I.1.(a) Krylov Subspace Projection Methods

1 Introduction

In this lecture, we discuss iterative methods based on Krylov subspace projection for extracting a few eigenvalues and eigenvectors of a large sparse matrix. Projection techniques are the foundation of many algorithms. We will first discuss the general framework of the Rayleigh-Ritz subspace projection procedure, and then discuss the widely used Arnoldi and Lanczos methods.

2 Rayleigh-Ritz procedure

Let A be an $n \times n$ complex matrix and $\mathcal{K}$ be an m-dimensional subspace of $\mathbb{C}^n$. An orthogonal projection technique seeks an approximate eigenpair $(\tilde{\lambda}, \widetilde{u})$, with $\tilde{\lambda}$ in $\mathbb{C}$ and $\widetilde{u}$ in $\mathcal{K}$. This approximate eigenpair is obtained by imposing the following ***Galerkin condition***:

$$A\widetilde{u} - \tilde{\lambda}\widetilde{u} \perp \mathcal{K} \ , \tag{2.1}$$

or, equivalently,

$$\langle A\widetilde{u} - \tilde{\lambda}\widetilde{u}, v\rangle \ = \ 0 \ , \ \ \forall \ v \in \mathcal{K}. \tag{2.2}$$

To translate the condition (2.2) into a matrix problem, assume that an orthonormal basis $\{v_1, v_2, \ldots, v_k\}$ of $\mathcal{K}$ is available. Denote $V = (v_1, v_2, \ldots, v_k)$, and let $\widetilde{u} \ = \ Vy$. Then, the condition (2.2) becomes

$$\langle AVy - \tilde{\lambda}Vy, v_j\rangle = 0, \quad j = 1, \ldots, k.$$

Therefore, y and $\tilde{\lambda}$ must satisfy

$$B_k y \ = \ \tilde{\lambda} y, \tag{2.3}$$

where

$$B_k \ = \ V^H A V.$$

Each eigenvalue $\widetilde{\lambda}_i$ of B_k is called a ***Ritz value***, and Vy_i is called ***Ritz vector***, where y_i is the eigenvector of B_k associated with $\widetilde{\lambda}_i$. This procedure A is known as the ***Rayleigh-Ritz procedure***.

Algorithm 2.1 (Rayleigh-Ritz Procedure)**.**

1. Compute an orthonormal basis $\{v_i\}_{i=1,\ldots,k}$ of the subspace $\mathcal{K}$. Let $V = (v_1, v_2, \ldots, v_k)$.
2. Compute $B_k = V^H A V$.
3. Compute the eigenvalues of B_k and select k_0 desired ones: $\tilde{\lambda}_i, i = 1, 2, \ldots, k_0$, where $k_0 \leq k$.
4. Compute the eigenvectors $y_i, i = 1, \ldots, k_0$, of B_k associated with $\tilde{\lambda}_i, i = 1, \ldots, k_0$, and the corresponding approximate eigenvectors of A, $\widetilde{u}_i = Vy_i, i = 1, \ldots, k_0$.

The numerical solution of the small $k \times k$ eigenvalue problem in the statements 3 and 4 can be treated by standard algorithms for solving small dense eigenvalue problems. Another important note is that in the statement 4 one can replace eigenvectors by Schur vectors to get approximate Schur vectors $\widetilde{u}_i$ instead of approximate eigenvectors. Schur vectors can

be obtained in a numerically stable way and, in general, eigenvectors are more sensitive to rounding errors than are Schur vectors.

Alternatively, there is an oblique projection technique. Here we first select two subspaces $\mathcal{L}$ and $\mathcal{K}$ and then seek an approximation $\widetilde{u} \in \mathcal{K}$ and an element $\tilde{\lambda} \in \mathbb{C}$ that satisfy the ***Petrov-Galerkin condition***:

$$\langle A\widetilde{u} - \tilde{\lambda}\widetilde{u}, v\rangle = 0, \quad \forall\ v \in \mathcal{L}\ . \tag{2.4}$$

The subspace $\mathcal{K}$ will be referred to as the ***right subspace*** and $\mathcal{L}$ as the ***left subspace***. A procedure similar to the Rayleigh-Ritz procedure can be devised. Let V denote the basis for the subspace $\mathcal{K}$ and W for $\mathcal{L}$. Then, writing $\widetilde{u} = Vy$, the Petrov-Galerkin condition (2.4) yields the reduced eigenvalue problem

$$B_k y \ = \ \tilde{\lambda} C_k y,$$

where

$$B_k \ = \ W^H A V \quad \text{and} \quad C_k = W^H V.$$

If $C_k = I$, then the two bases are called ***biorthonormal***. In order for a biorthonormal pair V and W to exist the following additional assumption for $\mathcal{L}$ and $\mathcal{K}$ must hold. *For any two bases V and W of $\mathcal{K}$ and $\mathcal{L}$, respectively,*

$$\det(W^H V) \neq 0 \quad . \tag{2.5}$$

As can be seen this condition does not depend on the bases selected and is equivalent to requiring that no vector of $\mathcal{K}$ is orthogonal to $\mathcal{L}$.

3 Optimality

We now examine the Rayleigh-Ritz procedure in further detail. Let $Q = (Q_k, Q_u)$ be an n-by-n orthogonal matrix, where Q_k is n-by-k, and Q_u is n-by-$(n-k)$, and $\text{span}(Q_k) = \mathcal{K}$. Then

$$T = Q^T A Q = (Q_k, Q_u)^T A (Q_k, Q_u) = \begin{pmatrix} Q_k^T A Q_k & Q_k^T A Q_u \\ Q_u^T A Q_k & Q_u^T A Q_u \end{pmatrix} \equiv \begin{pmatrix} T_k & T_{uk} \\ T_{ku} & T_u \end{pmatrix}$$

The Ritz values and Ritz vectors are considered *optimal* approximations to the eigenvalues and eigenvectors of A from the selected subsapce $\mathcal{K} = \text{span}(Q_k)$ as justified by the following theorem.

Theorem 3.1. *The minimum of $\|AQ_k - Q_k S\|_2$ over all k-by-k S is attained by $S = T_k$, in which case, $\|AQ_k - Q_k T_k\|_2 = \|T_{ku}\|_2$.*

Proof. Let $S = T_k + Z$. We will show that $\|AQ_k - Q_k S\|_2$ is minimized when $Z = 0$. This is a consequence of

$$\begin{aligned}
(AQ_k - Q_k S)^T (AQ_k - Q_k S) & \\
&= [AQ_k - Q_k(T_k + Z)]^T [AQ_k - Q_k(T_k + Z)] \\
&= (AQ_k - Q_k T_k)^T (AQ_k - Q_k T_k) - (AQ_k - Q_k T_k)^T (Q_k Z) \\
&\qquad - (Q_k Z)^T (AQ_k - Q_k T_k) + (Q_k Z)^T (Q_k Z) \\
&= (AQ_k - Q_k T_k)^T (AQ_k - Q_k T_k) - (Q_k^T A^T Q_k - T_k^T) Z \\
&\qquad - Z^T (Q_k^T A Q_k - T_k) + Z^T Z \\
&= (AQ_k - Q_k T_k)^T (AQ_k - Q_k T_k) + Z^T Z.
\end{aligned} \tag{3.6}$$

Furthermore, it is easy to compute the minimum value

$$\|AQ_k - Q_kT_k\|_2 = \|(Q_kT_k + Q_uT_{ku}) - Q_kT_k\|_2 = \|Q_uT_{ku}\|_2 = \|T_{ku}\|_2,$$

as expected. □

More can be said from the identity (3.6): The conclusion of Theorem 3.1 remains valid if the spectral norm $\|\cdot\|_2$ is replaced by any unitarily invariant norm, including in particular the Frobenius norm.

Corollary 3.1. *Let A be a symmetric matrix, $A^T = A$, and let $T_k = V\Lambda V^T$ be the eigendecomposition of T_k. The minimum of $\|AP_k - P_kD\|$ over all n-by-k orthogonal matrices P_k subject to* $\mathrm{span}(P_k) = \mathrm{span}(Q_k)$ *and over all diagonal D is also $\|T_{ku}\|_2$ and is attained by $P_k = Q_kV$ and $D = \Lambda$.*

Proof. If we replace Q_k with Q_kU in the above proof, where U is another orthogonal matrix, then the columns of Q_k and Q_kU span the same space, and

$$\|AQ_k - Q_kS\|_2 = \|AQ_kU - Q_kSU\|_2 = \|A(Q_kU) - (Q_kU)(U^TSU)\|_2.$$

These quantities are still minimized when $S = T_k$, and by choosing $U = V$ so that U^TT_kU is diagonal. □

4 Krylov subspace

As we know, the simplest thing to do is to compute just the largest eigenvalue in absolute value, along with its eigenvector. The power method is the simplest algorithm suitable for this task. Starting with a given u_0, k iterations of the power method produce a sequence of vectors $u_0, u_1, u_2, \ldots, u_k$. It is easy to see that these vectors span a ***Krylov Subspace***:

$$\mathcal{K}_{k+1}(A, u_0) = \mathrm{span}\{u_0, u_1, u_2, \ldots, u_k\} = \mathrm{span}\{u_0, Au_0, A^2u_0, \ldots, A^ku_0\}.$$

Now, rather than taking u_k as our approximate eigenvector, it is natural to ask for the "best" approximate eigenvector in $\mathcal{K}_{k+1}(A, u_0)$. We will see that the best eigenvector (and eigenvalue) approximations from $\mathcal{K}_{k+1}(A, u_0)$ are much better than u_k alone.

Charaterization of $\mathcal{K}_{k+1}(A, u_0)$:

$$\mathcal{K}_{k+1}(A, u_0) = \{q(A)u_0 \mid q \in \mathcal{P}_k\},$$

where $\mathcal{P}_k$ is the set of all polynomial of degree less than $k + 1$.

Properties of $\mathcal{K}_{k+1}(A, u_0)$:

1. $\mathcal{K}_k(A, u_0) \subset \mathcal{K}_{k+1}(A, u_0)$.
   $A\mathcal{K}_k(A, u_0) \subset \mathcal{K}_{k+1}(A, u_0)$.
2. If $\sigma \neq 0$, $\mathcal{K}_k(A, u_0) = \mathcal{K}_k(\sigma A, u_0) = \mathcal{K}_k(A, \sigma u_0)$.
3. For any κ, $\mathcal{K}_k(A, u_0) = \mathcal{K}_k(A - \kappa I, u_0)$.
4. If W is nonsingular, $\mathcal{K}_k(W^{-1}AW, W^{-1}u_0) = W^{-1}\mathcal{K}_k(A, u_0)$.

An explicit Krylov basis $\{u_0, Au_0, A^2u_0, \ldots, A^ku_0\}$ is not suitable for numerical computing. It is extremely ill-conditioned. Therefore, our first task is to replace a Krylov basis with a better conditioned basis, say an orthonormal basis.

Theorem 4.1. *Let the columns of* $K_{j+1} = \begin{pmatrix} u_0 & Au_0 & \dots & A^j u_0 \end{pmatrix}$ *be linearly independent. Let*

$$K_{j+1} = U_{j+1} R_{j+1} \tag{4.7}$$

be the QR factorization of K_{j+1}*. Then there is a* $(j+1) \times j$ *unreduced upper Hessenberg matrix* $\widehat{H}_j$ *such that*

$$AU_j = U_{j+1} \widehat{H}_j. \tag{4.8}$$

Conversely, if U_{j+1} *is orthonormal and satisfies* (4.8)*, then*

$$span(U_{j+1}) = span\{u_0, Au_0, \dots, A^j u_0\}. \tag{4.9}$$

Proof. Partitioning the QR decomposition (4.7), we have

$$\begin{pmatrix} K_j & A^j u_0 \end{pmatrix} = \begin{pmatrix} U_j & u_{j+1} \end{pmatrix} \begin{pmatrix} R_j & r_{j+1} \\ 0 & r_{j+1,j+1} \end{pmatrix},$$

where $K_j = U_j R_j$ is the QR decomposition of K_j. Then

$$AK_j = AU_j R_j$$

or

$$AU_j = AK_j R_j^{-1} = K_{j+1} \begin{pmatrix} 0 \\ R_j^{-1} \end{pmatrix} = U_{j+1} R_{j+1} \begin{pmatrix} 0 \\ R_j^{-1} \end{pmatrix}.$$

It is easy to verify that

$$\widehat{H}_j = R_{j+1} \begin{pmatrix} 0 \\ R_j^{-1} \end{pmatrix}$$

is a $(j+1) \times j$ unreduced upper Hessenberg matrix. Therefore we complete the proof of (4.8).

Conversely, suppose that U_{j+1} satisfies (4.8), then by induction, we can prove the identity (4.9). □

We note that by partitioning,

$$\widehat{H}_j = \begin{pmatrix} H_j \\ h_{j+1,j} e_j^T \end{pmatrix},$$

the decomposition (4.8) can be written in the following form that will be useful in the sequel:

$$AU_j = U_j H_j + h_{j+1,j} u_{j+1} e_j^T. \tag{4.10}$$

We call (4.10) an ***Arnoldi decomposition*** of order j. The decomposition (4.8) is a compact form.

5 Arnoldi algorithm

The Arnoldi algorithm for finding a few eigenpairs of a general matrix A combines the Arnoldi process for building a Krylov subspace with the Raleigh-Ritz procedure.

First, by the Arnoldi decomposition (4.10), we deduce the following process to generate an orthogonormal basis $\{v_1, v_2, \dots, v_m\}$ of the Krylov subspace $\mathcal{K}_m(A, v)$:

Algorithm 5.1 (Arnoldi Process)**.**

1. $v_1 = v/\|v\|_2$
2. for $j = 1, 2, \ldots, k$
3. compute $w = Av_j$
4. for $i = 1, 2, \ldots, j$
5. $h_{ij} = v_i^T w$
6. $w = w - h_{ij} v_i$
7. end for
8. $h_{j+1,j} = \|w\|_2$
9. If $h_{j+1,j} = 0$, *stop*
10. $v_{j+1} = w_j / h_{j+1,j}$
11. endfor

REMARK **5.1.** (1) Note that the matrix A is only referenced via the matrix-vector multiplication Av_j. Therefore, it is ideal for large sparse or large dense structured matrices. Any sparsity or structure of a matrix can be exploited in the matrix-vector multiplication. (2) The main storage requirement is $(m+1)n$ for storing Arnoldi vectors $\{v_i\}$ plus the storage requirement for the matrix A in question or the required matrix-vector multiplication. (3) The primary arithmetic cost of the procedure is the cost of m matrix-vector products plus $2m^2n$ for the rest. It is common that the matrix-vector multiplication is the dominant cost. (4) The Arnoldi procedure breaks down when $h_{j+1,j} = 0$ for some j. It is easy to see that if the Arnoldi procedure breaks down at step j (i.e. $h_{j+1,j} = 0$), then $\mathcal{K}_j = \text{span}(V_j)$ is invariant subspace of A. (5) Some care must be taken to insure that the vectors v_j remain orthogonal to working accuracy in the presence of rounding error. The usual technique is ***reorthogonalization***.

Denote

$$V_k = \begin{pmatrix} v_1 & v_2 & \ldots & v_k \end{pmatrix}$$

and

$$H_k = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1,k-1} & h_{1k} \\ h_{21} & h_{22} & \cdots & h_{2,k-1} & h_{2k} \\ & h_{32} & \ddots & h_{3,k-1} & h_{3k} \\ & & \ddots & \vdots & \vdots \\ & & & h_{k,k-1} & h_{kk} \end{pmatrix}.$$

In the matrix form, the Arnoldi process can be expressed in the following governing relations:

$$AV_k = V_k H_k + h_{k+1,k} v_{k+1} e_k^T \tag{5.11}$$

and

$$V_k^H V_k = I \quad \text{and} \quad V_k^H v_{k+1} = 0.$$

Note that the decomposition is uniquely determined by the starting vector v. This is commonly known as the implicit Q-Theorem.

Since $V_k^H v_{k+1} = 0$, we have

$$H_k = V_k^T A V_k.$$

Thus, H_k is called the ***generalized Rayleigh Quotient Matrix*** corresponding to V_k. Let μ be an eigenvalue of H_k and y be a corresponding eigenvector y, i.e.,

$$H_k y = \mu y, \quad \|y\|_2 = 1.$$

Then the corresponding Ritz pair is $(\mu, V_k y)$. Applying y to the right hand side of (5.11), the residual vector for $(\mu, V_k y)$ is given by

$$A(V_k y) - \mu(V_k y) = h_{k+1,k} v_{k+1} (e_k^T y).$$

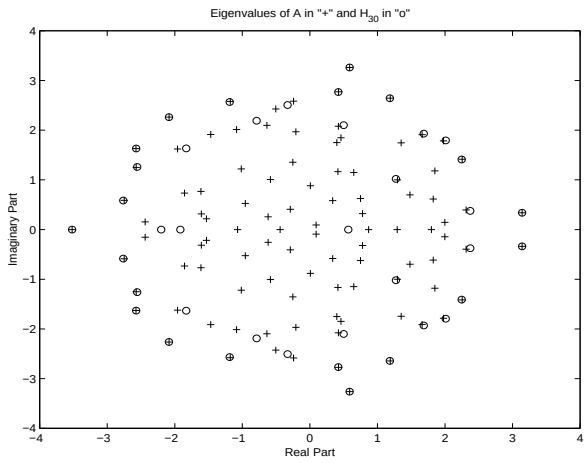
Using the backward error interpretation, we know that $(\mu, V_k y)$ is an exact eigenpair of $A+E$, where $\|E\|_2 = |h_{k+1,k}| \cdot |e_k^T y|$. This gives us a criterion of whether to accept the Ritz pair $(\mu, V_k y)$ as an accurate approximate eigenpair of A.[1]

An outline of Arnoldi's method for the nonsymmetric eigenvalue problem is as follows.

Algorithm 5.2 (Arnoldi algorithm)**.**

1. Choose a starting vector v;
2. Generate the Arnoldi decomposition of length k by the Arnoldi process;
3. Compute the Ritz pairs and decide which ones are acceptable;
4. If necessary, increase k and repeat.

Example 5.1. We apply the above Arnoldi algorithm to a 100-by-100 random sparse matrix A with approximately 1000 normally distributed nonzero entries: $A =$ `sprandn(100,100,0.1)`. All entries of the starting vector v are 1. The following figure illustrates typical convergence behavior of the Arnoldi algorithm for computing the eigenvalues of such kind of random matrix A. In this figure, "+" are the eigenvalues of matrix A (computed by `eig(full(A))`), and the "$\circ$" are the eigenvalues of the upper Hessenberg matrix H_{30} (computed by `eig(`H_{30}`)`).



In this example, we observe that *exterior eigenvalues converge first*. This is the typical convergence phenomenon of the Arnoldi algorithm (in fact, all Krylov subspace based methods). There is a general theory for the convergence analysis of the Arnoldi algorithm.

Restarting and Deflation. The algorithm has two nice aspects:

1. H_k is already in the Hessenberg form, so we can immediately apply the QR algorithm to find its eigenvalues.
2. After we increase k to, say $k+p$, we only have to orthogonalize p vectors to compute the $(k+p)$th Arnoldi decomposition. The work already completed previously is not thrown away.

Unfortunately, the algorithm has its drawbacks, too:

[1]Note that because of non-symmetry of A, we generally do not have the nice forward error estimation as the Lanczos algorithm for symmetric eigenproblem. But a similar error bound involving the condition number of the corresponding eigenvalue exists.

1. If A is large, we cannot increase k indefinitely, since V_k requires nk memory locations to store.

2. We have little control over which eigenpairs the algorithm finds. In a given application we will be interested in a certain set of eigenpairs. For example, eigenvalues lying near the imaginary axis. There is nothing in the algorithm to force desired eigenvectors into the subspace or the discard undesired ones.

We will now show how to implicitly restart the algorithm with a new Arnoldi decomposition in which (in exact arithmetic) the unwanted eigenvalues have been purged from H_k.

Implicit Restarting. We begin by asking how to cast an undesired eigenvalue out of an unreduced Hessenberg matrix H. Let μ be the eigenvalue of H, and suppose we perform one step of the QR algorithm with shift μ. The first step is to determine an orthogonal matrix U such that

$$R = U^H(H - \mu I)$$

is upper triangular. Since $H - \mu I$ is singular, R must have a zero on its diagonal. Because H is unreduced, that zero cannot occur at a diagonal position other than the last, namely $r_{nn} = 0$. and the last row of R is zero.

Furthermore, note that $U = P_{12}P_{23}\cdots P_{n-1,n}$, where $P_{i,i+1}$ is a rotation in the $(i, i+1)$-plane. Consequently, U is Hessenberg and can be partitioned in the form

$$U = \begin{pmatrix} U_* & u \\ u_{k,k-1}e_{k-1}^T & u_{k,k} \end{pmatrix}.$$

Hence

$$H' = RU + \mu I = \begin{pmatrix} \hat{H}_* & \hat{h} \\ 0 & \mu \end{pmatrix} = U^H H U.$$

In other words, the shifted QR step has found the eigenvalue μ exactly and has deflated the problem.

In the presence of rounding error we cannot expect the last element of R to be zero. This means that the matrix $\widehat{H}$ will have the form

$$\widehat{H}' = \widehat{U}^H H \widehat{U} = \begin{pmatrix} \widehat{H}_* & \widehat{h} \\ \widehat{h}_{k,k-1}e_{k-1}^T & \widehat{\mu} \end{pmatrix}.$$

We are now going to show how to use the transformation $\widehat{U}$ to reduce the size of the Arnoldi decomposition.

By the relation (5.11), we have

$$AV_k\widehat{U} = V_k\widehat{U}(\widehat{U}^T H_k \widehat{U}) + h_{k+1,k}v_{k+1}e_k^T\widehat{U}.$$

If we partition

$$\widehat{V}_k = V_k\widehat{U} = \begin{pmatrix} \widehat{V}_{k-1} & \widehat{v}_k \end{pmatrix}$$

Then

$$A\begin{pmatrix} \widehat{V}_{k-1} & \widehat{v}_k \end{pmatrix} = \begin{pmatrix} \widehat{V}_{k-1} & \widehat{v}_k \end{pmatrix}\begin{pmatrix} \widehat{H}_* & \widehat{h} \\ \widehat{h}_{k,k-1}e_{k-1}^T & \widehat{\mu} \end{pmatrix} + h_{k+1,k}v_{k+1}\begin{pmatrix} \widehat{u}_{k,k-1}e_{k-1}^T & \widehat{u}_{k,k} \end{pmatrix}.$$

Computing the first $k-1$ columns of this partition, we get

$$A\widehat{V}_{k-1} = \widehat{V}_{k-1}\widehat{H}_* + fe_{k-1}^T, \tag{5.12}$$

where $f = \widehat{h}_{k,k-1}\widehat{v}_k + h_{k+1,k}\widehat{u}_{k,k-1}v_{k+1}$.

The matrix $\widehat{H}_*$ is Hessenberg. The vector f is orthogonal to the columns of $\widehat{V}_{k-1}$. Hence (5.12) is an Arnoldi decomposition of length $k-1$. With exact computation, the eigenvalue μ is not presented in $\widehat{H}_*$.

The process may be repeated to remove other unwanted values from H. If the matrix is real, a pair of complex conjugate eigenvalues can be removed at the same time via an implicit double shift. The key observation here is that V is zero below its second subdiagonal element, so that truncating the last two columns and adjusting the residual results in an Arnoldi decomposition can be done together. Once un-desired eigenvalues have been removed from H, the Arnoldi decomposition may be expanded again and the process repeats.

Deflation. There are two important additions to the algorithm that are beyond the scope of this lecture.

1. First, as Ritz pairs converge they can be locked into the decomposition. The procedure amounts to computing an Arnoldi decomposition of the form

$$A \begin{pmatrix} V_1 & V_2 \end{pmatrix} = \begin{pmatrix} V_1 & V_2 \end{pmatrix} \begin{pmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{pmatrix} + h_{k+1,k} v_{k+1} e_k^T$$

   When this is done, one can work with the part of decomposition corresponding to V_2, thus saving multiplications by A. (However, care must be taken to maintain orthogonality to the columns of V_1.)

2. The second addition concerns unwanted Ritz pairs. The restarting procedure will tend to purge the unwanted eigenvalues from H. But the columns of U may have significant components along the eigenvectors corresponding to the purged pairs, which will then reappear as the decomposition is expanded. If certain pair are too persistent, it is best to keep them around by computing a block diagonal decomposition of the form

$$A \begin{pmatrix} V_1 & V_2 \end{pmatrix} = \begin{pmatrix} V_1 & V_2 \end{pmatrix} \begin{pmatrix} H_{11} & 0 \\ 0 & H_{22} \end{pmatrix} + \eta v_{k+1} e_k^T,$$

   where H_{11} contains the unwanted eigenvalues. This insures that U_2 has negligible components along the unwanted eigenvectors. We can then compute an Arnoldi decomposition by reorthognalizing the relation

$$AV_2 = H_{22}V_2 + \eta v_{k+1} e_m^T,$$

   where m is the order of H_2.

6 The symmetric Lanczos algorithm

The Lanczos algorithm for finding a few eigenpairs of a symmetric matrix A combines the Lanczos process for building a Krylov subspace with the Raleigh-Ritz procedure.

Let us begin with an observation that in the Arnoldi decomposition (4.10), if A is symmetric, then the upper Hessenberg matrix H_j is symmetric tridiagonal. Therefore, we have the following simplified process to compute an orthonormal basis of a Krylov subspace:

Algorithm 6.1 (Lanczos process)**.**

```
1   q_1 = v/‖v‖_2, β_0 = 0; q_0 = 0;
2   for j = 1 to k, do
3       w = Aq_j;
```

4 $\alpha_j = q_j^T w$;
5 $w = w - \alpha_j q_j - \beta_{j-1} q_{j-1}$;
6 $\beta_j = \|w\|_2$;
7 if $\beta_j = 0$, quit;
8 $q_{j+1} = w/\beta_j$;
9 EndDo

Denote

$$Q_k = \begin{pmatrix} q_1 & q_2 & \dots & q_k \end{pmatrix}$$

and

$$T_k = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \alpha_{k-1} & \beta_{k-1} \\ & & & \beta_{k-1} & \alpha_k \end{pmatrix} = \text{tridiag}(\beta_j, \alpha_j, \beta_{j+1}),$$

the k-step Lanczos process yields a fundamental relation

$$AQ_k = Q_k T_k + f_k e_k^T, \quad f_k = \beta_k q_{k+1} \tag{6.13}$$

and $Q_k^T Q_k = I$ and $Q_k^T q_{k+1} = 0$.

Let μ be an eigenvalue of T_k and y be a corresponding eigenvector y, i.e.,

$$T_k y = \mu y, \quad \|y\|_2 = 1.$$

Apply y to the right of (6.13) to get

$$A(Q_k y) = Q_k T_k y + f_k (e_k^T y) = \mu (Q_k y) + f_k (e_k^T y). \tag{6.14}$$

Here μ is a *Ritz value*, and $Q_k y$ is the corresponding *Ritz vector*. Equation (6.14) offers a quick insight into convergence of the Ritz value and vector.

If $f_k(e_k^T y) = 0$ for some k, then the associated Ritz value μ is an eigenvalue of A with the corresponding eigenvector $Q_k y$. In general, it is unlikely that $f_k(e_k^T y) = 0$, but we hope that the residual norm $\|f_k(e_k^T y)\|_2$ may be small; and when this happens we expect that μ is going to be a good approximate to A's eigenvalue. Indeed, we have the following result.

Lemma 6.1. *Let H be (real) symmetric, and $Hz - \mu z = r$ and $z \neq 0$. Then*

$$\min_{\lambda \in \text{eig}(H)} |\lambda - \mu| \leq \|r\|_2 / \|z\|_2.$$

Proof. Let $H = U \Lambda U^T$ be the eigen-decomposition of H. Then

$$(H - \mu I) z = r \quad \Rightarrow \quad U(\Lambda - \mu I) U^T z = r \quad \Rightarrow \quad (\Lambda - \mu I)(U^T z) = U^T r.$$

Notice that $\Lambda - \mu I$ is diagonal. Thus

$$\|r\|_2 = \|U^T r\|_2 = \|(\Lambda - \mu I)(U^T z)\|_2 \geq \min_{\lambda \in \text{eig}(H)} |\lambda - \mu| \, \|U^T z\|_2 = \min_{\lambda \in \text{eig}(H)} |\lambda - \mu| \|z\|_2,$$

as expected. □

The following corollary is a consequence of above Lemma 6.1.

Corollary 6.1. *There is an eigenvalue λ of A such that*

$$|\lambda - \mu| \leq \frac{\|f_k(e_k^T y)\|_2}{\|Q_k y\|_2} = \frac{|\beta_k| \cdot |e_k^T y|}{\|Q_k y\|_2}.$$

In the absence of roundoff error, $\|Q_k y\|_2 = \|y\|_2 = 1$, and thus the denominator $\|Q_k y\|_2$ can be dropped . But numerically the loss of orthogonality in the columns of Q_k can destroy this identity. This loss of orthogonality motivated the developments of various reorthogonalization strategies.

In summary, we have the following Lanczos algorithm in the simplest form.

Algorithm 6.2 (Simple Lanczos Algorithm)**.**

1. $q_1 = v/\|v\|_2$, $\beta_0 = 0$, $q_0 = 0$;
2. for $j = 1$ to k do
3. $\quad w = Aq_j$;
4. $\quad \alpha_j = q_j^T w$;
5. $\quad w = w - \alpha_j q_j - \beta_{j-1} q_{j-1}$;
6. $\quad \beta_j = \|w\|_2$;
7. $\quad$ if $\beta_j = 0$, quit;
8. $\quad q_{j+1} = w/\beta_j$;
9. $\quad$ Compute eigenvalues and eigenvectors of T_j
10. $\quad$ Test for convergence
11. EndDo

Caveat. All the discussion so far is under the assumption of exact arithmetic. In the presence of finite precision arithmetic, the numerical behaviors of the Lanczos algorithm could be significantly different. For example, in finite precision arithmetic, the orthogonality of the computed Lanczos vectors $\{q_j\}$ is lost when j is as small as 10 or 20. The simplest remedy (and also the most expensive one) is to implement the the full reorthogonalization, namely after the step 5, do

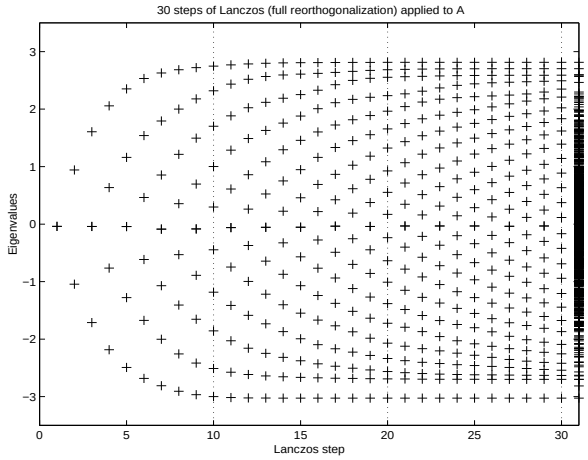
$$w = w - \sum_{i=1}^{j-1} (w^T q_i) q_i.$$

This is called the Lanczos algorithm with full reorthogonalization. (Sometimes, it may be needed to execute *twice*). A more elaborate scheme, necessary when convergence is slow and several eigenvalues are sought, is to use the selective orthogonalization.

Example 6.1. We illustrate the Lanczos algorithm by a running an example, a 1000-by-1000 diagonal matrix A, most of whose eigenvalues were chosen randomly from a normal Gaussian distribution. To make the plot easy to understand, we have also sorted the diagonal entries of A from largest to smallest, so $\lambda_i(A) = a_{ii}$ with the corresponding eigenvector e_i. There are a few extreme eigenvalues, and the rest cluster near the center of the spectrum. The starting Lanczos vector v has all equal entries.

Note that there is no loss in generality in experimenting with a diagonal matrix, since running the Lanczos algorithm on A with starting vector $q_1 = v/\|v\|_2$ is equivalent to running the Lanczos algorithm on $Q^T A Q$ with starting vector $Q^T q_1$.

The following figure illustrates convergence of the Lanczos algorithm for computing the eigenvalues of A. In this figure, the eigenvalues of each T_k are shown plotted in column k, for $k = 1$ to 30, with the eigenvalues of A plotted in an extra column at the right. Thus, column k has k "+"s, one marking each eigenvalue of T_k.



To understand convergence, consider the largest eigenvalues of each T_k, note that they increase monotonically as k increases; this is a consequence of the Cauchy interlace theorem. A completely analogous phenomenon occurs with the smallest eigenvalues.

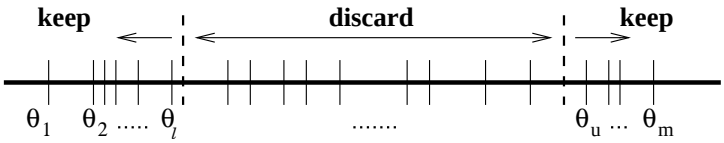
Now we can ask to which eigenvalue of A the eigenvalue $\lambda_i(T_k)$ converges as k increases. Clearly, the largest eigenvalue of T_k, $\lambda_1(T_k)$, ought to converge to the largest eigenvalue of A, $\lambda_1(A)$. Similarly, the ith largest eigenvalue $\lambda_i(T_k)$ of T_k must increase monotonically and converge to the ith largest eigenvalue $\lambda_i(A)$ of A.

In summarizing the discussion, we observe that

1. Extreme eigenvalues, i.e., the largest and smallest ones, converge first, and the interior eigenvalues converge last.

2. Convergence is monotonic, with the ith largest (smallest) eigenvalues of T_k increasing (decreasing) to the ith laregst (smallest) eigenvalue of A, provided that the Lanczos algorithm does not stop prematurely with some $\beta_k = 0$.

Thick Restarting. To reduce the cost of computing a large subspace, the iteration is restarted after a fixed number $m + 1$ of the basis vectors are computed. Instead of using the implicit restarting scheme as we discussed for the Arnoldi method, we can also use a so-called thick restarting scheme, and lead to a thick-restart Lanczos method (TRLan). The TRLan method is based on the observation that the Ritz values first converge to the exterior eigenvalues of A. At beginning, TRLan selects two indices ℓ and u to indicate those Ritz values to be kept at both ends of spectrum as showni in the following figure:



The corresponding kept Ritz vectors are denoted by

$$\widehat{Q}_k = [\widehat{q}_1, \widehat{q}_2, \ldots, \widehat{q}_k] = Q_m Y_k, \tag{6.15}$$

where

$$k = \ell + (m - u + 1), \tag{6.16}$$

$$Y_k = [y_1, y_2, \ldots, y_\ell, \, y_u, y_{u+1}, \ldots, y_m], \tag{6.17}$$

and y_i is the eigenvector of T_m corresponding to θ_i. TRLan sets these Ritz vectors $\widehat{Q}_k$ as the first k basis vectors at the restart and q_{m+1} as the $(k+1)$th basis vector (i.e., $\widehat{q}_{k+1} = q_{m+1}$).[2] To compute the $(k+2)$th basis vector $\widehat{q}_{k+2}$, TRLan computes $A\widehat{q}_{k+1}$ and orthonormalizes it against the previous $k+1$ basis vectors. That is,

$$\widehat{\beta}_{k+1}\widehat{q}_{k+2} = A\widehat{q}_{k+1} - \widehat{Q}_k(\widehat{Q}_k^H A\widehat{q}_{k+1}) - \widehat{q}_{k+1}(\widehat{q}_{k+1}^H A\widehat{q}_{k+1}). \tag{6.18}$$

Note that $A\widehat{Q}_k$ satisfies the relation:

$$A\widehat{Q}_k = \widehat{Q}_k D_k + \beta_m \widehat{q}_{k+1} s^H,$$

where D_k is the $k \times k$ diagonal matrix whose diagonal elements are the kept Ritz values, and $s = Y_k^H e_m$. Thus, the coefficients $\widehat{Q}_k^H A\widehat{q}_{k+1}$ in (6.18) can be computed efficiently:

$$\begin{aligned} \widehat{Q}_k^H A\widehat{q}_{k+1} &= (A\widehat{Q}_k)^H \widehat{q}_{k+1} = (\widehat{Q}_k D_k + \beta_m \widehat{q}_{k+1} s^H)^H \widehat{q}_{k+1} \\ &= D_k Y_k^H (Q_m^H q_{m+1}) + \beta_m s(\widehat{q}_{k+1}^H \widehat{q}_{k+1}) = \beta_m s. \end{aligned}$$

In general, at the ith iteration after the restart, the new basis vector $\widehat{q}_{k+i+1}$ satisfies the relation:

$$A\widehat{Q}_{k+i} = \widehat{Q}_{k+i}\widehat{T}_{k+i} + \widehat{\beta}_{k+i}\widehat{q}_{k+i+1}e_{k+i}^H,$$

where $\widehat{T}_{k+i} = \widehat{Q}_{k+i}^H A\widehat{Q}_{k+i}$ is of the form

$$\widehat{T}_{k+i} = \begin{pmatrix} D_k & \beta_m s & & & & \\ \beta_m s^H & \widehat{\alpha}_{k+1} & \widehat{\beta}_{k+1} & & & \\ & \widehat{\beta}_{k+1} & \widehat{\alpha}_{k+2} & \widehat{\beta}_{k+2} & & \\ & & \ddots & \ddots & \ddots & \\ & & & \widehat{\beta}_{k+i-2} & \widehat{\alpha}_{k+i-1} & \widehat{\beta}_{k+i-1} \\ & & & & \widehat{\beta}_{k+i-1} & \widehat{\alpha}_{k+i} \end{pmatrix}.$$

Note that the three-term recurrence is not valid only for computing the $(k+2)$th basis vector and is resumed afterward.

7 Convergence analysis of the symmetric Lanczos algorithm

In this subsection, we will justify the convergence observation in theory in the absence of roundoff errors. Throughout this subsection

$$\mathcal{K}_k \equiv \mathcal{K}_k(A, v).$$

An error bound in term of distance between $\mathcal{K}_k$ and an eigenvector. First, let us examine a special case.

Lemma 7.1. *Let $Ax = \lambda x$ and $x \neq 0$. If $x \in \mathcal{K}_k$, then $\lambda \in \mathrm{eig}(T_k)$.*

Proof. Since $x \in \mathcal{K}_k$, there is a y such that $x = Q_k y$. Apply y to the right of (6.13) to get

$$\lambda Q_k y = \lambda x = Ax = AQ_k y = Q_k T_k y + f_k e_k^T y.$$

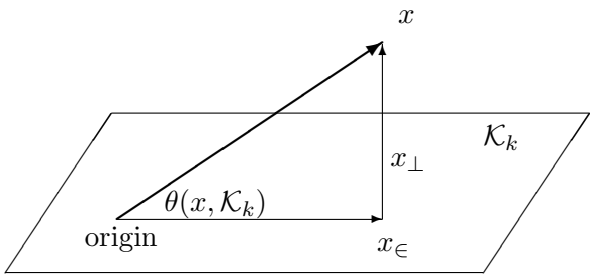
Notice that $f_k = \beta_k q_{k+1}$. Apply Q_k^T to the left of the above equation to get $\lambda y = T_k y$, as was to be shown. □

[2] The ith basis vector $\widehat{q}_i$ computed after the restart is distinguished from the ith basis vector q_i computed before the restart by the hat over it.

In general, if an eigenvector x of A is nearly in the Krylov subspace $\mathcal{K}_k$, we would expect the corresponding eigenvalue be well approximated by the process.

Theorem 7.1. *Let $Ax = \lambda x$ and $\|x\|_2 = 1$. Then T_k has an eigenvalue μ such that*

$$|\lambda - \mu| \le \|A\|_2 \tan\theta(x, \mathcal{K}_k).$$



Proof. Write $x = x_\in + x_\perp$, where $x_\in = Q_k Q_k^T x \in \mathcal{K}_k$ and $x_\perp = (I - Q_k Q_k^T)x \perp \mathcal{K}_k$. We have

$$\|x_\in\|_2 = \cos\theta(x, \mathcal{K}_k), \quad \|x_\perp\|_2 = \sin\theta(x, \mathcal{K}_k).$$

By $\lambda x = Ax$, we have

$$\lambda x_\in + \lambda x_\perp = \lambda x = Ax = Ax_\in + Ax_\perp = AQ_kQ_k^T x + Ax_\perp = Q_kT_kQ_k^T x + f_k e_k^T Q_k^T x + Ax_\perp,$$

multiplying which by Q_k^T from left yields

$$\lambda Q_k^T x_\in = T_k Q_k^T x + Q_k^T A x_\perp \quad \Rightarrow \quad T_k Q_k^T x - \lambda Q_k^T x = Q_k^T A x_\perp.$$

By Lemma 6.1, we conclude that T_k has an eigenvalue μ such that

$$|\lambda - \mu| \le \|Q_k^T A x_\perp\|_2 / \|Q_k^T x\|_2 \le \|Q_k^T\|_2 \|A\|_2 \|x_\perp\|_2 / \|x_\in\|_2 = \|A\|_2 \tan\theta(x, \mathcal{K}_k),$$

as was to be shown. □

The upper bound on the distance between $\mathcal{K}_k$ and an eigenvector of A. Let $A = U\Lambda U^T$ be the eigen-decomposition of A, where $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ and $U = (u_1, u_2, \ldots, u_n)$. Assume that

$$\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n.$$

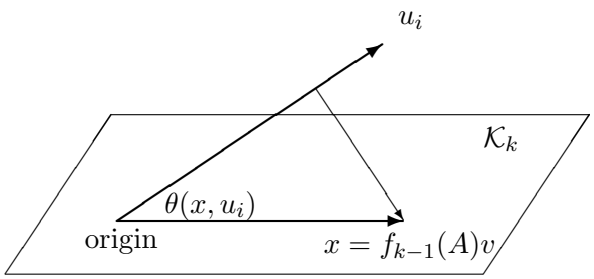
Let $P_i \stackrel{\text{def}}{=} u_i u_i^T$ which is the *spectral projector* onto $\mathrm{span}\{u_i\}$, and let $\mathbb{P}_j$ be the collection of polynomials of degree no bigger than j.

Theorem 7.2.

$$\tan\theta(u_i, \mathcal{K}_k) = \min_{f \in \mathbb{P}_{k-1}, f(\lambda_i)=1} \|f(A) y_i\|_2 \tan\theta(u_i, v),$$

where

$$y_i = \begin{cases} (I - P_i)v / \|(I - P_i)v\|_2, & \text{if } v \ne P_i v, \\ 0, & \text{otherwise.} \end{cases}$$



Proof. If $v = P_i v$, no proof is necessary since $\theta(u_i, \mathcal{K}_k) = 0$. Assume in what follows that $v \neq P_i v$.

Any $x \in \mathcal{K}_k$ takes the form $x = f_{k-1}(A)v$ for some $f_{k-1} \in \mathbb{P}_{k-1}$. Write

$$\begin{aligned} x &= P_i f_{k-1}(A)v + (I - P_i)f_{k-1}(A)v \\ &= f_{k-1}(A)P_i v + f_{k-1}(A)(I - P_i)v \\ &= f_{k-1}(\lambda_i)P_i v + f_{k-1}(A)(I - P_i)v \end{aligned}$$

since $P_i A = A P_i$ (prove it!). Set $g(t) = f_{k-1}(t)/f_{k-1}(\lambda_i)$. Thus

$$\begin{aligned} \tan\theta(u_i, x) &= \frac{\|f_{k-1}(A)(I - P_i)v\|_2}{\|f_{k-1}(\lambda_i)P_i v\|_2} \\ &= \frac{\|[f_{k-1}(A)/f_{k-1}(\lambda_i)](I - P_i)v\|_2}{\|P_i v\|_2} \\ &= \frac{\|g(A)(I - P_i)v\|_2}{\|(I - P_i)v\|_2} \frac{\|(I - P_i)v\|_2}{\|P_i v\|_2} \\ &= \|g(A)y_i\|_2 \tan\theta(u_i, v). \end{aligned}$$

Since $x \in \mathcal{K}_k$ is arbitrary, we have

$$\tan\theta(u_i, \mathcal{K}_k) = \min_{x \in \mathcal{K}_k} \tan\theta(u_i, x) = \min_{f \in \mathbb{P}_{k-1}, f(\lambda_i)=1} \|f(A)y_i\|_2 \tan\theta(u_i, v),$$

as expected. □

Theorem 7.3.

$$\tan\theta(u_i, \mathcal{K}_k) \leq \frac{\xi_i}{\mathcal{T}_{k-i}(1 + 2\delta_i)} \tan\theta(u_i, v),$$

where $\xi_1 = 1$ *and for* $i > 1$ $\xi_i = \prod_{j=1}^{i-1} \frac{\lambda_j - \lambda_n}{\lambda_j - \lambda_i}$ *and* $\delta_i = \frac{\lambda_i - \lambda_{i+1}}{\lambda_{i+1} - \lambda_n}$, *and* $\mathcal{T}_j(t)$ *is the* j*th Chebyshev polynomial (of the first kind).*

Proof. Let y_i be assigned as in Theorem 7.2. Since A's eigenvectors u_j form an orthonormal basis of the entire space, we can write

$$y_i = \sum_{j=1}^{n} \alpha_j u_j = \sum_{j \neq i} \alpha_j u_j,$$

since $y_i \perp u_i \quad \Rightarrow \quad \alpha_i = 0$. It can be proved that $\sum_{j=1, j\neq i}^{n} |\alpha_j|^2 = 1$ (prove it!). Thus

$$\|f(A)y_i\|_2^2 = \sum_{j=1, j\neq i}^{n} |f(\lambda_j)\alpha_j|^2 \leq \max_{1 \leq j \leq n, j \neq i} |f(\lambda_j)|^2 \sum_{j=1, j\neq i}^{n} |\alpha_j|^2 = \max_{1 \leq j \leq n, j \neq i} |f(\lambda_j)|^2. \quad (7.19)$$

We'd like to have a $f \in \mathbb{P}_{k-1}$ with $f(\lambda_i) = 1$ such that $\max_{1 \leq j \leq n, j \neq i} |f(\lambda_j)|^2$ is as small as possible. To this end, we again come to Chebyshev polynomials.

Notice we always order λ_i's as

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n.$$

Now for the case when $i = 1$. We choose

$$f(t) = \mathcal{T}_{k-1}\left(\frac{2t - (\lambda_2 + \lambda_n)}{\lambda_2 - \lambda_n}\right) \Big/ \mathcal{T}_{k-1}(1 + 2\delta_1),$$

for which $\max_{2\le j\le n} |f(\lambda_j)|^2 \le 1/\mathcal{T}_{k-1}(1+2\delta_1)$ and $f(\lambda_1)=1$. This, together with Theorem 7.2, conclude the proof.

In general for $i>1$, we shall consider polynomials of form

$$f(t)=\frac{(\lambda_1-t)\cdots(\lambda_{i-1}-t)}{(\lambda_1-\lambda_i)\cdots(\lambda_{i-1}-\lambda_i)}g(t), \tag{7.20}$$

and search a $g\in\mathbb{P}_{k-i}$ such that $\max_{i+1\le j\le n}|g(\lambda_j)|^2$ is as small as possible and $g(\lambda_i)=1$. To this end, we choose

$$g(t)=\mathcal{T}_{k-i}\left(\frac{2t-(\lambda_{i+1}+\lambda_n)}{\lambda_{i+1}-\lambda_n}\right)\Big/\,\mathcal{T}_{k-i}(1+2\delta_i),$$

for which $\max_{1\le j\le n, j\ne i}|g(\lambda_j)|^2\le 1/\mathcal{T}_{k-i}(1+2\delta_i)$ and $g(\lambda_i)=1$. This, together with Theorem 7.2 and (7.19) and (7.20), conclude the proof. □

The bounds in Theorem 7.3 turn to get more and more complicated as the index i gets bigger and bigger. Especially it yields nothing about approximating the last eigenvector u_n. This is in fact due to the way the theorem is derived, not the intrinsic property of Krylov subspace approximations; as a matter of fact, the Krylov subspaces favor equally well to eigenspaces corresponding to both ends of eigenvalues. This can be seen by applying Theorem 7.3 to $-A$, noticing that

$$\mathcal{K}_k(A,v)=\mathcal{K}_k(-A,v).$$

Error bounds of the Lanczos algorithm – Kaniel and Saad Theorems. We may exclude the case when some $\beta_i=0$ in the Lanczos Process and then an invariant subspace of A is found.

Notice that we have $q_1=v/\|v\|_2$ and

$$AQ_k=Q_kT_k+f_ke_k^T=Q_kT_k+\beta_kq_{k+1}e_k^T.$$

Lemma 7.2. *$q_1^TA^iq_{k+1}=0=q_1^TA^if_k$ for $i<k$. Thus $q_1^Tf(A)q_{k+1}=0=q_1^Tf(A)f_k$ for $f\in\mathbb{P}_{k-1}$.*

Proof. We have $Aq_1=\alpha_1q_1+\beta_1q_2$. By induction, one can show that

$$A^iq_1=\sum_{j=1}^{i+1}\gamma_jq_j\quad\text{for } i\le k.$$

Thus $q_{k+1}^TA^iq_1=0$ when $i+1<k+1$, i.e., $i<k$. □

Lemma 7.3. *For $i\le k$, we have*

$$A^iQ_k=Q_kT_k^i+\sum_{j=0}^{i-1}A^jf_kc_j^T=Q_kT_k^i+\sum_{j=0}^{i-1}\beta_kA^jq_{k+1}c_j^T,$$

where c_j's are some vectors. Thus $q_1^TA^iQ_k=e_1^TT_k^i$ for $i\le k$. Therefore for $f\in\mathbb{P}_{k-1}$, we have

$$q_1^Tf(A)Q_k=e_1^Tf(T_k).$$

Proof. By induction. □

Denote T_k's eigenvalues by $\mu_1\ge\mu_2\ge\cdots\ge\mu_k$, with corresponding (orthonormal) eigenvectors $y_1,y_2,\ldots,y_k$.

Lemma 7.4. *$e_1^T y_j \neq 0 \neq e_k^T y_j$ for all j, i.e., y_j's first and last components are not zeros.*

Proof. Notice that we assumed $\beta_i \neq 0$. Since $(T_k - \mu_j I)y_j = 0$, if to the contrary, $e_1^T y_j = 0$ then the first equation yields

$$\beta_1 e_2^T y_j = 0 \quad \Rightarrow \quad e_2^T y_j = 0;$$

continuing this argument to get $y_j = 0$, that's impossible since y_j is an eigenvector. Similarly one can show $e_k^T y_j \neq 0$. □

By the Courant-Fisher Minimax Theorem, we have

$$\mu_1 = \max_{0\neq y\in\mathbb{R}^k} \frac{y^T T_k y}{y^T y} = \max_{0\neq y\in\mathbb{R}^k} \frac{y^T Q_k^T A Q_k y}{y^T Q_k^T Q_k y} = \max_{f\in\mathbb{P}_{k-1}, f(A)v\neq 0} \frac{v^T f(A) A f(A) v}{v^T f(A)^2 v}.$$

In general, the Cauchy-Fisher Min-Max Principle yields

$$\begin{aligned}
\mu_i &= \max_{y\perp \operatorname{span}\{y_1,y_2,\ldots,y_{j-1}\}} \frac{y^T T_k y}{y^T y} \\
&= \max_{Q_k y\perp \operatorname{span}\{Q_k y_1, Q_k y_2,\ldots,Q_k y_{j-1}\}} \frac{y^T Q_k^T A Q_k y}{y^T Q_k^T Q_k y} \\
&= \max_{\substack{f\in\mathbb{P}_{k-1}, f(A)v\neq 0 \\ f(A)v\perp \operatorname{span}\{Q_k y_1, Q_k y_2,\ldots,Q_k y_{j-1}\}}} \frac{v^T f(A) A f(A) v}{v^T f(A)^2 v}.
\end{aligned}$$

Lemma 7.5. *Let $f \in \mathbb{P}_{k-1}$. Then $f(A)v \perp Q_k y_j$ if and only if $f(\mu_j) = 0$.*

Proof. Notice that $q_1 = v/\|v\|_2$. We have by Lemma 7.3

$$[f(A)v]^T Q_k y_j = v^T f(A) Q_k y_j = \|v\|_2 q_1^T f(A) Q_k y_j = \|v\|_2 e_1^T f(T_k) y_j = \|v\|_2 f(\mu_j) e_1^T y_j,$$

from which and Lemma 7.4, the conclusion follows. □

With this lemma, we have

$$\mu_i = \max_{\substack{f\in\mathbb{P}_{k-1}, f(A)v\neq 0 \\ f(\mu_1)=\cdots=f(\mu_{j-1})=0}} \frac{v^T f(A) A f(A) v}{v^T f(A)^2 v}.$$

Now we are ready for the main theorem of this section.

Theorem 7.4.

$$0 \leq \lambda_i - \mu_i \leq (\lambda_i - \lambda_n) \left[\frac{\zeta_i}{\mathcal{T}_{k-i}(1+2\delta_i)} \tan\theta(u_i, v) \right]^2,$$

where $\zeta_1 = 1$ and for $i > 1$, $\zeta_i = \max_{i+1\leq \ell\leq n} \prod_{j=1}^{i-1} \left| \frac{\mu_j - \lambda_\ell}{\mu_j - \lambda_i} \right|$.

Proof. We first show that $0 \leq \lambda_i - \mu_i$. By Courant-Fischer Max-min Theorem, we have

$$\begin{aligned}
\mu_i &= \max_{\dim(\mathcal{S})=i} \min_{0\neq y\in\mathcal{S}} \frac{y^T T_k y}{y^T y} = \max_{\dim(\mathcal{S})=i} \min_{0\neq y\in\mathcal{S}} \frac{y^T Q_k^T A Q_k y}{y^T Q_k^T Q_k y} \\
&= \max_{\dim(Q_k\mathcal{S})=i} \min_{0\neq Q_k y\in Q_k\mathcal{S}} \frac{y^T Q_k^T A Q_k y}{y^T Q_k^T Q_k y} \leq \max_{\dim(\mathcal{S}')=i} \min_{0\neq x\in\mathcal{S}'} \frac{x^T A x}{x^T x} = \lambda_i,
\end{aligned}$$

as expected. We now prove the other bound. We have

$$\begin{aligned}\lambda_i - \mu_i &= \frac{v^T f(A)\lambda_i f(A)v}{v^T f(A)^2 v} - \max_{\substack{f\in\mathbb{P}_{k-1}, f(A)v\neq 0\\ f(\mu_1)=\cdots=f(\mu_{j-1})=0}} \frac{v^T f(A)Af(A)v}{v^T f(A)^2 v}\\ &= \min_{\substack{f\in\mathbb{P}_{k-1}, f(A)v\neq 0\\ f(\mu_1)=\cdots=f(\mu_{i-1})=0}} \frac{v^T f(A)(\lambda_i I - A)f(A)v}{v^T f(A)^2 v}.\end{aligned}$$

Expand $v = \sum_{j=1}^{n} \gamma_j u_j$. Then

$$\begin{aligned}v^T f(A)^2 v &= \sum_{j=1}^{n} |\gamma_j|^2 |f(\lambda_j)|^2\\ &\geq |\gamma_i|^2 |f(\lambda_i)|^2,\\ v^T f(A)(\lambda_i I - A)f(A)v &= \sum_{j=1}^{n} |\gamma_j|^2 |f(\lambda_j)|^2(\lambda_i - \lambda_j)\\ &= \sum_{j=1}^{i-1} |\gamma_j|^2 |f(\lambda_j)|^2(\lambda_i - \lambda_j) + \sum_{j=i+1}^{n} |\gamma_j|^2 |f(\lambda_j)|^2(\lambda_i - \lambda_j)\\ &\leq \sum_{j=i+1}^{n} |\gamma_j|^2 |f(\lambda_j)|^2(\lambda_i - \lambda_j)\\ &\leq (\lambda_i - \lambda_n) \sum_{j=i+1}^{n} |\gamma_j|^2 |f(\lambda_j)|^2.\end{aligned}$$

Thus we have

$$\begin{aligned}0 \leq \lambda_i - \mu_i &\leq (\lambda_i - \lambda_n) \min_{\substack{f\in\mathbb{P}_{k-1}\\ f(\mu_1)=\cdots=f(\mu_{i-1})=0}} \frac{\sum_{j=i+1}^{n} |\gamma_j|^2 |f(\lambda_j)|^2}{|\gamma_i|^2 |f(\lambda_i)|^2}\\ &\leq (\lambda_i - \lambda_n) \min_{\substack{g\in\mathbb{P}_{k-1}, g(\lambda_i)=1\\ g(\mu_1)=\cdots=g(\mu_{i-1})=0}} \max_{i+1\leq j\leq n} |g(\lambda_j)|^2 [\tan\theta(u_i, v)]^2,\end{aligned}$$

where $g(t) = f(t)/f(\lambda_i)$. In the case when $i = 1$,

$$0 \leq \lambda_1 - \mu_1 \leq \min_{g\in\mathbb{P}_{k-1}, g(\lambda_i)=1} \max_{2\leq j\leq n} |g(\lambda_j)|^2 [\tan\theta(u_i, v)]^2,$$

a standard practice with Chebyshev polynomial completes the proof. For $i > 1$, the $g \in \mathbb{P}_{k-1}$ with $g(\mu_1) = \cdots = g(\mu_{i-1}) = 0$ and $g(\lambda_i) = 1$ takes the form

$$g(t) = \frac{(\mu_1 - t)\cdots(\mu_{i-1} - t)}{(\mu_1 - \lambda_i)\cdots(\mu_{i-1} - \lambda_i)} h(t),$$

where $h \in \mathbb{P}_{k-i}$ with $h(\lambda_i) = 1$. We have

$$\max_{i+1\leq j\leq n} |g(\lambda_j)|^2 \leq \zeta_i^2 \max_{i+1\leq j\leq n} |h(\lambda_j)|^2.$$

Hence

$$0 \le \lambda_i - \mu_i \le (\lambda_i - \lambda_n)\zeta_i^2 \min_{h\in\mathbb{P}_{k-i}, h(\lambda_i)=1} \max_{i+1\le j\le n} |h(\lambda_j)|^2 [\tan\theta(u_i, v)]^2.$$

Again by the standard practice with Chebyshev polynomial, the proof is completed. □

Exercises

7.1. Apply Theorem 7.3 to $-A$, and derive corresponding bounds.

7.2. Prove Lemma 7.3.

8 The nonsymmetric Lanczos method

The nonsymmetric Lanczos method is an oblique projection method for solving the non-Hermitian eigenvalue problem,

$$Ax = \lambda x \quad \text{and} \quad y^H A = \lambda y^H, \tag{8.21}$$

where A is a nonsymmetric matrix.

With two starting vectors q_1 and p_1, the Lanczos method builds a pair of biorthogonal bases for the Krylov subspaces $\mathcal{K}^j(A, q_1)$ and $\mathcal{K}^j(A^H, p_1)$, provided that the matrix-vector multiplications Az and $A^H z$ for an arbitrary vector z are available. The inner loop of Lanczos method uses two three-term recurrences. Therefore, it is cheaper in terms of memory requirements and references, compared to the Arnoldi method. The Lanczos method provides approximations for both right and left eigenvectors. When estimating errors and condition numbers of the computed eigenpairs, it is crucial that both the left and right eigenvectors are available. However, there are risks of breakdown and numerical instability with the method. In this section, we will start with the basic Lanczos method and its properties, and then present a number of numerical schemes for improving numerical stability and accuracy of the method.

Algorithm. The nonsymmetric Lanczos method as presented in Algorithm 8.1 is a two-sided iterative algorithm with starting vectors p_1 and q_1. It can be viewed as biorthogonalizing, via a two-sided Gram-Schmidt procedure, two Krylov sequences $\{q_1, Aq_1, A^2q_1, \ldots\}$ and $\{p_1, A^H p_1, (A^H)^2 p_1, \ldots, \}$. The two sequences of vectors $\{q_i\}$ and $\{p_i\}$ are generated using the three-term recurrences:

$$\beta_{j+1} q_{j+1} = Aq_j - \alpha_j q_j - \gamma_j q_{j-1}, \tag{8.22}$$

$$\bar{\gamma}_{j+1} p_{j+1} = A^H p_j - \bar{\alpha}_j p_j - \bar{\beta}_j p_{j-1}. \tag{8.23}$$

The vectors $\{q_i\}$ and $\{p_i\}$ are called ***Lanczos vectors***, which are the bases of $\mathcal{K}^j(A, q_1)$ and $\mathcal{K}^j(A^H, p_1)$, respectively, and are biorthogonal, namely, $p_k^H q_\ell = 0$ if $k \ne \ell$ and $p_k^H q_\ell = 1$ if $k = \ell$. In matrix notation, at the jth step, the Lanczos method generates two $n \times j$ matrices Q_j and P_j: $Q_j = (q_1, q_2, \ldots, q_j)$, $P_j = (p_1, p_2, \ldots, p_j)$ and $T_j = \text{tridiag}(\beta_j, \alpha_j, \gamma_{j+1})$, and they satisfy the governing relations:

$$AQ_j = Q_j T_j + \beta_{j+1} q_{j+1} e_j^H , \tag{8.24}$$

$$A^H P_j = P_j T_j^H + \bar{\gamma}_{j+1} p_{j+1} e_j^H , \tag{8.25}$$

$$P_j^H Q_j = I_j . \tag{8.26}$$

In addition, $P_j^H q_{j+1} = 0$ and $p_{j+1}^H Q_j = 0$. The relation (8.26) shows that the Lanczos vectors (bases) are bi-orthonormal. But note that none of Q_j and P_j is unitary. In the Lanczos bases, the matrix A is represented by a non-Hermitian tridiagonal matrix,

$$P_j^H A Q_j = T_j. \tag{8.27}$$

At any step j, we may compute eigensolutions of T_j,

$$T_j z_i^{(j)} = \theta_i^{(j)} z_i^{(j)} \quad \text{and} \quad (w_i^{(j)})^H T_j = \theta_i^{(j)} (w_i^{(j)})^H \ .$$

Eigenvalues of A are approximated by the eigenvalues $\theta_i^{(j)}$ of T_j, which are called ***Ritz values***. To each Ritz value $\theta_i^{(j)}$ the corresponds right and left ***Ritz vectors*** are

$$x_i^{(j)} = Q_j z_i^{(j)} \quad \text{and} \quad y_i^{(j)} = P_j w_i^{(j)} \ . \tag{8.28}$$

The approximations of Ritz values and vectors to the eigenvalues and eigenvectors of the original matrix A can be estimated by the norms of the residuals:

$$r_i^{(j)} = A x_i^{(j)} - \theta_i^{(j)} x_i^{(j)}, \tag{8.29}$$

$$(s_i^{(j)})^H = (y_i^{(j)})^H A - \theta_i^{(j)} (y_i^{(j)})^H. \tag{8.30}$$

Moreover, by (8.24), the right residual vector becomes

$$r_i^{(j)} = \beta_{j+1} q_{j+1} (e_j^H z_i^{(j)}) \ . \tag{8.31}$$

and by (8.25), the left residual vector becomes

$$(s_i^{(j)})^H = \gamma_{j+1} p_{j+1}^H ((w_i^{(j)})^H e_j) \ . \tag{8.32}$$

Therefore, as in the Hermitian case, the residual norms are available without explicitly computing the Ritz vectors $x_i^{(j)}$ and $y_i^{(j)}$.

A Ritz value $\theta_i^{(j)}$ is considered to be convergent if both residual norms are small. There are more discussions on convergence.

Algorithm 8.1. *Nonsymmetric Lanczos Method*

(1) choose starting vectors q_1 and p_1 such that $p_1^T q_1 = 1$
(2) $r = A q_1$;
(3) $s = A^H p_1$;
(4) **for** $j = 1, 2, \ldots,$ **until** convergence,
(5) $\alpha_j = p_j^H r$;
(6) $r = r - q_j \alpha_j$;
(7) $s = s - p_j \bar{\alpha}_j$;
(8) **if** ($\|r\| = 0$ and/or $\|s\| = 0$), **stop**;
(9) $\omega_j = r^H s$;
(10) **if** ($\omega_j = 0$), **stop**;
(11) $\beta_{j+1} = |\omega_j|^{1/2}$;
(12) $\gamma_{j+1} = \bar{\omega}_j / \beta_{j+1}$;
(13) $q_{j+1} = r / \beta_{j+1}$;
(14) $p_{j+1} = s / \overline{\gamma}_{j+1}$;
(15) compute eigentriplets $(\theta_i^{(j)}, z_i^{(j)}, w_i^{(j)})$ of T_j;
(16) test for convergence;
(17) re-biorthogonalize if necessary;
(18) $r = A q_{j+1}$;
(19) $s = A^H p_{j+1}$;
(20) $r = r - q_j \gamma_{j+1}$;
(21) $s = s - p_j \overline{\beta}_{j+1}$;
(22) **end for**
(23) compute approximate eigenvectors $x_i^{(j)}$ and $y_i^{(j)}$.

We now comment on some of the steps in Algorithm 8.1:

(1) The initial starting vectors p_1 and q_1 are best selected by the user to embody any available knowledge concerning A's wanted eigenvectors. In the absence of such knowledge, one may simply choose q_1 with randomly distributed entries and let $p_1 = q_1$.

(2), (3) and (18), (19) The matrix-vector multiplication routines for Az and $A^H z$ of an arbitrary vector z must be provided in these steps. These are usually the computational bottleneck. See the discussion of convergence properties below for implementation notes in the shift-invert case.

(8) This is one of two cases where the method breaks down. In fact, this is a good breakdown. Say if r is null, then the Lanczos vectors $\{q_1, q_2, \ldots, q_j\}$ span a (right) invariant subspace of A. All eigenvalues of T_j are the eigenvalues of A. One can either exit the algorithm or continue the algorithm by taking the vector q_{j+1} to be any vector orthogonal to the left Lanczos vectors $\{p_1, p_2, \ldots, p_j\}$ and set $\beta_{j+1} = 0$. Similar treatment can be done when s is null or both r and s are null. Therefore, this case should not really be regarded as a "failure" of the algorithm. It merely gives us freedom of choices.

In practice, an exact null vector is rare. It might happen that the norms of r and/or s are tiny. A proper tolerance value for the detection of nearly null vector should be given. A default tolerance value is ϵ, the machine precision.

(10) If $\omega_j = r^H s = 0$ before either r or s vanishes, the method breaks down completely. In most cases we may continue finding new vectors in the Krylov subspaces $\mathcal{K}^{j+k}(A, r)$ and $\mathcal{K}^{j+k}(A^H, s)$ for some integer $k > 0$, and add a block outside the three diagonals of T_j; this so-called a ***lookahead*** procedure as implemented in QMRPACK. If, however our starting vectors q_1 and p_1 have different minimal polynomials, even this does not help, and we have a ***mismatch***, also called an ***incurable breakdown***.

In practice, the exact breakdown is rare. A near breakdown occurs more often, i.e., ω_j is non-zero but extremely small in absolute value. Near breakdowns cause ***stagnation***. Any criterion for detecting a near breakdown stops too early in certain applications and too late in others. A reasonable compromise criterion for detecting near breakdowns in an eigenvalue problem is to stop if $|\omega_j| \leq \sqrt{\epsilon}\|r\|_2\|s\|_2$.

(11) – (14) Several different scalings of the vectors q_j and p_j have been proposed. Here q_j and p_j are scaled such that $p_j^H q_j = 1$ for all j. Specifically, β_j and γ_j are given equal absolute values, and the sub-diagonals β_j are taken real and positive. The choice $\beta_j = 1$ avoids scaling the p_j vectors, but unfortunately is highly susceptible to overflow. The choice of $\beta_j = \gamma_j = \sqrt{\omega_j}$ leads to a complex symmetric tridiagonal matrix T_j. The choice of scaling such that the vectors q_j and p_j have unity norms is also a popular alternative.

The condition numbers of the Lanczos vectors Q_j and P_j can be monitored by the scalars $\{\omega_i\}$. In fact, it can be shown that

$$\mathrm{cond}(Q_j) = \|Q_j\|_2\|Q_j^\dagger\|_2 \leq \sum_{i=1}^{j} \omega_i^{-1},$$

where $Q_j^\dagger$ denotes the generalized inverse of Q_j. The bound also applies to $\mathrm{cond}(P_j)$.

(15) For each step j, the eigen-decomposition of the tridiagonal matrix T_j (8.27) is computed, which is potentially very costly when j gets large. A simple way to reduce the cost is to solve the eigen-problem periodically, say every 10 steps.

One may use the general QR method to compute the eigensolution of T_j. If the scaling is chosen so that T_j is a complex symmetric tridiagonal matrix, a QL algorithm is available that exploits this special structure. However, due to the loss of unitary transformation in the QL algorithm, care must be taken to monitor and maintain numerical stability;

(16) Computation halts once bases Q_j and P_j have been determined so that eigenvalues $\theta_i^{(j)}$ of T_j (8.27) approximate all the desired eigenvalues of A with small residuals, which are calculated according to the equations (8.31) and (8.32). There are more discussions on convergence properties later.

If there is no re-biorthogonalization, then in finite precision arithmetic after a Ritz value converges to an eigenvalue of A, copies of this Ritz value may appear at later Lanczos steps. For example, a cluster of Ritz values of the reduced tridiagonal matrix, T_j, may approximate a single eigenvalue of the original matrix A. A "spurious" value is a simple Ritz value that is also an eigenvalue of the matrix of order $j-1$ obtained by deleting the first row and column from T_j. Such spurious value should be discarded from consideration. Eigenvalues of T_j which are not "spurious" are accepted as approximations to eigenvalues of the original matrix A and are tested for convergence. It is called the ***identification test***.

(17) As in the Hermitian case, in the presence of finite precision arithmetic, the computed Lanczos vectors $\{q_i\}$ and $\{p_i\}$ starts to lose the biorthogonality quickly. One may use the two-sided modified Gram-Schmidt process to re-biorthogonalize them. Specifically, the following loop may be applied at this step:

for $i = 1, 2, \ldots, j$
  $q_{j+1} = q_{j+1} - q_i(p_i^H q_{j+1})$;
  $p_{j+1} = p_{j+1} - p_i(q_i^H p_{j+1})$;
end for

This is called the ***full re-biorthogonalization***. However, it is in general very costly in terms of memory references and flops, and becomes a computational bottleneck.

(23) The approximate eigenvectors of the original matrix A are only calculated after the test in step (16) indicates convergence of Ritz values $\theta_i^{(j)}$ to the desired eigenvalues of A. The bases Q_j and P_j are used to get the approximate eigenvectors $x_i^{(j)} = Q_j z_i^{(j)}$ and $y_i^{(j)} = P_j w_i^{(j)}$ for each i that is flagged as converged.

The residuals (8.29) and (8.30) should be checked again using $x_i^{(j)}$ and $y_i^{(j)}$. This is the actual residual norms. Note that the actual norms can be much bigger than the estimated ones computed at step (16).

It is adviced to compute approximate eigenvectors $x_i^{(j)}$ and $y_i^{(j)}$ using the computed Lanczos vectors Q_j and P_j only if a certain level of biorthogonality is enforced in the implementation of the algorithm (see step (17)).

Convergence properties. A theory of convergence can be based on the theory of polynomials as in the Hermitian case, but with two additional complications. First the eigenvalues may be complex, and the Ritz values $\theta_i^{(j)}$ do not necessarily move monotonously out towards the ends of the spectrum with increasing j. Second A may be *defective* (possessing an eigenvalue whose algebraic multiplicity is strictly greater than its geometric multiplicity). Assuming exact computation, the tridiagonal matrix T_j is also defective after a sufficient number of steps j.

In floating point computation with machine precision ϵ, an eigenvalue of multiplicity m is perturbed by up to $O(\epsilon^{1/m})$, which is much larger than ϵ for $m > 1$. Barring these complications, eigenvalues that are peripheral in the spectrum, seen as a set in the complex plane, converge first.

An inherent difficulty of a non-Hermitian eigen-problem is that there is no practically computable bound on the distance from $\theta_i^{(j)}$ to an eigenvalue of A. It is possible though to bound the distance, $\|F\|$, to the nearest matrix, $A - F$, with eigen-triplet $(\theta_i^{(j)}, x_i^{(j)}, y_i^{(j)})$. Approximate information about both left and right eigenvectors has many uses, such as revealing the conditioning (or sensitivity) of an eigenvalue. Recall that residual vectors $r_i^{(j)}$ and $s_i^{(j)}$ as defined in (8.29) and (8.30). Next observe that the biorthogonality condition (8.26) implies that $(s_i^{(j)})^H x_i^{(j)} = (y_i^{(j)})^H r_i^{(j)} = 0$. Together these relations imply that

$$F = \frac{r_i^{(j)}(x_i^{(j)})^H}{\|x_i^{(j)}\|_2^2} + \frac{y_i^{(j)}(s_i^{(j)})^H}{\|y_i^{(j)}\|_2^2} \tag{8.33}$$

is the desired perturbation such that

$$(A - F)x_i^{(j)} = x_i^{(j)}\theta_i^{(j)}, \tag{8.34}$$

$$(y_i^{(j)})^H(A - F) = \theta_i^{(j)}(y_i^{(j)})^H, \tag{8.35}$$

and $\|F\|_{\mathrm{F}}^2 = \|r_i^{(j)}\|_2^2/\|x_i^{(j)}\|_2^2 + \|s_i^{(j)}\|_2^2/\|y_i^{(j)}\|_2^2$. Using the standard perturbation analysis, one can derive estimated error bounds on the accuracy of approximate eigenvalues and eigenvectors to the exact eigenvalues and eigenvectors of A. To this end, we should point out all these results are under the assumption of exact arithmetic. In the presence of finite precision arithmetic and the losses in biorthogonality, those results are generally optimistic.

9 Symmetric definite eigenvalue problems

Most iterative methods for solving large sparse standard eigenvalue problem, such as Lanczos method and Arnoldi method, can be modified to treat the generalized eigenvalue problems. Let us focus on the generalized symmetric definite eigenvalue problem, namely $Ax = \lambda Bx$, where A and B are symmetric, and B is positive definite.

If we can compute the *sparse* Cholesky factorization of B explicitly, then as shown in the previous section, it is equivalent solve the standard symmetric eigenvalue problem

$$Cy = \lambda y \qquad \Longleftrightarrow \qquad (L^{-1}AL^{-H})(L^H x) = \lambda(L^H x).$$

One can use the Lanczos method for C. It is straightforward.

Note that the required matrix-vector multiplication $w = Cq$ for some q in the Lanczos process can be done in the following three stages:

1. solve $L^H z_1 = q_1$ for z_1;
2. compute $z_2 = Az_1$;
3. solve $Lw = z_2$ for w.

This approach is usually very satisfactory and is a standard working practice, particularly, when B is a banded matrix with narrow bandwidth.

On the other hand it is possible to use Lanczos on $B^{-1}A$, again implicitly. This option is valuable when B cannot be factored conveniently. In this case, the Lanczos method must be reformulated.

First note that the matrix $B^{-1}A$ is symmetric with respect to B, namely,

$$\langle B^{-1}Ax, y\rangle_B = \langle x, B^{-1}Ay\rangle_B,$$

where the inner product $\langle x, y\rangle_B$ is defined as y^TBx, known as ***B-inner product***. $\|\cdot\|_B$ is the corresponding ***B-norm*** $\|x\|_B = \sqrt{x^TBx}$.

Mathematically the extension of the Lanczos process in B-inner product is immediate:

Algorithm 9.1 (Lanczos Process in B-inner product for $B^{-1}A$)**.**

1. $q_1 = v/\|v\|_B$, $\beta_0 = 0$; $q_0 = 0$;
2. for $j = 1$ to k, do
3. $\quad w = B^{-1}Aq_j$;
4. $\quad \alpha_j = \langle w, q_j\rangle_B$;
5. $\quad w = w - \alpha_j q_j - \beta_{j-1}q_{j-1}$;
6. $\quad \beta_j = \|w\|_B$;
7. $\quad$ if $\beta_j = 0$, quit;
8. $\quad q_{j+1} = w/\beta_j$;
9. EndDo

The governing relations are $B^{-1}AQ_k = Q_kT_k + \beta_k q_{k+1}e_k^T$, or

$$AQ_k = BQ_kT_k + \beta_k Bq_{k+1}e_k^T, \tag{9.36}$$

and

$$Q_k^TBQ_k = I_k, \quad Q_k^TBq_{k+1} = 0. \tag{9.37}$$

Let μ be an eigenvalue of T_k and y be the corresponding eigenvector y, i.e.,

$$T_ky = \mu y, \quad \|y\|_2 = 1.$$

Then applying y to the right hand side of (9.36), we have

$$AQ_ky = BQ_kT_ky + \beta_k Bq_{k+1}e_k^Ty = \mu BQ_ky + \beta_k Bq_{k+1}e_k^Ty.$$

The Ritz value is μ and the Ritz vector is Q_ky. The residual vector r is

$$r = AQ_ky - \mu BQ_ky = \beta_k Bq_{k+1}(e_k^Ty).$$

The norm of residual vector can be used as a stopping criterion.

We now show that the above Lanczos process can be further simplified. Note that

1. $\alpha_j = \langle w, q_j\rangle_B = q_j^TBw = q_j^TBB^{-1}Aq_j = q_j^TAq_j$, and
2. $\|w\|_B^2 = w^TBw = w^T(Aq_j - \alpha_j Bq_j - \beta_{j-1}Bq_{j-1}) = w^TAq_j$. The last equality is obtained by the fact that $w(= \beta_j q_{j+1})$ is generated to be B-orthogonal to all previous Lanczos vectors $q_1, q_2, \ldots, q_j$.

In summary, we have the following Lanczos method for the eigenvalue problem of a definite pencil $A - \lambda B$ with positive definite B.

Algorithm 9.2 (Lanczos Algorithm for a symmetric definite pair (A, B))**.**

1. choose an initial vector q_1 of B-norm unity;
2. for $j = 1$ to k, do
3. $\quad v = Aq_j$;
4. $\quad \alpha_j = q_j^Tv$;
5. $\quad w = B^{-1}v - \alpha_j q_j - \beta_{j-1}q_{j-1}$;

6. $\beta_j = \sqrt{w^T v}$;
7. if $\beta_j = 0$, **quit** (*an invariant subspace found*);
8. $q_{j+1} = w/\beta_j$;
9. Compute eigenvalues and eigenvectors of T_j ;
10. Test for convergence;
11. EndDo

Shift-and-invert spectral transformation. The above two approaches are efficient if only the exterior eigenvalues are sought. If the interesting eigenvalues are interior ones, then the common approach is to first use the shift-and-invert spectral transformation. Let σ is a shift which close to the desired eigenvalues. Then the generalized eigenvalue problem $Ax = \lambda Bx$ is equivalent to

$$\left[(A - \sigma B)^{-1} B\right] x = \frac{1}{\lambda - \sigma} x.$$

Note that the coefficient matrix $\left[(A - \sigma B)^{-1} B\right]$ is B-symmetric.

Exercises

9.1. Verify that $\langle x, y \rangle_B$ is an inner product, where B is symmetric positive definite.

10 Further reading

Krylov subspace projection methods are covered extensively in

- Z. Bai, J. Demmel, J. Dongarra, A. Ruhe and H. van der Vorst (editors). Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide. SIAM, Philadelphia, 2000.

The implicitly restarted Arnoldi method is proposed in

- D. Sorensen, Implicit application of polynomial filters in a k-step Arnoldi method, SIMAX, Vol. 13, pp.357–385, 1992.

MATLAB function `eigs` is an implementation of the implicitly restarted Arnoldi algorithm for finding a few eigenpairs. The Fortran software package is ARPACK[3]. See

- R. Lehoucq, D. C. Sorensen and C. Yang. *ARPACK User's Guide*, SIAM, 1998.

An excellent reference on symmetric Lanczos method, theory and practice, is

- B. N. Parlett, The Symmetric Eigenvalue Problem, reprinted (with some revision) by SIAM Press, 1998.

The thick restarting symmetric Lanczos algorithm is described in

- K. Wu and H. Simon, Thick-restart lanczos method for large symmetric eigenvalue problems, SIAM J. Mat. Anal. Appl., 22:602–616, 2000.

A recent development can be found in

- I. Yamazaki, Z. Bai, H. Simon, L.-W. Wang and K. Wu, Adaptive projection subspace dimension for the thick-restart Lanczos method, 2009 (submitted)[4]

[3] `http://www.caam.rice.edu/software/ARPACK`
[4] http://www.cs.ucdavis.edu/∼bai

A Fortran implementation of the nonsymmetric Lanczos procedure with look-ahead to cure breakdowns is available in QMRPACK[5]. The algorithm is described in

- R. Freund and N. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear system, Numer. Math. 60:315–339,1991.

A set of MATLAB routines for implementing the adaptive block Lanczos method can be used to implement Algorithm 8.1 by defining the blocksize as one at the initial step. Optional re-biorthogonalization schemes are available. An adaptive blocksize scheme is used for the treatment of (near) breakdown and/or multiple or closely clustered eigenvalues of interest. See

- Z. Bai, D. Day and Q. Ye, ABLE: an adaptive block Lanczos method for non-Hermitian eigenvalue problems, SIMAX, Vol.20, pp. 1060–1082, 1999.

[5] `http://www.netlib/linalg/qmrpack.tgz`