 2, \ldots, j \\ \omega_{j+1,k} &= \frac{1}{\beta_j} [\beta_k \omega_{j,k+1} + (\alpha_k - \alpha_j)\omega_{jk} \\ &\qquad -\beta_{k-1}\omega_{j,k-1} - \beta_{j-1}\omega_{j-1,k}] + \vartheta_{i,k}, \qquad 1 \le k \le j. \end{aligned} \tag{9.35}$$

Here, we set $\omega_{j,0} = 0$. The values ψ_k and $\vartheta_{i,k}$ could be defined to be random variables of the correct magnitude, i.e., $\mathcal{O}(\varepsilon k)$. Following a suggestion of Parlett [7] we used

$$\psi_k = \varepsilon\|A\|, \qquad \vartheta_{i,k} = \varepsilon\sqrt{\|A\|}.$$

0																																																	
0	0																																																
1	1	0																																															
1	0	1	0																																														
1	1	0	1	0																																													
1	1	1	0	1	0																																												
0	1	1	0	0	0	0																																											
1	0	1	1	0	1	1	0																																										
1	0	0	1	0	1	1	1	0																																									
1	0	0	1	1	1	1	0	1	0																																								
0	1	0	1	1	1	1	0	0	0	0																																							
0	1	0	1	1	0	1	1	0	0	0	0																																						
1	1	1	1	1	1	0	1	1	0	1	1	0																																					
1	1	1	1	1	1	0	0	1	0	1	0	1	0																																				
1	1	0	1	0	1	1	1	1	0	1	1	0	1	0																																			
1	1	1	0	1	1	1	0	1	0	1	1	1	0	1	0																																		
0	1	1	1	1	1	0	1	1	1	0	1	1	0	0	1	0																																	
1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	0																																
0	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0																															
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	0	0																														
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0	0																													
1	1	1	1	2	1	2	1	1	1	1	1	1	1	1	0	1	1	0	0	1	0																												
0	2	1	2	1	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	0	0	0																											
1	1	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	0	0	0																										
1	2	2	2	2	2	2	2	2	2	2	2	1	2	1	2	1	1	1	1	1	0	0	1	0																									
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	0	0	1	0																								
2	2	2	2	2	3	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	0	0	0	0																							
2	2	3	3	3	3	3	3	3	2	3	2	2	2	2	2	2	2	2	1	1	1	1	1	1	0	1	0																						
2	3	3	3	3	3	3	3	3	3	3	3	2	3	2	2	2	2	2	2	2	2	1	1	1	0	1	1	0																					
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	1	2	1	1	1	1	0																				
3	3	3	4	3	4	3	3	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	1	1	1	1	1	1	0																			
3	3	4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	2	2	2	2	2	2	1	2	1	1	0	1	0																		
4	4	4	4	4	4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	2	2	2	2	1	1	1	1	0	0	0																	
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	2	2	2	2	1	1	1	1	1	1	0																
4	5	5	5	5	5	5	5	5	5	4	5	4	4	4	4	4	4	4	4	3	3	3	3	3	2	2	2	2	2	1	1	1	1	0															
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	4	4	4	4	4	3	3	3	3	3	3	2	2	2	1	1	2	0	0	0														
5	5	5	6	5	6	5	5	5	5	5	5	5	5	5	5	5	5	4	4	4	4	4	4	3	3	3	3	2	2	2	1	1	1	1	0	0													
6	6	6	6	6	6	6	6	6	6	6	5	6	5	5	5	5	5	5	5	5	4	4	4	4	3	3	3	3	2	2	2	2	1	1	1	0	0												
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	5	5	5	5	5	5	4	4	4	4	4	3	3	3	2	2	2	2	1	1	0	0	0											
6	6	7	7	7	7	7	7	7	6	7	6	6	6	6	6	6	6	6	5	5	5	5	5	5	4	4	4	4	3	3	3	2	2	2	1	1	1	1	0										
7	7	7	7	7	7	7	7	7	7	7	7	7	7	6	7	6	6	6	6	6	6	5	5	5	5	4	4	4	4	3	3	3	3	2	2	1	1	1	1	0									
7	7	8	8	8	8	8	7	8	7	8	7	7	7	7	7	7	7	7	6	6	6	6	6	6	5	5	5	5	4	4	4	3	3	3	2	2	1	2	0	1	0								
8	8	8	8	8	8	8	8	8	8	8	8	8	8	7	8	7	7	7	7	7	7	6	6	6	6	5	5	5	5	4	4	4	4	3	3	2	1	1	1	1	1	0							
8	8	9	9	9	9	9	9	9	8	9	8	8	8	8	8	8	8	8	7	7	7	7	7	7	6	6	6	6	5	5	5	4	4	4	3	3	2	2	1	2	1	1	0						
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	8	8	8	8	8	8	7	7	7	7	7	6	6	6	5	5	5	5	4	4	3	3	3	2	1	2	1	1	0					
10	10	10	10	10	10	10	10	10	10	10	10	10	9	9	9	9	9	9	9	9	8	8	8	8	7	7	7	7	6	6	6	6	5	5	4	4	4	3	3	2	1	2	1	1	0				
10	11	10	11	10	11	10	11	10	11	10	10	10	10	10	10	10	10	9	9	9	9	9	9	8	8	8	8	7	7	7	7	6	6	5	5	5	4	4	3	3	2	1	2	1	2	0			
11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	10	10	10	10	10	10	9	9	9	9	8	8	8	8	7	7	7	6	6	6	5	5	4	4	3	1	1	2	1	1	0		
12	12	12	12	12	12	12	12	12	12	12	12	12	12	11	12	11	11	11	11	11	11	10	10	10	10	9	9	9	9	8	8	8	8	7	7	6	6	5	5	5	4	3	2	1	2	2	1	0	
13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	12	12	12	12	12	12	11	11	11	11	11	11	10	10	10	9	9	9	8	8	8	7	7	7	6	6	5	4	4	3	1	3	2	1	0

Table 9.1: MATLAB demo on the loss of orthogonality among Lanczos vectors. Unmodified Lanczos. `round(log10(abs(I-Q*_50 Q_50)/eps))`

Reorthogonalization takes place in the j-th Lanczos step if $\max_k(\omega_{j+1,k}) > \sqrt{\text{macheps}}$. $\mathbf{q}_{j+1}$ is orthogonalized against all vectors $\mathbf{q}_k$ with $\omega_{j+1,k} > \text{macheps}^{3/4}$. In the following iteration step also $\mathbf{q}_{j+2}$ is orthogonalized against these vectors. In Table 9.2 the base-10 logarithms of the values $|w_{i,j}|/\text{macheps}$ obtained with this procedure are listed where $|w_{i,j}| = |\mathbf{q}_i^* \mathbf{q}_j|$, $1 \le j \le i \le 50$ and macheps $\approx 2.2 \cdot 10^{-16}$. In Table 9.3 the base-10 logarithms of the estimates $|\omega_{i,j}|/\text{macheps}$ are given. The estimates are too high by (only) an order of magnitude. However, the procedure succeeds in that the resulting $\{\mathbf{q}_k\}$ *are* semi-orthogonal.

9.7 Block Lanczos

As we have seen, the Lanczos algorithm produces a sequence $\{\mathbf{q}_i\}$ of orthonormal vectors. These Lanczos vectors build an orthonormal basis for the Krylov subspace $\mathcal{K}^j(\mathbf{x}) = \text{span}\{\mathbf{q}_1, \ldots, \mathbf{q}_j\} \subset \mathbb{R}^n$. The restriction of A to $\mathcal{K}^j(\mathbf{x})$ is an unreduced tridiagonal matrix. However the Lanczos algorithm cannot detect the *multiplicity* of the eigenvalues it computes. This limitation prompted the development of the block version of the Lanczos process (*Block Lanczos algorithm*), which is capable of determining multiplicities of eigenvalues up to the block size.

The idea is not to start with a single vector $\mathbf{q}_1 \in \mathbb{R}^n$ but with a set of mutually orthogonal vectors which we take as the columns of the matrix $Q_1 \in \mathbb{R}^{n \times p}$ with the *block size* $p > 1$.

Associated with Q_1 is the 'big' Krylov subspace

$$\mathcal{K}^{jp}(Q_1) = \text{span}\{Q_1, AQ_1, \ldots, A^{j-1}Q_1\}. \tag{9.36}$$

(We suppose, for simplicity, that $A^{j-1}Q_1$ has rank p. Otherwise we would have to consider variable block sizes.)

The approach is similar to the scalar case with $p = 1$: Let $Q_1, \ldots, Q_j \in \mathbb{R}^{n \times p}$ be pairwise orthogonal block matrices ($Q_i^* Q_k = O$ for $i \ne k$) with orthonormal columns ($Q_i^* Q_i = I_p$ for all $i \le j$). Then, in the j-th iteration step, we obtain the matrix AQ_j and orthogonalize it against matrices Q_i, $i \le j$. The columns of the matrices are obtained by means of the QR factorization or with the Gram–Schmidt orthonormalization process. We obtained the following:

Algorithm 9.5 Block Lanczos algorithm

Choose $Q_1 \in \mathbb{F}^{n \times p}$ such that $Q_1^* Q_1 = I_p$. Set $j := 0$ and $\mathbb{F}^{n \times p} \ni V := 0$.
This algorithm generates a block tridiagonal matrix $\hat{T}_j$ with the diagonal blocks A_i, $i \le j$, the lower diagonal blocks $B_i, i < j$, and the Krylov basis $[Q_1, \ldots, Q_j]$ of $\mathcal{K}^{jp}(Q_1)$.
for $j \ge 0$ **do**
  if $j > 0$ **then**
    $V =: Q_{j+1} B_j$; /* QR decomposition */
    $V := -Q_j B_j^*$;
  end if
  $j := j + 1$;
  $A_j := Q_j^* V$;
  $V := V - Q_j A_j$;
  Test for convergence (Ritz pairs, evaluation of error)
end for

0																																																	
0	0																																																
1	1	0																																															
1	0	1	0																																														
1	1	0	1	0																																													
1	1	1	0	1	0																																												
0	1	1	0	0	0	0																																											
1	0	1	1	0	1	1	0																																										
1	0	0	1	0	1	1	1	0																																									
1	0	0	1	1	1	1	0	1	0																																								
0	1	0	1	1	1	1	0	0	0	0																																							
0	1	0	1	1	0	1	1	0	0	0	0																																						
1	1	1	1	1	1	0	1	1	0	1	1	0																																					
1	1	1	1	1	1	0	0	1	0	1	0	1	0																																				
1	1	0	1	0	1	1	1	1	0	1	1	0	1	0																																			
1	1	1	0	1	1	1	0	1	0	1	1	1	0	1	0																																		
0	1	1	1	1	1	0	1	1	1	0	1	1	0	0	1	0																																	
1	1	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	0																																
0	1	1	2	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0																															
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	0	0																														
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	0	0	0																													
1	1	1	1	2	1	2	1	1	1	1	1	1	1	1	0	1	1	0	0	1	0																												
0	2	1	2	1	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	0	0	0																											
1	1	2	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	0	0	0																										
1	2	2	2	2	2	2	2	2	2	2	2	1	2	1	2	1	1	1	1	1	0	0	1	0																									
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	0	0	1	0																								
2	2	2	2	2	3	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1	0	0	0	0																							
2	2	3	3	3	3	3	3	3	2	3	2	2	2	2	2	2	2	2	1	1	1	1	1	1	0	1	0																						
2	3	3	3	3	3	3	3	3	3	3	3	2	3	2	2	2	2	2	2	2	2	1	1	1	0	1	1	0																					
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	2	2	1	2	1	1	1	1	0																				
3	3	3	4	3	4	3	3	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	1	1	1	1	1	1	0																			
3	3	4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	2	2	2	2	2	2	1	2	1	1	0	1	0																		
4	4	4	4	4	4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	2	2	2	2	1	1	1	1	0	0	0																	
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	2	2	2	2	1	1	1	1	1	1	0																
4	5	5	5	5	5	5	5	5	5	4	5	4	4	4	4	4	4	4	4	3	3	3	3	3	2	2	2	2	2	1	1	1	1	0															
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	4	4	4	4	4	3	3	3	3	3	3	2	2	2	1	1	2	0	0	0														
5	5	5	6	5	6	5	5	5	5	5	5	5	5	5	5	5	5	4	4	4	4	4	4	3	3	3	3	2	2	2	1	1	1	1	0	0													
6	6	6	6	6	6	6	6	6	6	6	5	6	5	5	5	5	5	5	5	5	4	4	4	4	3	3	3	3	2	2	2	2	1	1	1	0	0												
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	5	5	5	5	5	5	4	4	4	4	4	3	3	3	2	2	2	2	1	1	0	0	0											
6	6	7	7	7	7	7	7	7	6	7	6	6	6	6	6	6	6	6	5	5	5	5	5	5	4	4	4	4	3	3	3	2	2	2	1	1	1	1	0										
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	2	2	1	1	1	1	0									
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	3	3	2	2	1	2	0	1	0								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	3	3	3	3	2	1	1	1	1	1	0							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	3	3	3	3	3	2	2	1	2	1	0	0						
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	3	3	3	4	3	3	3	3	2	1	2	1	1	0					
1	1	1	0	0	0	1	0	0	0	1	0	0	0	1	0	1	1	1	1	1	1	0	1	1	0	1	0	3	3	4	4	4	4	4	4	4	4	3	3	2	1	1	1	0	0				
1	1	2	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	2	1	2	1	1	1	1	2	0	3	4	4	4	4	4	4	4	4	4	4	4	3	3	2	2	2	1	1	0			
2	2	2	2	1	1	2	1	2	1	2	1	1	2	2	2	2	2	2	2	2	2	1	2	2	2	4	4	4	5	5	5	5	5	5	5	5	5	4	4	4	3	2	2	2	1	1	0		
3	3	3	3	3	2	2	2	2	3	2	3	2	2	2	3	3	3	3	3	3	2	3	3	3	4	5	5	5	5	6	6	6	6	6	5	5	5	5	5	5	4	3	2	2	2	1	1	0	
4	4	4	4	3	3	3	3	3	3	4	3	3	3	4	4	4	4	4	4	4	4	3	4	5	5	6	6	6	6	6	7	6	7	6	6	6	6	6	6	5	5	4	4	2	2	2	2	2	0

Table 9.2: Matlab demo on the loss of orthogonality among Lanczos vectors: Lanczos with partial reorthogonalization. `round(log10(abs(I-Q*_50 Q_50)/eps))`

0																																																	
2	0																																																
0	2	0																																															
1	0	2	0																																														
0	1	0	2	0																																													
1	0	1	0	2	0																																												
0	1	0	1	0	2	0																																											
1	0	1	0	1	0	2	0																																										
0	1	1	1	0	1	0	2	0																																									
1	1	1	1	1	1	1	0	2	0																																								
1	1	1	1	1	1	1	1	0	2	0																																							
1	1	1	1	1	1	1	1	1	0	2	0																																						
1	1	1	1	1	1	1	1	1	1	0	2	0																																					
1	1	1	1	1	1	1	1	1	1	1	0	2	0																																				
1	1	1	1	1	2	1	1	1	1	1	1	0	2	0																																			
1	1	2	1	2	1	2	1	2	1	1	1	1	0	2	0																																		
1	2	1	2	2	2	2	2	1	2	1	1	1	1	0	2	0																																	
1	2	2	2	2	2	2	2	2	1	2	1	2	1	1	0	2	0																																
1	2	2	2	2	2	2	2	2	2	2	2	1	2	1	1	0	2	0																															
2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	1	1	0	2	0																														
2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	1	1	0	2	0																													
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	2	1	1	0	2	0																												
2	2	2	3	2	3	2	3	2	2	2	2	2	2	2	2	2	2	1	2	0	2	0																											
2	2	3	3	3	3	3	3	3	2	3	2	2	2	2	2	2	2	2	1	2	0	2	0																										
2	3	3	3	3	3	3	3	3	3	3	3	2	3	2	2	2	2	2	2	1	2	0	2	0																									
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	2	1	2	0	2	0																								
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	3	2	2	2	2	1	2	0	2	0																							
3	3	4	3	4	3	4	3	4	3	3	3	3	3	3	3	3	3	3	2	2	2	2	1	2	0	2	0																						
3	4	4	4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3	2	2	2	2	1	2	0	2	0																					
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	2	2	2	2	1	2	0	2	0																				
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	3	4	3	3	3	3	2	3	2	2	1	2	0	2	0																			
4	4	5	5	5	5	5	5	5	4	5	4	4	4	4	4	4	4	4	3	3	3	3	3	3	2	2	1	2	0	2	0																		
4	5	5	5	5	5	5	5	5	5	5	5	4	5	4	4	4	4	4	4	3	3	3	3	3	3	2	2	2	2	0	2	0																	
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	5	4	4	4	4	4	4	3	3	3	3	2	2	2	2	0	2	0																
5	6	5	6	6	6	6	6	5	6	5	5	5	5	5	5	5	5	4	4	4	4	4	4	3	3	3	3	2	2	2	2	0	2	0															
6	6	6	6	6	6	6	6	6	6	6	6	6	5	5	5	5	5	5	5	5	4	4	4	4	3	3	3	3	2	2	2	2	0	2	0														
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	5	5	5	5	5	5	4	4	4	4	4	4	3	3	2	3	2	2	0	2	0													
6	6	7	7	7	7	7	7	7	7	7	6	6	6	6	6	6	6	6	5	5	5	5	5	5	4	4	4	4	3	3	3	3	2	2	0	2	0												
7	7	7	7	7	7	7	7	7	7	7	7	7	7	6	7	6	6	6	6	6	6	5	5	5	5	4	4	4	4	3	3	3	3	2	2	0	2	0											
7	7	8	7	8	8	8	7	8	7	7	7	7	7	7	7	7	7	7	6	6	6	6	5	5	5	5	5	5	4	4	3	3	3	3	2	2	0	2	0										
8	8	8	8	8	8	8	8	8	8	8	8	8	8	7	7	7	7	7	7	7	7	6	6	6	6	5	5	5	5	4	4	4	4	3	3	2	2	0	2	0									
7	7	8	8	8	8	8	7	8	7	7	7	7	7	7	7	7	7	7	6	6	6	6	5	6	5	5	5	5	4	4	3	3	4	4	3	3	2	2	0	2	0								
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	3	4	4	4	3	3	2	2	0	2	0							
2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	2	2	2	2	2	4	4	4	4	4	4	4	3	3	2	2	0	2	0						
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	4	3	3	2	2	0	2	0					
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	3	4	4	5	5	5	5	5	5	5	4	4	3	4	2	3	0	2	0				
4	4	5	5	5	5	5	5	5	5	5	5	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	4	5	5	5	5	5	5	5	5	5	5	5	4	4	3	3	0	2	0			
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	6	5	6	6	6	6	6	6	6	5	5	5	5	4	4	3	3	0	2	0		
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	5	6	6	6	6	7	6	7	6	7	6	6	6	6	5	5	4	4	3	3	0	2	0	
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	6	7	6	7	7	7	7	8	7	7	7	7	7	7	6	6	6	6	5	5	3	3	0	2	0

Table 9.3: MATLAB demo on the loss of orthogonality among Lanczos vectors: Lanczos with partial reorthogonalization. `round(log10(abs(I-W_50)/eps))`

Let $\hat{Q}_j := [Q_1, Q_2, \ldots, Q_j]$ be the Krylov basis generated by Algorithm 9.5. Then, in this basis, the projection of A is the block tridiagonal matrix $\hat{T}_j$

$$\hat{Q}_j^* A \hat{Q}_j = \hat{T}_j = \begin{pmatrix} A_1 & B_1^* & & \\ B_1 & A_2 & \ddots & \\ & \ddots & \ddots & B_{j-1}^* \\ & & B_{j-1} & A_j \end{pmatrix}, \quad A_i, B_i \in \mathbb{R}^{p\times p}.$$

If matrices B_i are chosen to be upper triangular, then $\hat{T}_j$ is a band matrix with bandwidth $2p+1$!

Similarly as in scalar case, in the j-th iteration step we obtain the equation

$$A\hat{Q}_j - \hat{Q}_j\hat{T}_j = Q_{j+1}B_jE_j^* + \hat{F}_j, \qquad E_j = \begin{pmatrix} O \\ \vdots \\ O \\ I_p \end{pmatrix},$$

where $\hat{F}_j$ accounts for the effect of roundoff error. Let $(\vartheta_i, \mathbf{y}_i)$ be a Ritz pair of A in $\mathcal{K}^{jp}(Q_1)$. Then

$$\mathbf{y}_i = \hat{Q}_j\mathbf{s}_i, \qquad \hat{T}_j\mathbf{s}_i = \vartheta_i\mathbf{s}_i.$$

As before, we can consider the residual norm to study the accuracy of the Ritz pair $(\vartheta_i, \mathbf{y}_i)$ of A

$$\|A\mathbf{y}_i - \vartheta_i\mathbf{y}_i\| = \|A\hat{Q}_j\mathbf{s}_i - \vartheta_i\hat{Q}_j\mathbf{s}_i\| \approx \|Q_{j+1}B_jE_j^*\mathbf{s}_i\| = \left\| B_j \begin{pmatrix} s_{j(p-1)+1,i} \\ \vdots \\ s_{jp+1,i} \end{pmatrix} \right\|.$$

We have to compute the bottom p components of the eigenvectors $\mathbf{s}_i$ in order to test for convergence.

Similarly as in the scalar case, the mutual orthogonality of the Lanczos vectors (i.e., the columns of $\hat{Q}_j$) is lost, as soon as convergence sets in. The remedies described earlier are available: full reorthogonalization or selective orthogonalization.

9.8 External selective reorthogonalization

If many eigenvalues are to be computed with the Lanczos algorithm, it is usually advisable to execute shift-and-invert Lanczos with *varying shifts* [4].

In each new start of a Lanczos procedure, one has to prevent the algorithm from finding already computed eigenpairs. We have encountered this problem when we tried to compute multiple eigenpairs by simple vector iteration. Here, the remedy is the same as there. In the second and further runs of the Lanczos algorithm, the starting vectors are made orthogonal to the already computed eigenvectors. We know that in theory all Lanczos vectors will be orthogonal to the previously computed eigenvectors. However, because the previous eigenvectors have been computed only approximately the initial vectors are not orthogonal to the true eigenvectors. Because of this and because of floating point errors loss of orthogonality is observed. The loss of orthogonality can be monitored similarly as with partial reorthogonalization. For details see [4].

Bibliography

[1] W. E. Arnoldi, *The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Quarterly of Applied Mathematics, 9 (1951), pp. 17–29.

[2] J. K. Cullum and R. A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations*, vol. 1: Theory, Birkhäuser, Boston, 1985.

[3] G. H. Golub and J. H. Welsch, *Calculation of Gauss quadrature rules*, Math. Comp., 23 (1969), pp. 221–230.

[4] R. Grimes, J. G. Lewis, and H. Simon, *A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 228–272.

[5] N. Krylov and N. Bogoliubov, *Sur le calcul des racines de la transcendante de Fredholm les plus voisines d'une nombre donné par les méthodes des moindres carres et de l'algorithme variationel*, Izv. Akad. Naik SSSR, Leningrad, (1929), pp. 471–488.

[6] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bureau Standards, Sec. B, 45 (1950), pp. 255–282.

[7] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice Hall, Englewood Cliffs, NJ, 1980. (Republished by SIAM, Philadelphia, 1998.).

[8] H. Simon, *Analysis of the symmetric Lanczos algorithm with reorthogonalization methods*, Linear Algebra Appl., 61 (1984), pp. 101–132.

[9] ——, *The Lanczos algorithm with partial reorthogonalization*, Math. Comp., 42 (1984), pp. 115–142.