Chapter 10

Restarting Arnoldi and Lanczos algorithms

The number of iteration steps can be very high with the Arnoldi or the Lanczos algorithm. This number is, of course, not predictable. The iteration count depends on properties of the matrix, in particular the distribution of its eigenvalues, but also on the initial vectors.

High iteration counts entail a large memory requirement to store the Arnoldi/Lanczos vectors and a high amount of computation because of growing cost of the reorthogonalization.

The idea behind the implicitely restarted Arnoldi (IRA) and implicitely restarted Lanczos (IRL) algorithms is to reduce these costs by limiting the dimension of the search space. This means that the iteration is stopped after a number of steps (which is bigger than the number of desired eigenvalues), reduce the dimension of the search space without destroying the Krylov space tructure, and finally resume the Arnoldi / Lanczos iteration.

The implicitely restarted Arnoldi has first been proposed by Sorensen [6, 7]. It is implemented together with the implicitely restarted Lanczos algorithms in the software package ARPACK [3]. The ARPACK routines are the basis for the sparse matrix eigensolver `eigs` in MATLAB.

10.1 The m-step Arnoldi iteration

Algorithm 10.1 The m-step Arnoldi iteration

Let $A \in \mathbb{F}^{n\times n}$. This algorithm executes m steps of the Arnoldi algorithm.
$\mathbf{q}_1 = \mathbf{x}/\|\mathbf{x}\|; \quad \mathbf{z} = A\mathbf{q}_1; \quad \alpha_1 = \mathbf{q}_1^*\mathbf{z};$
$\mathbf{r}_1 = \mathbf{w} - \alpha_1\mathbf{q}_1; \quad Q_1 = [\mathbf{q}_1]; \quad H_1 = [\alpha_1];$
for $j = 1, \ldots, m-1$ **do**
  $\beta_j := \|\mathbf{r}_j\|; \quad \mathbf{q}_{j+1} = \mathbf{r}_j/\beta_j;$
  $Q_{j+1} := [Q_j, \mathbf{q}_{j+1}]; \quad \hat{H}_j := \begin{bmatrix} H_j \\ \beta_j \mathbf{e}_j^T \end{bmatrix} \in \mathbb{F}^{(j+1)\times j};$
  $\mathbf{z} := A\mathbf{q}_j;$
  $\mathbf{h} := Q_{j+1}^*\mathbf{z}; \quad \mathbf{r}_{j+1} := \mathbf{z} - Q_{j+1}\mathbf{h};$
  $H_{j+1} := [\hat{H}_j, \mathbf{h}];$
end for

We start with the Algorithm 10.1 that is a variant of the Arnoldi Algorithm 9.1. It

executes just m Arnoldi iteration steps. We will now show how the dimension of the search space is reduced withouth losing the information regarding the eigenvectors one is looking for.

Remark 10.1. Step 8 in Algorithm 10.1 is classical Gram–Schmidt orthogonalization. As

$$\mathbf{r}_{j+1} = \mathbf{z} - Q_{j+1}\mathbf{h} = \mathbf{z} - Q_{j+1}Q_{j+1}^*\mathbf{z},$$

we formally have $Q_{j+1}^*\mathbf{r}_{j+1} = \mathbf{0}$. However, classical Gram–Schmidt orthogonalization is faster but not so accurate as modified Gram–Schmidt orthogonalization [1]. So, often, $Q_{j+1}^*\mathbf{r}_{j+1}$ is quite large. Therefore, the orthogonalization is iterated to get sufficient orthogonality.

A possible modification of step 8 that incorporates a second iteration is

---

8: $\mathbf{h} := Q_{j+1}^*\mathbf{z}; \quad \mathbf{r}_{j+1} := \mathbf{z} - Q_{j+1}\mathbf{h};$
$\mathbf{c} := Q_{j+1}^*\mathbf{r}_{j+1}; \quad \mathbf{r}_{j+1} := \mathbf{r}_{j+1} - Q_{j+1}\mathbf{c}; \quad \mathbf{h} = \mathbf{h} + \mathbf{c};$

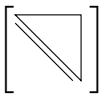
Now we have,

$$\begin{aligned}\tilde{\mathbf{r}}_{j+1} &= \text{corrected } \mathbf{r}_{j+1}\\ &= \mathbf{r}_{j+1} - Q_{j+1}\underbrace{Q_{j+1}^*\mathbf{r}_{j+1}}_{\mathbf{c}}\\ &= \mathbf{z} - Q_{j+1}\underbrace{Q_{j+1}^*\mathbf{z}}_{\mathbf{h}} - Q_{j+1}\underbrace{Q_{j+1}^*\mathbf{r}_{j+1}}_{\mathbf{c}} = \mathbf{z} - Q_{j+1}(\mathbf{h}+\mathbf{c})\end{aligned}$$

More iterations are possible but seldom necessary. □

After the execution of Algorithm 10.1 we have the Arnoldi / Lanczos relation

$$AQ_m = Q_m H_m + \mathbf{r}_m\mathbf{e}_m^*, \qquad H_m = \begin{bmatrix} \\ \\ \end{bmatrix} \tag{10.1}$$



available with

$$\mathbf{r}_m = \beta_m\mathbf{q}_{m+1}, \qquad \|\mathbf{q}_{m+1}\| = 1.$$

If $\beta_m = 0$ then $\mathcal{R}(Q_m)$ is invariant under A, i.e., $A\mathbf{x} \in \mathcal{R}(Q_m)$ for all $\mathbf{x} \in \mathcal{R}(Q_m)$. This lucky situation implies that $\sigma(H_m) \subset \sigma_m(A)$. So, the Ritz values and vectors are eigenvalues and eigenvectors of A.

What we can realistically hope for is β_m being small. Then,

$$AQ_m - \mathbf{r}_m\mathbf{e}_m^* = (A - \mathbf{r}_m\mathbf{q}_m^*)Q_m = Q_m H_m.$$

Then, $\mathcal{R}(Q_m)$ is invariant under a matrix $A + E$, that differs from A by a perturbation E with $\|E\| = \|\mathbf{r}_m\| = |\beta_m|$. From general eigenvalue theory we know that this in this situation well-conditioned eigenvalues of H_m are good approximations of eigenvalues of A.

In the sequel we investigate how we can find a q_1 such that β_m becomes small?

10.2 Implicit restart

Let us start from the Arnoldi relation

$$AQ_m = Q_m H_m + \mathbf{r}_m\mathbf{e}_m^*, \tag{10.2}$$

Algorithm 10.2 k implicit QR steps applied to H_m

1: $H_m^+ := H_m$.
2: **for** $i := 1, \ldots, k$ **do**
3: $\quad H_m^+ := V_i^* H_m^+ V_i, \qquad$ where $H_m^+ - \mu_i I = V_i R_i \quad$ (QR factorization)
4: **end for**

that is obtained after calling Algorithm 10.1.

We apply $k < m$ implicit QR steps to H_m with shifts $\mu_1, \ldots, \mu_k$, see Algorithm 10.2. Let $V^+ := V_1 V_2 \cdots V_k$. V^+ is the product of k (unitary) Hessenberg matrices whence it has k nonzero off-diagonals below its main diagonal.

$$V_m^+ = \begin{bmatrix} & & \\ & & \\ & & \\ & & \\ \underbrace{\qquad}_{k} & & \end{bmatrix}.$$

We define

$$Q_m^+ := Q_m V^+, \qquad H_m^+ := (V^+)^* H_m V^+.$$

Then, from (10.2) we obtain

$$AQ_m V^+ = Q_m V^+ (V^+)^* H_m V^+ + \mathbf{r}_m \mathbf{e}_m^* V^+,$$

or

$$AQ_m^+ = Q_m^+ H_m^+ + \mathbf{r}_m \mathbf{e}_m^* V^+. \tag{10.3}$$

As V^+ has k nonzero off-diagonals below the main diagonal, the last row of V^+ has the form

$$\mathbf{e}_m^* V^+ = (\underbrace{0, \ldots, 0}_{p-1}, \underbrace{*, \ldots, *}_{k+1}), \qquad k + p = m.$$

We now simply discard the last k columns of (10.3).

$$\begin{aligned} AQ_m^+(:, 1:p) &= Q_m^+ H_m^+(:, 1:p) + \mathbf{r}_m \mathbf{e}_m^* V^+(:, 1:p) \\ &= Q_m^+(:, 1:p) H_m^+(1:p, 1:p) + \underbrace{h_{p+1,p}^+}_{\beta_p^+} \mathbf{q}_{p+1}^+ \mathbf{e}_p^* + v_{m,p}^+ \mathbf{r}_m \mathbf{e}_p^* \\ &= Q_m^+(:, 1:p) H_m^+(1:p, 1:p) + \underbrace{(\mathbf{q}_{p+1}^+ h_{p+1,p}^+ + \mathbf{r}_m v_{m,p}^+)}_{\mathbf{r}_p^+} \mathbf{e}_p^*. \end{aligned}$$

In Algorithm 10.3 we have collected what we have derived so far. We have however left open in step 3 of the algorithm *how* the shifts $\mu_1, \ldots, \mu_k$ should be chosen. In ARPACK [3], all eigenvalues of H_m are computed. Those k eigenvalues that are furthest away from some target value are chosen as shifts. We have not specified how we determine convergence, too.

One can show that a QR step with shift μ_i transforms the vector $\mathbf{q}_1$ in a multiple of $(A - \mu_i I)\mathbf{q}_1$. In fact, a simple modification of the Arnoldi relation (10.2) gives

$$(A - \mu_i I) Q_m = Q_m \underbrace{(H_m - \mu_i I)}_{V_1 R_1} + \mathbf{r}_m \mathbf{e}_m^* = Q_m V_1 R_1 + \mathbf{r}_m \mathbf{e}_m^*.$$

Algorithm 10.3 Implicitely restarted Arnoldi (IRA)

Let the Arnoldi relation $AQ_m = Q_m H_m + \mathbf{r}_m \mathbf{e}_m^*$ be given.
repeat
  Determine k shifts $\mu_1, \ldots, \mu_k$;
  $\mathbf{v}^* := \mathbf{e}_m^*$;
  for $i = 1, \ldots, k$ **do**
    $H_m - \mu_i I = V_i R_i$; /* QR factorization */
    $H_m := V_i^* H_m V_i$; $Q_m := Q_m V_i$;
    $\mathbf{v}^* := \mathbf{v}^* V_i$;
  end for
  $\mathbf{r}_p := \mathbf{q}_{p+1}^+ \beta_p^+ + \mathbf{r}_m v_{m,p}^+$;
  $Q_p := Q_m(:, 1:p)$; $H_p := H_m(1:p, 1:p)$;
  Starting with

$$AQ_p = Q_p H_p + \mathbf{r}_p \mathbf{e}_p^*$$

  execute k additional steps of the Arnoldi algorithm until

$$AQ_m = Q_m H_m + \mathbf{r}_m \mathbf{e}_m^*.$$

until convergence

Comparing the first columns in this equation gives

$$(A - \mu_i I)\mathbf{q}_1 = \mathbf{q}_1^{(1)} V_1 \mathbf{e}_1 r_{11} + \mathbf{0} = \mathbf{q}_1^{(1)} r_{11}.$$

By consequence, all k steps combined give

$$\mathbf{q}_1 \longleftarrow \Psi(A)\mathbf{q}_1, \qquad \Psi(\lambda) = \prod_{i=1}^{k} (\lambda - \mu_i).$$

If μ_i were an eigenvalue of A then $(A - \mu_i I)\mathbf{q}_1$ removes components of $\mathbf{q}_1$ in the direction of the corresponding eigenvector. More general, if μ_i is close to an eigenvalue of A then $(A - \mu_i I)\mathbf{q}_1$ will have only small components in the direction of eigenvectors corresponding to nearby eigenvalues. Choosing the μ_i equal to Ritz values far away from the desired part of the spectrum thus enhances the desired component. Still there is the danger that in each sweep on Algorithm 10.3 the same undesired Ritz values are recovered. Therefore, other strategies for choosing the shifts have been proposed [2]. Experimental results indicate however, that the original strategy chosen in ARPACK mostly works best.

10.3 Convergence criterion

Let $H_m \mathbf{s} = \mathbf{s}\vartheta$ with $\|\mathbf{s}\| = 1$. Let $\hat{\mathbf{x}} = Q_m \mathbf{s}$. Then we have as earlier

$$\|A\hat{\mathbf{x}} - \vartheta\hat{\mathbf{x}}\| = \|AQ_m\mathbf{s} - Q_m H_m \mathbf{s}\| = \|\mathbf{r}_m\| |\mathbf{e}_m^* \mathbf{s}| = \beta_m |\mathbf{e}_m^* \mathbf{s}|. \tag{10.4}$$

In the Hermitian case, $A = A^*$, the Theorem 9.1 of Krylov–Bogoliubov provides an interval that contains an eigenvalue of A. In the general case, we have

$$(A + E)\hat{\mathbf{x}} = \vartheta\hat{\mathbf{x}}, \qquad E = -\mathbf{r}_m \mathbf{q}_m^*, \quad \|E\| = \|\mathbf{r}_m\| = \beta_m. \tag{10.5}$$

According to an earlier theorem we know that if $\lambda \in \sigma(A)$ is simple and ϑ is the eigenvalue of $A + E$ closest to λ, then

$$|\lambda - \vartheta| \leq \frac{\|E\|}{\mathbf{y}^*\mathbf{x}} + \mathcal{O}(\|E\|^2). \tag{10.6}$$

Here, $\mathbf{y}$ and $\mathbf{x}$ are left and right eigenvectors of E corresponding to the eigenvalue λ. A similar statement holds for the eigenvectors, but the distance (gap) to the next eigenvalue comes into play as well.

In ARPACK, a Ritz pair $(\vartheta, \hat{\mathbf{x}})$ is considered converged if

$$\beta_m |\mathbf{e}_m^* \mathbf{s}| \leq \max(\varepsilon_M \|H_m\|, \mathrm{tol} \cdot |\vartheta|). \tag{10.7}$$

As $|\vartheta| \leq \|H_m\| \leq \|A\|$, the inequality $\|E\| \leq \mathrm{tol} \cdot \|A\|$ holds at convergence. According to (10.6) well-conditioned eigenvalues are well approximated.

10.4 The generalized eigenvalue problem

Let us consider now the generalized eigenvalue problem

$$A\mathbf{x} = \lambda M\mathbf{x}. \tag{10.8}$$

Applying a shift-and-invert spectral transformation with shift σ transforms (10.8) into

$$S\mathbf{x} = (A - \sigma M)^{-1} M\mathbf{x} = \mu\mathbf{x}, \qquad \mu = \frac{1}{\lambda - \sigma}. \tag{10.9}$$

We now execute an Arnoldi/Lanczos iteration with S to obtain

$$SQ_m = Q_m H_m + \mathbf{r}_m \mathbf{e}_m^*, \qquad Q_m^* M Q_m = I_m, \quad Q_m^* M \mathbf{r}_m = \mathbf{0}. \tag{10.10}$$

Let $\mathbf{s}$ with $\|\mathbf{s}\| = 1$ be an eigenvector of H_m with Ritz value ϑ. Let $\mathbf{y} = Q_m\mathbf{s}$ be the associated Ritz vector. Then,

$$SQ_m\mathbf{s} = S\mathbf{y} = Q_m H_m \mathbf{s} + \mathbf{r}_m \mathbf{e}_m^* \mathbf{s} = \mathbf{y}\vartheta + \mathbf{r}_m \mathbf{e}_m^* \mathbf{s}. \tag{10.11}$$

So, $\mathbf{y}\vartheta + \mathbf{r}_m \mathbf{e}_m^* \mathbf{s}$ can be considered a vector that is obtained by one step of inverse iteration. This vector is an improved approximation to the desired eigenvector, obtained at negligible cost. This so-called **eigenvector purification** is particularly important if M is singular.

Let us bound the residual norm of the purified vector. With (10.11) we have

$$M\mathbf{y} = (A - \sigma M)(\underbrace{\mathbf{y}\vartheta + \mathbf{r}_m \mathbf{e}_m^* \mathbf{s}}_{\tilde{\mathbf{y}}}) \tag{10.12}$$

with

$$\|\tilde{\mathbf{y}}\|_M = \sqrt{\vartheta^2 + \beta_k^2 |\mathbf{e}_m^* \mathbf{s}|^2}.$$

This equality holds as $\mathbf{y} \perp_M \mathbf{r}$. By consequence,

$$\begin{aligned} \|A\tilde{\mathbf{y}} - \lambda M\tilde{\mathbf{y}}\| &= \|(A - \sigma M)\tilde{\mathbf{y}} + M\tilde{\mathbf{y}}(\underbrace{\sigma - \lambda}_{-\frac{1}{\vartheta}})\| \\ &= \|M\mathbf{y} - M(\mathbf{y}\vartheta + \mathbf{r}_m \mathbf{e}_m^* \mathbf{s})/\vartheta\| = \|M\mathbf{r}\| \, |\mathbf{e}_m^* \mathbf{s}|/|\vartheta|. \end{aligned} \tag{10.13}$$

Since $|\vartheta|$ is large in general, we obtain good bounds for the residual of the purified eigenvectors.

```
EIGS  Find a few eigenvalues and eigenvectors of a matrix using ARPACK
   D = EIGS(A) returns a vector of A's 6 largest magnitude eigenvalues.
   A must be square and should be large and sparse.

   [V,D] = EIGS(A) returns a diagonal matrix D of A's 6 largest magnitude
   eigenvalues and a matrix V whose columns are the corresponding
   eigenvectors.

   [V,D,FLAG] = EIGS(A) also returns a convergence flag. If FLAG is 0 then
   all the eigenvalues converged; otherwise not all converged.

   EIGS(A,B) solves the generalized eigenvalue problem A*V == B*V*D. B
   must be symmetric (or Hermitian) positive definite and the same size as
   A. EIGS(A,[],...) indicates the standard eigenvalue problem A*V == V*D.

   EIGS(A,K) and EIGS(A,B,K) return the K largest magnitude eigenvalues.

   EIGS(A,K,SIGMA) and EIGS(A,B,K,SIGMA) return K eigenvalues. If SIGMA is:
      'LM' or 'SM' - Largest or Smallest Magnitude
   For real symmetric problems, SIGMA may also be:
      'LA' or 'SA' - Largest or Smallest Algebraic
      'BE' - Both Ends, one more from high end if K is odd
   For nonsymmetric and complex problems, SIGMA may also be:
      'LR' or 'SR' - Largest or Smallest Real part
      'LI' or 'SI' - Largest or Smallest Imaginary part
   If SIGMA is a real or complex scalar including 0, EIGS finds the
   eigenvalues closest to SIGMA. For scalar SIGMA, and when SIGMA = 'SM',
   B need only be symmetric (or Hermitian) positive semi-definite since it
   is not Cholesky factored as in the other cases.

   EIGS(A,K,SIGMA,OPTS) and EIGS(A,B,K,SIGMA,OPTS) specify options:
   OPTS.issym: symmetry of A or A-SIGMA*B represented by AFUN [{false} | true]
   OPTS.isreal: complexity of A or A-SIGMA*B represented by AFUN [false | {true}]
   OPTS.tol: convergence: Ritz estimate residual <= tol*NORM(A) [scalar | {eps}]
   OPTS.maxit: maximum number of iterations [integer | {300}]
   OPTS.p: number of Lanczos vectors: K+1<p<=N [integer | {2K}]
   OPTS.v0: starting vector [N-by-1 vector | {randomly generated}]
   OPTS.disp: diagnostic information display level [0 | {1} | 2]
   OPTS.cholB: B is actually its Cholesky factor CHOL(B) [{false} | true]
   OPTS.permB: sparse B is actually CHOL(B(permB,permB)) [permB | {1:N}]
   Use CHOL(B) instead of B when SIGMA is a string other than 'SM'.

   EIGS(AFUN,N) accepts the function AFUN instead of the matrix A. AFUN is
   a function handle and Y = AFUN(X) should return
      A*X            if SIGMA is unspecified, or a string other than 'SM'
      A\X            if SIGMA is 0 or 'SM'
      (A-SIGMA*I)\X  if SIGMA is a nonzero scalar (standard problem)
      (A-SIGMA*B)\X  if SIGMA is a nonzero scalar (generalized problem)
   N is the size of A. The matrix A, A-SIGMA*I or A-SIGMA*B represented by
   AFUN is assumed to be real and nonsymmetric unless specified otherwise
   by OPTS.isreal and OPTS.issym. In all these EIGS syntaxes, EIGS(A,...)
   may be replaced by EIGS(AFUN,N,...).

   Example:
      A = delsq(numgrid('C',15));  d1 = eigs(A,5,'SM');

   Equivalently, if dnRk is the following one-line function:
      %-----------------------------%
      function y = dnRk(x,R,k)
      y = (delsq(numgrid(R,k))) \ x;
      %-----------------------------%

      n = size(A,1);  opts.issym = 1;
      d2 = eigs(@(x)dnRk(x,'C',15),n,5,'SM',opts);

   See also eig, svds, ARPACKC, function_handle.
```

10.5 A numerical example

This example is taken from the MATLAB document pages regarding `eigs`. `eigs` is the MATLAB interface to the ARPACK code, see page 172. The matrix called `west0479` is a 479×479 matrix originating in a chemical engineering plant model. The matrix is available from the Matrix Market [4], a web site that provides numerous test matrices. Its nonzero structure is given in Fig. 10.1

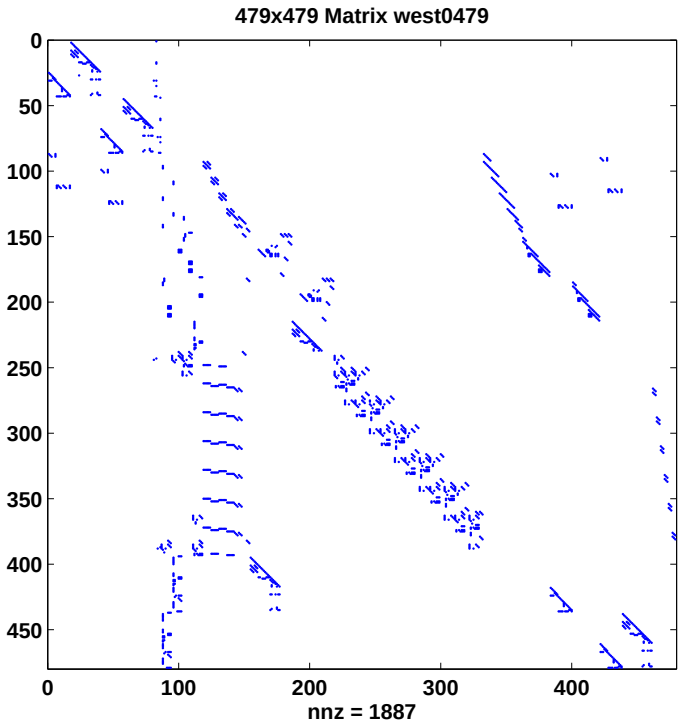


Figure 10.1: Nonzero structure of the 479×479 matrix west0479

To compute the eight largest eigenvalues of this matrix we issue the following MATLAB commands.

```
>> load west0479
>> d = eig(full(west0479));
>> dlm=eigs(west0479,8);
Iteration 1: a few Ritz values of the 20-by-20 matrix:
     0
     0
     0
     0
     0
     0
     0
     0
     0

Iteration 2: a few Ritz values of the 20-by-20 matrix:
   1.0e+03 *

  -0.0561 - 0.0536i
```

```
    0.1081 + 0.0541i
    0.1081 - 0.0541i
   -0.1009 - 0.0666i
   -0.1009 + 0.0666i
   -0.0072 + 0.1207i
   -0.0072 - 0.1207i
    0.0000 - 1.7007i
    0.0000 + 1.7007i

Iteration 3: a few Ritz values of the 20-by-20 matrix:
   1.0e+03 *

   -0.0866
   -0.1009 - 0.0666i
   -0.1009 + 0.0666i
   -0.0072 + 0.1207i
   -0.0072 - 0.1207i
    0.1081 - 0.0541i
    0.1081 + 0.0541i
    0.0000 - 1.7007i
    0.0000 + 1.7007i

Iteration 4: a few Ritz values of the 20-by-20 matrix:
   1.0e+03 *

    0.0614 - 0.0465i
   -0.0072 - 0.1207i
   -0.0072 + 0.1207i
    0.1081 + 0.0541i
    0.1081 - 0.0541i
   -0.1009 + 0.0666i
   -0.1009 - 0.0666i
    0.0000 - 1.7007i
    0.0000 + 1.7007i

Iteration 5: a few Ritz values of the 20-by-20 matrix:
   1.0e+03 *

   -0.0808
   -0.0072 + 0.1207i
   -0.0072 - 0.1207i
   -0.1009 - 0.0666i
   -0.1009 + 0.0666i
    0.1081 + 0.0541i
    0.1081 - 0.0541i
    0.0000 + 1.7007i
    0.0000 - 1.7007i

Iteration 6: a few Ritz values of the 20-by-20 matrix:
   1.0e+03 *

    0.0734 - 0.0095i
   -0.0072 + 0.1207i
   -0.0072 - 0.1207i
    0.1081 - 0.0541i
```

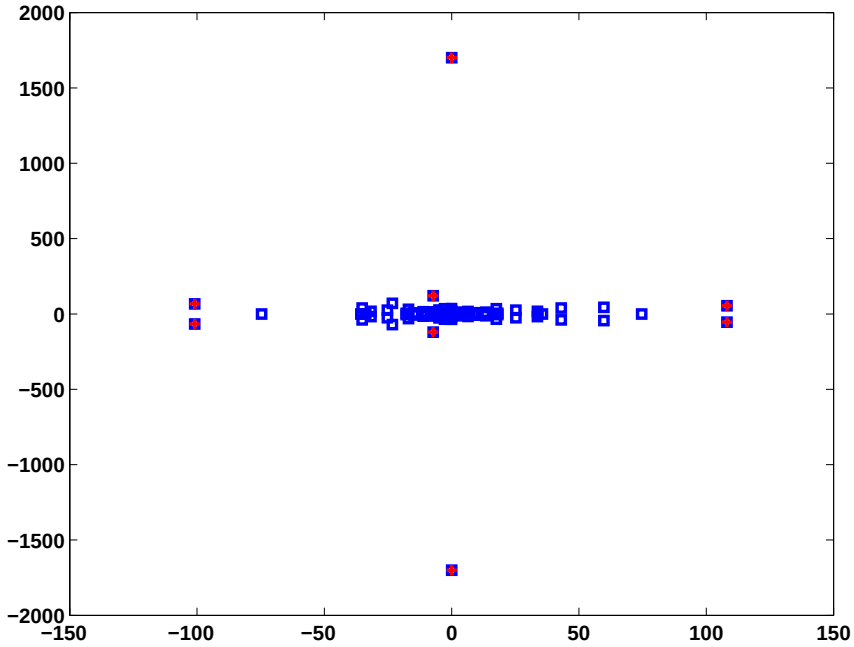


Figure 10.2: Spectrum of the matrix west0479

```
   0.1081 + 0.0541i
  -0.1009 - 0.0666i
  -0.1009 + 0.0666i
   0.0000 - 1.7007i
   0.0000 + 1.7007i

Iteration 7: a few Ritz values of the 20-by-20 matrix:
   1.0e+03 *

  -0.0747
  -0.0072 - 0.1207i
  -0.0072 + 0.1207i
   0.1081 + 0.0541i
   0.1081 - 0.0541i
  -0.1009 + 0.0666i
  -0.1009 - 0.0666i
   0.0000 + 1.7007i
   0.0000 - 1.7007i
```

The output indicates that `eigs` needs seven sweeps to compute the eigenvalues to the default accuracy of macheps$\|A\|$. The Ritz values given are the approximations of the eigenvalues we want to compute. The complete spectrum of `west0479` is given in Fig. 10.2. Notice the different scales of the axes! Fig. 10.3 is a zoom that shows all eigenvalues except the two very large ones. Here the axes are equally scaled. From the two figures it becomes clear that `eigs` has computed the eight eigenvalues (and corresponding eigenvectors) of *largest modulus*.

To compute the eigenvalues smallest in modulus we issue the following command.

```
dsm=eigs(west0479,8,'sm');
Iteration 1: a few Ritz values of the 20-by-20 matrix:
     0
```

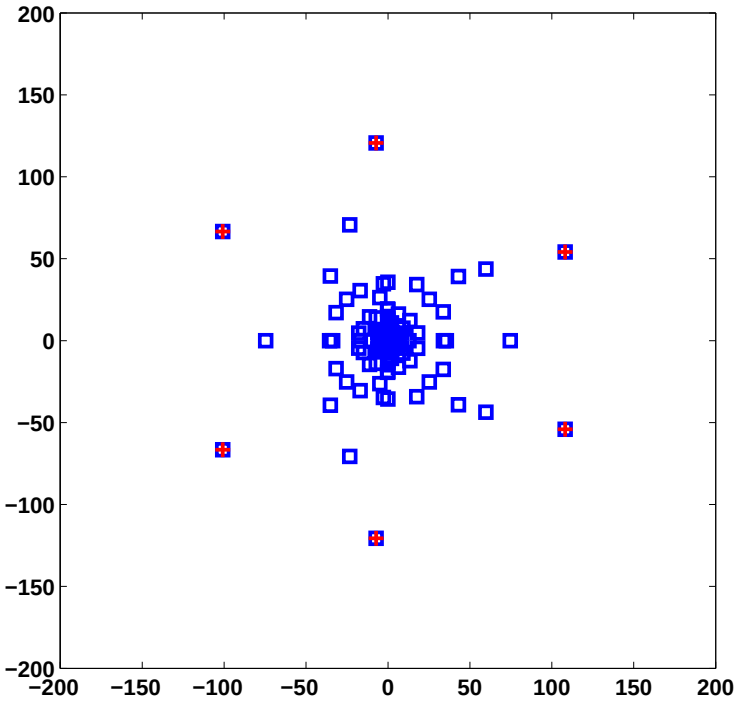


Figure 10.3: A zoom to the center of the spectrum of matrix west0479 that excludes the largest two eigenvalues on the imaginary axis

```
     0
     0
     0
     0
     0
     0
     0
     0

Iteration 2: a few Ritz values of the 20-by-20 matrix:
   1.0e+03 *

  -0.0228 - 0.0334i
   0.0444
  -0.0473
   0.0116 + 0.0573i
   0.0116 - 0.0573i
  -0.0136 - 0.1752i
  -0.0136 + 0.1752i
  -3.4455
   5.8308

Iteration 3: a few Ritz values of the 20-by-20 matrix:
   1.0e+03 *
```

```
  -0.0228 - 0.0334i
   0.0444
  -0.0473
   0.0116 + 0.0573i
   0.0116 - 0.0573i
  -0.0136 + 0.1752i
  -0.0136 - 0.1752i
  -3.4455
   5.8308

Iteration 4: a few Ritz values of the 20-by-20 matrix:
   1.0e+03 *

  -0.0228 + 0.0334i
   0.0444
  -0.0473
   0.0116 - 0.0573i
   0.0116 + 0.0573i
  -0.0136 + 0.1752i
  -0.0136 - 0.1752i
  -3.4455
   5.8308

Iteration 5: a few Ritz values of the 20-by-20 matrix:
   1.0e+03 *

  -0.0228 + 0.0334i
   0.0444
  -0.0473
   0.0116 - 0.0573i
   0.0116 + 0.0573i
  -0.0136 + 0.1752i
  -0.0136 - 0.1752i
  -3.4455
   5.8308

>> dsm

dsm =

   0.0002
  -0.0003
  -0.0004 - 0.0057i
  -0.0004 + 0.0057i
   0.0034 - 0.0168i
   0.0034 + 0.0168i
  -0.0211
   0.0225
```

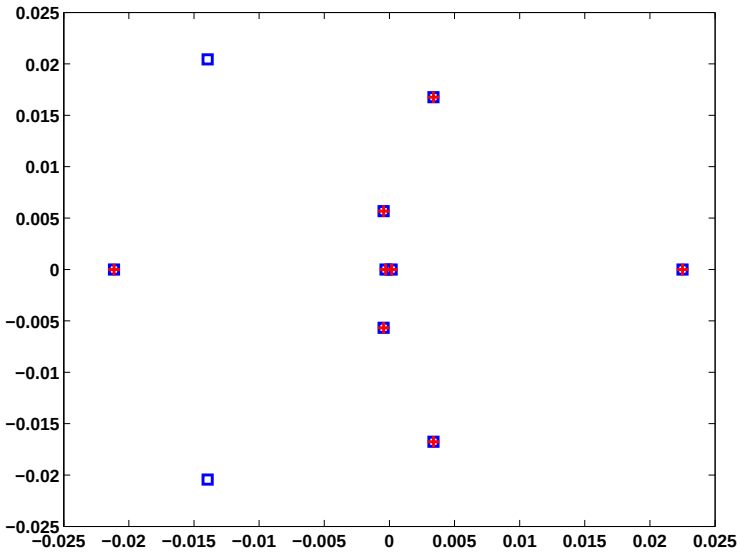


Figure 10.4: Smallest eigenvalues of the matrix west0479

```
>> 1./dsm

ans =

   1.0e+03 *

   5.8308
  -3.4455
  -0.0136 + 0.1752i
  -0.0136 - 0.1752i
   0.0116 + 0.0573i
   0.0116 - 0.0573i
  -0.0473
   0.0444
```

The computed eigenvalues are depicted in Fig. 10.4

10.6 Another numerical example

We revisit the determination the acoustic eigenfrequencies and modes in the interior of a car, see section 1.6.3. The computations are done with the finest grid depicted in Fig. 1.9. We first compute the lowest ten eigenpairs with simultaneous inverse vector iteration (`sivit`). The dimension of the search space is 15.

```
>> [p,e,t]=initmesh('auto');
>> [p,e,t]=refinemesh('auto',p,e,t);
>> [p,e,t]=refinemesh('auto',p,e,t);
>> p=jigglemesh(p,e,t);
>> [A,M]=assema(p,t,1,1,0);
>> whos
  Name      Size                    Bytes  Class
```

```
  A      1095x1095                91540  double array (sparse)
  M      1095x1095                91780  double array (sparse)
  e         7x188                 10528  double array
  p         2x1095                17520  double array
  t         4x2000                64000  double array

Grand total is 26052 elements using 275368 bytes

>> sigma=-.01;
>> p=10; tol=1e-6; X0=rand(size(A,1),15);
>> [V,L] = sivit(A,M,p,X0,sigma,tol);

 ||Res(0)|| = 0.998973
 ||Res(5)|| = 0.603809
 ||Res(10)|| = 0.0171238
 ||Res(15)|| = 0.00156298
 ||Res(20)|| = 3.69725e-05
 ||Res(25)|| = 7.11911e-07
>> %  25 x 15 =  375 matrix - vektor - multiplications until convergence
>>
>> format long, L

L =

   0.00000000000000
   0.01269007628847
   0.04438457596824
   0.05663501055565
   0.11663116522140
   0.13759210393200
   0.14273438015546
   0.20097619880776
   0.27263682280769
   0.29266080747831

>> format short
>> norm(V'*M*V - eye(10))

ans =

     1.8382e-15
```

Then we use MATLAB's solver `eigs`. We set the tolerance and the shift to be the same as with `sivit`. Notice that ARPACK applies a shift-and-invert spectral transformation if a shift is given.

```
>> options.tol=tol; options.issym=1;
>> [v,l,flag]=eigs(A,M,p,sigma,options);
Iteration 1: a few Ritz values of the 20-by-20 matrix:
     0
     0
     0
     0
     0
     0
     0
```

```
     0
     0
     0

Iteration 2: a few Ritz values of the 20-by-20 matrix:
    3.3039
    3.5381
    4.7399
    6.5473
    6.7754
    7.8970
   15.0071
   18.3876
   44.0721
  100.0000

Iteration 3: a few Ritz values of the 20-by-20 matrix:
    3.3040
    3.5381
    4.7399
    6.5473
    6.7754
    7.8970
   15.0071
   18.3876
   44.0721
  100.0000

>> flag

flag =

     0

>> l=diag(l);  l=l(end:-1:1); norm(l-L)

ans =

   3.7671e-14

>> norm(v'*M*v - eye(10))

ans = 8.0575e-15
```

Clearly the eigenvectors are mutually m-orthogonal. Notice that `eigs` returns the eigenvalues sorted from large to small such that they have to be reordered before comparing with those `sivit` computed.

In the next step we compute the largest eigenvalues of the matrix

$$S = R(A - \sigma M)^{-1} R^T, \tag{10.14}$$

where $R^T R = M$ is the Cholesky factorization of M. The matrix in (10.14) is transferred to `eigs` as a function.

```
>> type afun
```

```
function x = afun(x)
global RA RB

x = RB*(RA\(RA'\(RB'*x)));

>> global RA RB
>> RA = chol(A-sigma*M);
>> RB = chol(M);
>> [v,l1,flag]=eigs('afun',n,10,'lm',options);
Iteration 1: a few Ritz values of the 20-by-20 matrix:
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0

Iteration 2: a few Ritz values of the 20-by-20 matrix:
    3.3030
    3.5380
    4.7399
    6.5473
    6.7754
    7.8970
   15.0071
   18.3876
   44.0721
  100.0000

Iteration 3: a few Ritz values of the 20-by-20 matrix:
    3.3040
    3.5381
    4.7399
    6.5473
    6.7754
    7.8970
   15.0071
   18.3876
   44.0721
  100.0000

>> flag

flag =

     0

>> l1 = diag(l1)

l1 =
```

```
  100.0000
   44.0721
   18.3876
   15.0071
    7.8970
    6.7754
    6.5473
    4.7399
    3.5381
    3.3040

>> sigma + 1./l1

ans =

    0.0000
    0.0127
    0.0444
    0.0566
    0.1166
    0.1376
    0.1427
    0.2010
    0.2726
    0.2927

>> norm(sigma + 1./l1 - l)

ans =

   4.4047e-14
```

10.7 The Lanczos algorithm with thick restarts

The implicit restarting procedures discussed so far are very clever ways to get rid of unwanted directions in the search space and still keeping a Lanczos or Arnoldi basis. The latter admits to continue the iteration in a known framework. The Lanczos or Arnoldi relations hold that admit very efficient checks for convergence. The restart has the effect of altering the starting vector.

In this and the next section we discuss algorithms that work with Krylov spaces but are not restricted to Krylov or Arnoldi bases. Before continuing we make a step back and consider how we can determine if a given subspace of $\mathbb{F}^{n\times n}$ is a Krylov space at all.

Let A be an n-by-n matrix and let $\mathbf{v}_1, \ldots, \mathbf{v}_k$ be linearly independent n-vectors. Is the subspace $\mathcal{V} := \text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ a Krylov space, i.e. is there a vector $\mathbf{q} \in \mathcal{V}$ such that $\mathcal{V} = \mathcal{K}_k(A, \mathbf{q})$? The following theorem gives the answer.

Theorem 10.1 *$\mathcal{V} = span\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$ is a Krylov space if and only if there is a k-by-k matrix M such that*

$$R := AV - VM, \qquad V = [\mathbf{v}_1, \ldots, \mathbf{v}_k], \tag{10.15}$$

has rank one.

Proof. Let us first assume that $\mathcal{V} = \mathcal{K}_k(A, \mathbf{q})$ for some $\mathbf{q} \in \mathcal{V}$. Let $Q = [\mathbf{q}_1, \ldots, \mathbf{q}_k]$ be the Arnoldi basis of $\mathcal{K}_k(A, \mathbf{q})$. Then $Q = VS$ with S a nonsingular k-by-k matrix. We now multiply the Arnoldi relation

$$AQ = QH + \tilde{\mathbf{q}}_{k+1}\mathbf{e}_k^T, \quad Q^*\tilde{\mathbf{q}}_{k+1} = \mathbf{0}, \qquad H \text{ Hessenberg.}$$

by S^{-1} from the right to get

$$AV = VSHS^{-1} + \tilde{\mathbf{q}}_{k+1}\mathbf{e}_k^* S^{-1}.$$

which is (10.15) with $M = SHS^{-1}$.

Let us now assume that R in (10.15) has rank 1 so that we can write

$$AV = VM + R = VM + \mathbf{v}\mathbf{w}^*, \qquad M \in \mathbb{F}^{k\times k}. \tag{10.16}$$

with some $\mathbf{v} \in \mathbb{F}^n$ and $\mathbf{w} \in \mathbb{F}^k$. Let S_1, $S_1^{-1} = S_1^*$, be the Householder reflector that maps $\mathbf{w}$ onto a multiple of $\mathbf{e}_k$, $S_1^*\mathbf{w} = \gamma\mathbf{e}_k$. Then, (10.16) becomes

$$AVS_1 = VS_1S_1^*MS_1 + \gamma\mathbf{v}\mathbf{e}_k^T.$$

There is another unitary matrix S_2 with $S_2^*\mathbf{e}_k = \mathbf{e}_k$ that transforms $S_1^*MS_1$ similarly to Hessenberg form,

$$S^*MS = H, \qquad H \text{ Hessenberg,}$$

where $S = S_1S_2$. S_2 can be formed as the product of Householder reflectors. In contrast to the well-known transformation of full matrices to Hessenberg form, here the zeros are generated row-wise starting with the last. Thus,

$$AVS = VSH + \gamma\mathbf{v}\mathbf{e}_k^T.$$

So, $\mathcal{V} = \mathcal{K}_k(A, \mathbf{q})$ with $\mathbf{q} = VS\mathbf{e}_1$. ∎

We apply this theorem to the case where a subspace is spanned by some Ritz vectors. Let $A = A^*$ and let

$$AQ_k - Q_kT_k = \beta_{k+1}\mathbf{q}_{k+1}\mathbf{e}_k^T \tag{10.17}$$

be a Lanczos relation. Let

$$T_kS_k = S_k\Theta_k, \qquad S_k = [\mathbf{s}_1^{(k)}, \ldots, \mathbf{s}_k^{(k)}], \quad \Theta_k = \operatorname{diag}(\vartheta_1, \ldots, \vartheta_k).$$

be the spectral decomposition of the tridiagonal matrix T_k. Then, for all i, the Ritz vector

$$\mathbf{y}_i = Q_k\mathbf{s}_i^{(k)} \in \mathcal{K}_k(A, \mathbf{q})$$

gives rise to the residual

$$\mathbf{r}_i = A\mathbf{y}_i - \mathbf{y}_i\vartheta_i = \beta_{k+1}\mathbf{q}_{k+1}\mathbf{e}_k^*\mathbf{s}_i^{(k)} \in \mathcal{K}_{k+1}(A, \mathbf{q}) \ominus \mathcal{K}_k(A, \mathbf{q}).$$

Therefore, for any set of indices $1 \le i_1 < \cdots < i_j \le k$ we have

$$A[\mathbf{y}_{i_1}, \mathbf{y}_{i_2}, \ldots, \mathbf{y}_{i_j}] - [\mathbf{y}_{i_1}, \mathbf{y}_{i_2}, \ldots, \mathbf{y}_{i_j}]\operatorname{diag}(\vartheta_{i_1}, \ldots, \vartheta_{i_j}) = \beta_{k+1}\mathbf{q}_{k+1}[s_{i_1}^{(k)}, s_{i_2}^{(k)}, \ldots, s_{i_j}^{(k)}].$$

By Theorem 10.1 we see that any set $[\mathbf{y}_{i_1}, \mathbf{y}_{i_2}, \ldots, \mathbf{y}_{i_j}]$ of Ritz vectors forms a Krylov space. Note that the generating vector differs for each set.

Algorithm 10.4 Thick restart Lanczos

Let us be given k Ritz vectors $\mathbf{y}_i$ and a residual vector $\mathbf{r}_k$ sucht that $A\mathbf{y}_i = \vartheta_i \mathbf{y}_i + \sigma_i \mathbf{r}_k$, $i = 1, \ldots, k$. The value k may be zero in which case $\mathbf{r}_0$ is the initial guess.
This algorithm computes an orthonormal basis $\mathbf{y}_1, \ldots, \mathbf{y}_j, \mathbf{q}_{j+1}, \ldots, \mathbf{q}_m$ that spans a m-dimensional Krylov space whose generating vector is not known unless $k = 0$.
$\mathbf{q}_{k+1} := \mathbf{r}_k / \|\mathbf{r}_k\|$.
$\mathbf{z} := A\mathbf{q}_{k+1}$;
$\alpha_{k+1} := \mathbf{q}_{k+1}^* \mathbf{z}$;
$\mathbf{r}_{k+1} = \mathbf{z} - \beta_k \mathbf{q}_{k+1} - \sum_{i=1}^{k} \sigma_i \mathbf{y}_i$
$\beta_{k+1} := \|\mathbf{r}_{k+1}\|$
for $i = k+2, \ldots, m$ **do**
  $\mathbf{q}_i := \mathbf{r}_{i-1} / \beta_{i-1}$.
  $\mathbf{z} := A\mathbf{q}_i$;
  $\alpha_i := \mathbf{q}_i^* \mathbf{z}$;
  $\mathbf{r}_i = \mathbf{z} - \alpha_i \mathbf{q}_i - \beta_{i-1} \mathbf{q}_{i-1}$
  $\beta_i = \|\mathbf{r}_i\|$
end for

We now split the indices $1, \ldots, k$ in two sets. The first set contains the 'good' Ritz vectors that we want to keep and that we collect in Y_1, the second set contains the 'bad' Ritz vectors ones that we want to remove. Those we put in Y_2. In this way we get

$$A[Y_1, Y_2] - [Y_1, Y_2] \begin{bmatrix} \Theta_1 & \\ & \Theta_2 \end{bmatrix} = \beta_{k+1} \mathbf{q}_{k+1} [\mathbf{s}_1^*, \mathbf{s}_2^*]. \tag{10.18}$$

Keeping the first set of Ritz vectors and **purging** (deflating) the rest yields

$$AY_1 - Y_1 \Theta_1 = \beta_{k+1} \mathbf{q}_{k+1} \mathbf{s}_1^*.$$

We now can restart a Lanczos procedure by orthogonalizing $A\mathbf{q}_{k+1}$ against $Y_1 =: [\mathbf{y}_1^*, \ldots, \mathbf{y}_j^*]$ and $\mathbf{q}_{k+1}$. From the equation

$$A\mathbf{y}_i - \mathbf{y}_i \vartheta_i = \mathbf{q}_{k+1} \sigma_i, \qquad \sigma_i = \beta_{k+1} \mathbf{e}_k^* \mathbf{s}_i^{(k)}$$

we get

$$\mathbf{q}_{k+1}^* A \mathbf{y}_\ell = \sigma_\ell,$$

whence

$$\mathbf{r}_{k+1} = A\mathbf{q}_{k+1} - \beta_k \mathbf{q}_{k+1} - \sum_{i=1}^{j} \sigma_i \mathbf{y}_i \perp \mathcal{K}_{k+1}(A, \mathbf{q}_\cdot) \tag{10.19}$$

From this point on the Lanczos algorithm proceeds with the ordinary three-term recurrence. We finally arrive at a relation similar to (10.17), however, with

$$Q_m = [\mathbf{y}_1, \ldots, \mathbf{y}_j, \mathbf{q}_{k+1}, \ldots, \mathbf{q}_{m+k-j}]$$

and

$$T_m = \begin{pmatrix} \vartheta_1 & & & \sigma_1 & & & \\ & \ddots & & \vdots & & & \\ & & \vartheta_j & \sigma_j & & & \\ \sigma_1 & \cdots & \sigma_j & \alpha_{k+1} & \ddots & & \\ & & & \ddots & \ddots & \beta_{m+k-j-1} & \\ & & & & \beta_{m+k-j-1} & \alpha_{m+k-j} & \end{pmatrix}$$

This procedure called **thick restart** has been suggested by Wu & Simon [9], see Algorithm 10.4. It allows to restart with any number of Ritz vectors. In contrast to the implicitly restarted Lanczos procedure, here we need the spectral decomposition of T_m. Its computation is not an essential overhead in general. The spectral decomposition admits a simple sorting of Ritz values. We could further split the first set of Ritz pairs into converged and unconveregd ones, depending on the value $\beta_{m+1}|s_{k,i}|$. If this quantity is below a given threshold we set the value to zero and **lock** (deflate) the corresponding Ritz vector, i.e., accept it as an eigenvector.

The procedure is mathematically equivalent with the implicitely restarted Lanczos algorithm. In fact, the generating vector of the Krylov space span$\{\mathbf{y}_1, \ldots, \mathbf{y}_j, \mathbf{q}_{j+1}, \ldots, \mathbf{q}_m\}$ that we do not compute is $\mathbf{q}_1' = (A - \vartheta_{j+1}I)\cdots(A - \vartheta_m I)\mathbf{q}_1$. This restarting procedure is probably simpler than with IRL.

The problem of losing orthogonality is similar to plain Lanczos. Wu & Simon [9] investigate the various reorthogonalizing strategies known from plain Lanczos (full, selective, partial). In their numerical experiments the simplest procedure, full reorthogonalization, performs similarly or even faster than the more sophisticated reorthogonalization procedures.

Remark 10.2. The thick restart Lanczos procedure does not need a Krylov basis of span$\{\mathbf{y}_1, \ldots, \mathbf{y}_j\}$ or, equivalently, the tridiagonalization of

$$\begin{pmatrix} \vartheta_1 & & & \sigma_1 \\ & \ddots & & \vdots \\ & & \vartheta_j & \sigma_j \\ \sigma_1 & \cdots & \sigma_j & \alpha_{k+1} \end{pmatrix}.$$

However, at the next restart, the computation of the spectral decomposition will most probably require it.

Question: How can the arrow matrix above be tridiagonalized economically? □

10.8 Krylov-Schur algorithm

The Krylov-Schur algorithm introduced by Stewart [8] is a generalization of the thick-restart procedure for non-Hermitian problems. The Arnoldi algorithm constructs the Arnoldi relation

$$AQ_m = Q_m H_m + \mathbf{r}_m \mathbf{e}_m^*, \tag{10.1}$$

where H_m is Hessenberg. Let $H_m = S_m T_m S_m^*$ be a Schur decomposition of H_m with unitary S_m and triangular T_m. Then, similarly as in the previous section we have

$$AY_m = Y_m T_m + \mathbf{r}_m \mathbf{s}^*, \qquad Y_m = Q_m S_m, \quad \mathbf{s} = S^* \mathbf{e}_m. \tag{10.20}$$

The upper trangular form of T_m eases the analysis of the individual Ritz pairs. In particular, it admits moving unwanted Ritz values to the lower-right corner of T_m. (See the subroutine `_trexc` in LAPACK for details.) Similarly as in (10.18) we collect the 'good' and 'bad' Ritz vectors in matrices Y_1 and Y_2, respectively. In this way we get

$$A[Y_1, Y_2] - [Y_1, Y_2] \begin{bmatrix} T_{11} & T_{12} \\ & T_{22} \end{bmatrix} = \beta_{k+1} \mathbf{q}_{k+1} [\mathbf{s}_1^*, \mathbf{s}_2^*]. \tag{10.21}$$

Keeping the first set of Ritz vectors and purging the rest yields

$$AY_1 - Y_1 T_{11} = \beta_{k+1}\mathbf{q}_{k+1}\mathbf{s}_1^*.$$

The determination of a converged subspace is not so easy as with the thick-restart Lanczos procedure. However, if we manage to bring $\mathbf{s}_1$ into the form

$$\mathbf{s}_1 = \begin{bmatrix} \mathbf{s}_1' \\ \mathbf{s}_1'' \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{s}_1'' \end{bmatrix}$$

then we found an invariant subspace.

$$A[Y_1', Y_1''] - [Y_1', Y_1'''] \begin{bmatrix} T_{11}' & T_{12}' \\ & T_{22}' \end{bmatrix} = \beta_{k+1}\mathbf{q}_{k+1}[\mathbf{0}^T, \mathbf{s}_1''^*]$$

i.e.,

$$AY_1' = Y_1' T_{11}'$$

In most cases $\mathbf{s}_1'$ consists of a single element [8]. The columns in Y_1' are **locked**, i.e., they are not altered anymore. Orthogonality against them has to be enforced in the continuation of the eigenvalue computation.

10.9 The rational Krylov space method

After having computed a number of eigenvalue–eigen/Schurvector pairs in the neighborhood of some shift σ_1 with the shift-invert Lanczos, Arnoldi, or Krylov-Schur algorithm it may be advisable to restart with a changed shift σ_2. This is in fact possible without discarding the available Krylov space [5]. In this section we consider the generalized eigenvalue problem $A\mathbf{x} = \lambda B\mathbf{x}$.

The rational Krylov space method starts out as a shift-invert Arnoldi iteration with shift σ_1 and starting vector v_1. It computes an orthonormal basis V_j using the basic recurrence,

$$(A - \sigma_1 B)^{-1} B Q_j = Q_j H_j + \mathbf{r}_j \mathbf{e}^T = Q_{j+1}\bar{H}_j. \tag{10.22}$$

or, using the Schur decomposition of H_j, cf. (10.20),

$$(A - \sigma_1 B)^{-1} B Y_j = Y_j T_j + \mathbf{r}_j \mathbf{s}^* = Y_{j+1} \begin{bmatrix} T_j \\ \mathbf{s}^* \end{bmatrix}, \qquad Y_{j+1} = [Y_j, \mathbf{r}_j] \tag{10.23}$$

We want to derive a Krylov-Schur relation for a new shift $\sigma_2 \neq \sigma_1$ from (10.23) for the same space $\mathcal{R}(Y_{j+1})$ without accessing the matrices A or B. The tricky thing is to avoid discard all the information gathered in the basis Y_{j+1} that was computed with the old shift σ_1. This is indeed possible if we replace the basis Y_{j+1} with a new basis W_{j+1}, which spans the same subspace as Y_{j+1} but can be interpreted as the orthonormal basis of a Krylov-Schur relation with the new shift σ_2.

We rewrite the relation (10.23) as

$$BY_j = BY_{j+1} \begin{bmatrix} I_j \\ \mathbf{0}^* \end{bmatrix} = (A - \sigma_1 B) Y_{j+1} \begin{bmatrix} T_j \\ \mathbf{s}^* \end{bmatrix}.$$

Introducing the shift σ_2 this becomes

$$BY_{j+1} \left\{ \begin{bmatrix} I_j \\ \mathbf{0}^* \end{bmatrix} + (\sigma_1 - \sigma_2) \begin{bmatrix} T_j \\ \mathbf{s}^* \end{bmatrix} \right\} = (A - \sigma_2 B) Y_{j+1} \begin{bmatrix} T_j \\ \mathbf{s}^* \end{bmatrix}. \tag{10.24}$$

To construct a Krylov-Schur relation we must get rid of the last non-zero row of the matrix in braces in (10.24). To that end we use the QR factorization

$$\begin{bmatrix} I_j \\ \mathbf{0}^T \end{bmatrix} + (\sigma_1 - \sigma_2) \begin{bmatrix} T_j \\ \mathbf{s}^* \end{bmatrix} = Q_{j+1} \begin{bmatrix} R_j \\ \mathbf{0}^T \end{bmatrix}.$$

Using it we obtain

$$BY_{j+1}Q_{j+1} \begin{bmatrix} R_j \\ \mathbf{0}^T \end{bmatrix} \equiv BW_{j+1} \begin{bmatrix} R_j \\ \mathbf{0}^T \end{bmatrix} = BW_jR_j = (A - \sigma_2 B)W_{j+1}Q_{j+1}^* \begin{bmatrix} T_j \\ \mathbf{s}^* \end{bmatrix}$$

Multiplying with $(A - \sigma_2 B)^{-1}$ from the left we obtain

$$(A - \sigma_2 B)^{-1}BW_j = W_{j+1}Q_{j+1}^* \begin{bmatrix} T_jR_j^{-1} \\ \mathbf{s}^* \end{bmatrix} = W_{j+1} \begin{bmatrix} M_j \\ \mathbf{t}^* \end{bmatrix} \tag{10.25}$$

or

$$(A - \sigma_2 B)^{-1}BW_j = W_jM_j + \mathbf{w}_{j+1}\mathbf{t}^*. \tag{10.26}$$

This equation can easily been transformed into an Arnoldi or Krylov-Schur relation.

All these transformations can be executed without performing any operations on the large sparse matrices A and B.

In a practical implementation, the mentioned procedure is combined with locking, purging, and implicit restart. First run shifted and inverted Arnoldi with the first shift σ_1. When an appropriate number of eigenvalues around σ_1 have converged, lock these converged eigenvalues and purge those that are altogether outside the interesting region, leaving an Arnoldi (10.22) or Krylov-Schur recursion (10.22) for the remaining vectors. Then introduce the new shift σ_2 and perform the steps above to get a new basis W_{j+1} that replaces V_{j+1}. Start at the new shift by operating on the last vector of this new basis

$$\mathbf{r} := (A - \sigma_2 B)^{-1}B\mathbf{w}_{j+1}$$

and get the next basis vector w_{j+2} in the Arnoldi recurrence with the new shift σ_2. Continue until we get convergence for a set of eigenvalues around σ_2, and repeat the same procedure with new shifts until either all interesting eigenvalues have converged or all the shifts in the prescribed frequency range have been used.

Bibliography

[1] Å. Björck, *Numerics of Gram–Schmidt orthogonalization*, Linear Algebra Appl., 197/198 (1994), pp. 297–316.

[2] D. Calvetti, L. Reichel, and D. C. Sorensen, *An implicitely restarted Lanczos method for large symmetric eigenvalue problems*, Electron. Trans. Numer. Anal., 2 (1994), pp. 1–21.

[3] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems by Implicitely Restarted Arnoldi Methods*, SIAM, Philadelphia, PA, 1998. (The software and this manual are available at URL `http://www.caam.rice.edu/software/ARPACK/`).

[4] *The Matrix Market.* A repository of test data for use in comparative studies of algorithms for numerical linear algebra. Available at URL `http://math.nist.gov/MatrixMarket/`.

[5] A. Ruhe, *Rational Krylov subspace method*, in Templates for the solution of Algebraic Eigenvalue Problems: A Practical Guide, Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, eds., SIAM, Philadelphia, PA, 2000, pp. 246–249.

[6] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

[7] ——, *Implicitly restarted Arnoldi/Lanczos methods for large scale eigenvalue calculations*, in Parallel Numerical Algorithms, D. E. Keyes, A. Sameh, and V. Venkatakrishnan, eds., Kluwer, Dordrecht, 1997, pp. 119–165. (ICASE/LaRC Interdisciplinary Series in Science and Engineering, 4).

[8] G. W. Stewart, *A Krylov-Schur algorithm for large eigenproblems*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 601–614.

[9] K. Wu and H. D. Simon, *Thick-restart Lanczos method for large symmetric eigenvalue problems*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 602–616.