Chapter 12

The Jacobi-Davidson Method

The Lanczos and Arnoldi methods are very effective to compute extremal eigenvalues provided these are well separated from the rest of the spectrum. Lanczos and Arnoldi methods combined with a shift-and-invert spectral transformation are also efficient to compute eigenvalues in the vicinity of the shift σ. In this case it is necessary to solve a system of equation

$$(A - \sigma I)\mathbf{x} = \mathbf{y}, \qquad \text{or} \qquad (A - \sigma M)\mathbf{x} = \mathbf{y},$$

respectively, in each iteration step. These systems have to be solved very accurately since otherwise the Lanczos or Arnoldi relation does not hold anymore. In most cases the matrix $A - \sigma I$ (or $A - \sigma M$) is LU or Cholesky factored. The Jacobi–Davidson (JD) algorithm is particularly attractive if this factorization is not feasible [11].

12.1 The Davidson algorithm

Let $\mathbf{v}_1, \ldots, \mathbf{v}_m$ be a set of orthonormal vectors, spanning the *search space* $\mathcal{R}(V_m)$ with $V_m = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$. In the Galerkin approach we are looking for vectors $\mathbf{s} \in \mathbb{F}^m$ such that the *Galerkin condition* holds,

$$AV_m\mathbf{s} - \vartheta V_m\mathbf{s} \perp \mathbf{v}_1, \ldots, \mathbf{v}_m. \tag{12.1}$$

This immediately leads to the (small) eigenvalue problem

$$V_m^* A V_m \mathbf{s} = \vartheta V_m^* V_m \mathbf{s} \tag{12.2}$$

with solutions $(\vartheta_j^{(m)}, \mathbf{s}_j^{(m)})$, $j = 1, \ldots, m$. $\vartheta_j^{(m)}$ is called a Ritz value and $V_m\mathbf{s}_j^{(m)}$ is called a Ritz vector. In the sequel we omit the superscript m for readability. The dimension of the search space should become evident from the context.

Let us consider, say, the Ritz value ϑ_j, its Ritz vector $\mathbf{u}_j = V_m\mathbf{s}_j$ and their residual $\mathbf{r}_j = A\mathbf{u}_j - \vartheta_j\mathbf{u}_j$. Often we are looking for the largest or smallest eigenvalue of A in which case $j = 1$ or $j = m$, respectively. The question immediately arises how we can improve $(\vartheta_j, \mathbf{u}_j)$ if $\|\mathbf{r}_j\|$ is still too large. It is straightforward to try to find a better approximate eigenpair by *expanding* the search space. Davidson, in his original paper [2], suggested to compute a vector $\mathbf{t}$ from

$$(D_A - \vartheta_j I)\mathbf{t} = \mathbf{r}_j, \tag{12.3}$$

where D_A is the *diagonal* of the matrix A. The vector $\mathbf{t}$ is then made orthogonal to the basis vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$. The resulting vector, after normalization, is chosen as $\mathbf{v}_{m+1}$ by which $\mathcal{R}(V_m)$ is expanded, i.e., $V_{m+1} = [\mathbf{v}_1, \ldots, \mathbf{v}_m, \mathbf{v}_{m+1}]$.

This method is successful in finding dominant eigenvalues of (strongly) diagonally dominant matrices. The matrix $D_A - \vartheta_j I$ has therefore often been viewed as a preconditioner for the matrix $A - \vartheta_j I$. A number of investigations were made with more sophisticated preconditioners $M - \vartheta_j I$, see e.g. [7, 8]. They lead to the conclusion that $M - \vartheta_j I$ should not be too close to $A - \vartheta_j I$ which contradicts the notion of a preconditioner as being an easily invertible (factorizable) approximation of $A - \vartheta_j I$.

12.2 The Jacobi orthogonal component correction

In his seminal paper, Jacobi [6] not only presented the solution of symmetric eigenvalue problems by successive application of (later to be called) Jacobi rotations, but also presented an approach to improve an approximate eigenpair with an iterative procedure. Here, we give Jacobi's approach in a generalized form presented by Sleijpen and van der Vorst [11]. Let $\mathbf{u}_j$ be an approximation to the eigenvector $\mathbf{x}$ of A corresponding to the eigenvalue λ. Jacobi proposed to *correct* $\mathbf{u}_j$ by a vector $\mathbf{t}$, $\mathbf{u}_j \perp \mathbf{t}$, such that

$$A(\mathbf{u}_j + \mathbf{t}) = \lambda(\mathbf{u}_j + \mathbf{t}), \qquad \mathbf{u}_j \perp \mathbf{t}. \tag{12.4}$$

This is called the **Jacobi orthogonal component correction (JOCC)** by Sleijpen & van der Vorst [11]. As $\mathbf{t} \perp \mathbf{u}_j$ we may split equation (12.4) in the part parallel to $\mathbf{u}_j$ and in the part orthogonal to $\mathbf{u}_j$. If $\|\mathbf{u}_j\| = 1$ then the part parallel to $\mathbf{u}_j$ is

$$\mathbf{u}_j {\mathbf{u}_j}^* A(\mathbf{u}_j + \mathbf{t}) = \lambda \mathbf{u}_j {\mathbf{u}_j}^* (\mathbf{u}_j + \mathbf{t}) \tag{12.5}$$

which simplifies to the scalar equation

$$\vartheta_j + {\mathbf{u}_j}^* A \mathbf{t} = \lambda. \tag{12.6}$$

Here ϑ_j is the Rayleigh quotient of $\mathbf{u}_j$, $\vartheta_j = \rho(\mathbf{u}_j)$. The part orthogonal to $\mathbf{u}_j$ is

$$(I - \mathbf{u}_j {\mathbf{u}_j}^*) A(\mathbf{u}_j + \mathbf{t}) = \lambda (I - \mathbf{u}_j {\mathbf{u}_j}^*)(\mathbf{u}_j + \mathbf{t}) \tag{12.7}$$

which is equivalent to

$$\begin{aligned}(I - \mathbf{u}_j {\mathbf{u}_j}^*)(A - \lambda I)\mathbf{t} &= (I - \mathbf{u}_j {\mathbf{u}_j}^*)(-A\mathbf{u}_j + \lambda \mathbf{u}_j) \\ &= -(I - \mathbf{u}_j {\mathbf{u}_j}^*)A\mathbf{u}_j = -(A - \vartheta_j I)\mathbf{u}_j =: -\mathbf{r}_j.\end{aligned}$$

As $(I - \mathbf{u}_j {\mathbf{u}_j}^*)\mathbf{t} = \mathbf{t}$ we can rewrite this equation in symmetrized form as

$$(I - \mathbf{u}_j {\mathbf{u}_j}^*)(A - \lambda I)(I - \mathbf{u}_j {\mathbf{u}_j}^*)\mathbf{t} = -\mathbf{r}_j. \tag{12.8}$$

If A is symmetric then the matrix in (12.8) is symmetric as well.

Unfortunately, we do not know λ! Therefore, we replace λ by ϑ_j to get the **Jacobi–Davidson correction equation**

$$\boxed{(I - \mathbf{u}_j \mathbf{u}_j^*)(A - \vartheta_j I)(I - \mathbf{u}_j \mathbf{u}_j^*)\mathbf{t} = -\mathbf{r}_j = -(A - \vartheta_j I)\mathbf{u}_j, \qquad \mathbf{t} \perp \mathbf{u}_j.} \tag{12.9}$$

As $\mathbf{r}_j \perp \mathbf{u}_j$ (in fact $\mathbf{r}_j \perp \mathcal{V}_m$) this equation is consistent if $A - \vartheta_j I$ is nonsingular.

The correction equation (12.9) is, in general, solved iteratively by the GMRES or MINRES algorithm [1]. Often, only little accuracy in the solution is required.

Once $\mathbf{t}$ is (approximately) known we set

$$\mathbf{u}_{j+1} = \mathbf{u}_j + \mathbf{t}. \tag{12.10}$$

From (12.6) we may then obtain

$$\vartheta_{j+1} = \vartheta_j + \mathbf{u}_j{}^* A\mathbf{t}. \tag{12.11}$$

If A is symmetric ϑ_{j+1} may be set equal to the Rayleigh quotient $\rho(\mathbf{u_{j+1}})$.

Let us analyze (12.9) more closely. Let us first investigate the role of the orthogonality condition $\mathbf{t} \perp \mathbf{u}_j$. If this condition is omitted then the equation to be solved is

$$(I - \mathbf{u}_j\mathbf{u}_j^*)(A - \vartheta_j I)\mathbf{t} = -\mathbf{r}_j = -(A - \vartheta_j I)\mathbf{u}_j. \tag{12.12}$$

This equation has the solution $\mathbf{t} = -\mathbf{u}_j$. Therefore, without the condition $\mathbf{t} \perp \mathbf{u}_j$ there is no progress in solving the eigenvalue problem $A\mathbf{x} = \lambda\mathbf{x}$.

One can argue that this is the approach suggested by Davidson [2]. Davidson approximated A on the left side of (12.12) by an approximation of it, typically the diagonal, say D_A, of A. As his matrices were diagonally dominant, he solved a reasonably good approximation of (12.12). If D_A in (12.3) is considered a preconditioner of A then any matrix closer to A should lead to better performance of the algorithm. In extremis, A should be a possible choice for the matrix on the left. But we have just seen that this leads to a situation without progress. In fact the progess in the iteration deteriorates the better the 'preconditioner' approximates the system matrix. In consequence, D_A in (12.3) must *not* be considered a preconditioner.

Let us now investigate what happens if the correction equation is solved exactly. To that end we write it as

$$(I - \mathbf{u}_j\mathbf{u}_j^*)(A - \vartheta_j I)\mathbf{t} = -\mathbf{r}_j, \qquad \mathbf{t} \perp \mathbf{u}_j,$$

which immediately leads to

$$(A - \vartheta_j I)\mathbf{t} - \mathbf{u}_j \underbrace{\mathbf{u}_j^*(A - \vartheta_j I)\mathbf{t}}_{\alpha \in \mathbb{F}} = -\mathbf{r}_j,$$

or,

$$(A - \vartheta_j I)\mathbf{t} = \alpha\mathbf{u}_j - \mathbf{r}_j.$$

Assuming that ϑ_j is not an eigenvalue of A we get

$$\mathbf{t} = \alpha(A - \vartheta_j I)^{-1}\mathbf{u}_j - (A - \vartheta_j I)^{-1}\mathbf{r}_j.$$

The constraint $\mathbf{u}_j^*\mathbf{t} = 0$ allows us to determine the free variable α,

$$0 = \alpha\mathbf{u}_j^*(A - \vartheta_j I)^{-1}\mathbf{u}_j - \mathbf{u}_j^*(A - \vartheta_j I)^{-1}\mathbf{r}_j,$$

whence

$$\alpha = \frac{\mathbf{u}_j^*(A - \vartheta_j I)^{-1}\mathbf{r}_j}{\mathbf{u}_j^*(A - \vartheta_j I)^{-1}\mathbf{u}_j}.$$

By (12.10), the next approximate is then

$$\mathbf{u}_{j+1} = \mathbf{u}_j + \mathbf{t} = \mathbf{u}_j + \alpha(A - \vartheta_j I)^{-1}\mathbf{u}_j - \underbrace{(A - \vartheta_j I)^{-1}\mathbf{r}_j}_{\mathbf{u}_j} = \alpha(A - \vartheta_j I)^{-1}\mathbf{u}_j \tag{12.13}$$

which is a step of Rayleigh quotient iteration! This implies a fast (quadratic in general, cubic in the Hermitian case) convergence rate of this algorithm.

In general the correction equation

$$\tilde{A}\mathbf{t} = (I - \mathbf{u}_j\mathbf{u}_j^*)(A - \vartheta_j I)(I - \mathbf{u}_j\mathbf{u}_j^*)\mathbf{t} = -\mathbf{r}_j, \qquad \mathbf{t} \perp \mathbf{u}_j, \tag{12.14}$$

is solved iteratively with a Krylov space solver like GMRES or MINRES [1]. To get a decent performance a preconditioner is needed. Sleijpen and van der Vorst suggest preconditioners of the form

$$\tilde{K} = (I - \mathbf{u}_j\mathbf{u}_j^*)K(I - \mathbf{u}_j\mathbf{u}_j^*), \qquad K \approx A - \vartheta_j I. \tag{12.15}$$

We assume that K is (easily) invertible, i.e., that it is computationaly much cheaper to solve a system of equation with K than with A. With this assumption the system of equation

$$\tilde{K}\mathbf{z} = \mathbf{v}, \qquad \mathbf{z} \perp \mathbf{u}_j,$$

can be solved provided that the right-hand side $\mathbf{v}$ is in the range of $\tilde{K}$, i.e., provided that $\mathbf{v} \perp \mathbf{u}_j$. We formally denote the solution by $\mathbf{z} = \tilde{K}^+\mathbf{v}$. So, instead of (12.14) we solve the equation

$$\tilde{K}^+\tilde{A}\mathbf{t} = -\tilde{K}^+\mathbf{r}_j, \qquad \mathbf{t} \perp \mathbf{u}_j. \tag{12.16}$$

Let $\mathbf{t}_0 = \mathbf{0}$ be the initial approximation to the solution of (12.16). (Notice that $\mathbf{t}_0$ trivially satisfies the orthogonality constraint.) Because of the projectors $I - \mathbf{u}_j\mathbf{u}_j^*$ in the definitions of $\tilde{A}$ and $\tilde{K}$ all approximations are orthogonal to $\mathbf{u}_j$.

In each iteration step we have to compute

$$\mathbf{z} = \tilde{K}^+\tilde{A}\mathbf{v}, \qquad \mathbf{z} \perp \mathbf{u}_j$$

where $\mathbf{v} \perp \mathbf{u}_j$. To do this we proceed as follows. First we write

$$\tilde{A}\mathbf{v} = \underbrace{(I - \mathbf{u}_j\mathbf{u}_j^*)(A - \vartheta_j I)\mathbf{v}}_{\mathbf{y}} =: \mathbf{y}.$$

Then,

$$\tilde{K}\mathbf{z} = \mathbf{y}, \qquad \mathbf{z} \perp \mathbf{u}_j.$$

With (12.15) this becomes

$$(I - \mathbf{u}_j\mathbf{u}_j^*)K\mathbf{z} = K\mathbf{z} - \mathbf{u}_j\mathbf{u}_j^*K\mathbf{z} = \mathbf{y},$$

the solution of which is

$$\mathbf{z} = K^{-1}\mathbf{y} - \alpha K^{-1}\mathbf{u}_j,$$

where, formally, $\alpha = -\mathbf{u}_j^*K\mathbf{z}$. Similarly as earlier, we determine the scalar by means of the constraint $\mathbf{z}^*\mathbf{u}_j = 0$. Thus

$$\alpha = \frac{\mathbf{u}_j^*K^{-1}\mathbf{y}}{\mathbf{u}_j^*K^{-1}\mathbf{u}_j}.$$

Remark 12.1. Since $\mathbf{u}_j$ is fixed during the solution of the secular equation, the vector $K^{-1}\mathbf{u}_j$ has to be computed just once. Thus, if the iterative solver needs k steps until convergence, $k+1$ systems of equations have to be solved with the matrix K. □

Algorithm 12.1 The Jacobi–Davidson algorithm to compute the eigenvalue of A closest to a target value τ

```
 1: Let $A, B \in \mathbb{F}^{n\times n}$. This algorithm computes the eigenvalue of $A$ that is closest to $\tau$.
    Let $\mathbf{t}$ be an initial vector. Set $V_0 = []$, $V_0^A = []$, $m = 0$.
 2: loop
 3:   for $i = 1, \ldots, m-1$ do
 4:     $\mathbf{t} := \mathbf{t} - (\mathbf{v}_i^*\mathbf{t})\mathbf{v}_i$;          /* $\mathbf{t} = (I - V_{m-1}V_{m-1}^*)\mathbf{t}$ */
 5:   end for
 6:   $\mathbf{v}_m := \mathbf{t}/\|\mathbf{t}\|$;  $\mathbf{v}_m^A := A\mathbf{v}_m$;  $V_m := [V_{m-1}, \mathbf{v}_m]$;  $V_m^A := [V_{m-1}^A, \mathbf{v}_m^A]$;
 7:   for $i = 1, \ldots, m$ do
 8:     $M_{i,m} := \mathbf{v}_i^*\mathbf{v}_m^A$;  $M_{m,i} := \mathbf{v}_m^*\mathbf{v}_i^A$;          /* $M = V_m^*AV_m$ */
 9:   end for
10:   $M_{m,m} := \mathbf{v}_m^*\mathbf{v}_m^A$;
11:   Compute the eigenvalue $\vartheta$ of $M$ closest to $\tau$ and the          /* Rayleigh Ritz step */
      corresponding eigenvector $\mathbf{s}$: $M\mathbf{s} = \vartheta\mathbf{s}$;  $\|\mathbf{s}\| = 1$;
12:   $\mathbf{u} := V_m\mathbf{s}$;  $\mathbf{u}^A := V_m^A\mathbf{s}$;  $\mathbf{r} := \mathbf{u}^A - \vartheta\mathbf{u}$;
13:   if $\|\mathbf{r}\| <$ tol then
14:     return  $(\tilde\lambda = \vartheta,\ \tilde{\mathbf{x}} = \mathbf{u})$
15:   end if
16:   (Approximatively) solve the correction equation for $\mathbf{t}$,
          $(I - \mathbf{u}\mathbf{u}^*)(A - \vartheta_j I)(I - \mathbf{u}\mathbf{u}^*)\mathbf{t} = -\mathbf{r}$,    $\mathbf{t} \perp \mathbf{u}$;
17: end loop
```

12.2.1 Restarts

Evidently, in Algorithm 12.1, the dimension m of the search space can get large. To limit memory consumption, we limit m such that $m \le m_{\max}$. As soon as $m = m_{\max}$ we restart: $V_m = V_{m_{\max}}$ is replaced by the q Ritz vectors corresponding to the Ritz values closest to the target τ. Notice that the Schur decomposition of $M = M_{m,m} = V_m^*AV_m$ is computed already in step 11 of the Algorithm. Let $M = S^*TS$ be this Schur decomposition with $|t_{11} - \tau| \le |t_{22} - \tau| \le \cdots$. Then we set $V_q = V_m \cdot S_{:,1:q}$, $V_q^A = V_m^A \cdot S_{:,1:q}$, $M = T \cdot S_{1:q,1:q}$. Notice that the restart is easy because the Jacobi–Davidson algorithm is not a Krylov space method.

12.2.2 The computation of several eigenvalues

Let $\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_k$ be already computed eigenvectors or Schur vectors with $\tilde{\mathbf{x}}_i^*\tilde{\mathbf{x}}_j = \delta_{ij}$, $1 \le i, j \le k$. Then

$$AQ_k = Q_kT_k, \qquad Q_k = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_k]. \tag{12.17}$$

is a **partial Schur decomposition** of A [13]. We want to *extend* the partial Schur decomposition by one vector employing the Jacobi–Davidson algorithm. Since Schur vectors are mutually orthogonal we can apply the Jacobi–Davidson algorithm in the orthogonal complement of $\mathcal{R}(Q_k)$, i.e., we apply the Jacobi–Davidson algorithm to the matrix

$$(I - Q_kQ_k^*)A(I - Q_kQ_k^*), \qquad Q_k = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_k]. \tag{12.18}$$

The correction equation gets the form

$$(I - \mathbf{u}_j\mathbf{u}_j^*)(I - Q_kQ_k^*)(A - \vartheta_jI)(I - Q_kQ_k^*)(I - \mathbf{u}_j\mathbf{u}_j^*)\mathbf{t} = -\mathbf{r}_j, \qquad \mathbf{t} \perp \mathbf{u}_j, \mathbf{t} \perp Q_k. \tag{12.19}$$

As $\mathbf{u}_j \perp Q_k$ we have

$$(I - \mathbf{u}_j\mathbf{u}_j^*)(I - Q_kQ_k^*) = I - \tilde{Q}_k\tilde{Q}_k^*, \qquad \tilde{Q}_k = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_k, \mathbf{u}_j].$$

Thus, we can write (12.19) in the form

$$(I - \tilde{Q}_k\tilde{Q}_k^*)(A - \vartheta_jI)(I - \tilde{Q}_k\tilde{Q}_k^*)\mathbf{t} = -\mathbf{r}_j, \qquad \tilde{Q}_k^*\mathbf{t} = \mathbf{0}. \tag{12.20}$$

The preconditioner becomes

$$\tilde{K} = (I - \tilde{Q}_k\tilde{Q}_k^*)K(I - \tilde{Q}_k\tilde{Q}_k^*), \qquad K \approx A - \vartheta_jI. \tag{12.21}$$

Similarly as earlier, for solving

$$\tilde{K}\mathbf{z} = \tilde{A}\mathbf{v}, \qquad \tilde{Q}_k^*\mathbf{z} = \tilde{Q}_k^*\mathbf{v} = \mathbf{0},$$

we execute the following steps. Since

$$\tilde{A}\mathbf{v} = (I - \tilde{Q}_k\tilde{Q}_k^*)\underbrace{(A - \vartheta_jI)\mathbf{v}}_{\mathbf{y}} =: (I - \tilde{Q}_k\tilde{Q}_k^*)\mathbf{y} =: \mathbf{y} - \tilde{Q}_k\underbrace{\tilde{Q}_k^*\mathbf{y}}_{\mathbf{a}}.$$

we have to solve

$$\tilde{K}\mathbf{z} = (I - \tilde{Q}_k\tilde{Q}_k^*)K\mathbf{z} = (I - \tilde{Q}_k\tilde{Q}_k^*)\mathbf{y}, \qquad \mathbf{z} \perp \tilde{Q}_k.$$

Thus,

$$\mathbf{z} = K^{-1}\mathbf{y} - K^{-1}\tilde{Q}_k\mathbf{a}.$$

Similarly as earlier, we determine $\mathbf{a}$ by means of the constraint $\tilde{Q}_k^*\mathbf{z} = \mathbf{0}$,

$$\mathbf{a} = (\tilde{Q}_k^*K^{-1}\tilde{Q}_k)^{-1}\tilde{Q}_k^*K^{-1}\mathbf{y}.$$

If the iteration has converged to the vector $\tilde{\mathbf{x}}_{k+1}$ we can extend the partial Schur decomposition (12.17). Setting

$$Q_{k+1} := [Q_k, \tilde{\mathbf{x}}_{k+1}],$$

we get

$$AQ_{k+1} = Q_{k+1}T_{k+1} \tag{12.22}$$

with

$$T_{k+1} = \begin{bmatrix} T_k & Q_k^*A\tilde{\mathbf{x}}_{k+1} \\ 0 & \tilde{\mathbf{x}}_{k+1}^*A\tilde{\mathbf{x}}_{k+1} \end{bmatrix}.$$

12.2.3 Spectral shifts

In the correction equation (12.9) and implicitely in the preconditioner (12.15) a spectral shift ϑ_j appears. Experiments show that it is not wise to always choose the Rayleigh quotient of the recent eigenvector approximation $\mathbf{u}$ as the shift. In particular, far away from convergence, i.e., in the first few iteration steps, the Rayleigh quotient may be far away from the (desired) eigenvalue, and in fact may direct the JD iteration to an unwanted solution. So, one proceeds similarly as in the plain Rayleigh quotient iteration, cf. Remark 7.6 on page 143. Initially, the shift is held fixed, usually equal to the *target value* τ. As soon as the norm of the residual is small enough, the Rayleigh quotient of the actual approximate is chosen as the spectral shift in the correction equation. For efficiency

Algorithm 12.2 The Jacobi–Davidson QR algorithm to compute p of the eigenvalues closest to a target value τ

```
1: $Q_0 := []$;   $k = 0$.                                  /* Initializations */
2: Choose $\mathbf{v}_1$ with $\|\mathbf{v}_1\| = 1$.
3: $\mathbf{w}_1 = A\mathbf{v}_1$;   $H_1 := \mathbf{v}_1^*\mathbf{w}_1$;   $V_1 := [\mathbf{v}_1]$;   $W_1 := [\mathbf{W}_1]$;
4: $\tilde{\mathbf{q}} = \mathbf{v}_1$;   $\tilde{\vartheta} = \mathbf{v}_1^*\mathbf{w}_1$;   $\mathbf{r} := \mathbf{w}_1 - \tilde{\vartheta}\tilde{\mathbf{q}}$.
5: $j := 1$;
6: while $k < p$ do
7:                              /* Compute Schur vectors one after the other */
8:   Approximatively solve the correction equation for $\mathbf{t}$
       $(I - \tilde{Q}_k\tilde{Q}_k^*)(A - \tilde{\vartheta}I)(I - \tilde{Q}_k\tilde{Q}_k^*)\mathbf{t} = -\mathbf{r}_j, \qquad \tilde{Q}_k^*\mathbf{t} = \mathbf{0}.$
     where $\tilde{Q}_k = [Q_k, \tilde{\mathbf{q}}]$.
9:   $\mathbf{v}_j = (I - V_{j-1}V_{j-1}^*)\mathbf{t}/\|(I - V_{j-1}V_{j-1}^*)\mathbf{t}\|$;   $V_j := [V_{j-1}, \mathbf{v}_j]$.
10:  $\mathbf{w}_j = A\mathbf{v}_j$;   $H_j = \begin{bmatrix} H_{j-1} & V_{j-1}^*\mathbf{w}_j \\ \mathbf{v}_j^*W_{j-1} & \mathbf{v}_j^*\mathbf{w}_j \end{bmatrix}$;   $W_j = [W_{j-1}, \mathbf{w}_j]$.
11:  Compute the Schur decomposition of
       $H_j =: S_jR_jS_j$
     with the eigenvalues $r_{ii}^{(j)}$ sorted according to their distance to $\tau$.
12:  /* Test for convergence */
13:  repeat
14:    $\tilde{\vartheta} = \lambda_1^{(j)}$;   $\tilde{\mathbf{q}} = V_j\mathbf{s}_1$;   $\tilde{\mathbf{w}} = W_j\mathbf{s}_1$;   $\mathbf{r} = \tilde{\mathbf{w}} - \tilde{\vartheta}\tilde{\mathbf{q}}$
15:    found $:= \|\mathbf{r}\| < \varepsilon$
16:    if found then
17:      $Q_{k+1} = [Q_k, \tilde{\mathbf{q}}]$;   $k := k + 1$;
18:    end if
19:  until not found
20:  /* Restart */
21:  if $j = j_{\max}$ then
22:    $V_{j_{\min}} := V_j[\mathbf{s}_1, \ldots, \mathbf{s}_{\min}]$;   $T_{j_{\min}} := T_j(1 : j_{\min}, 1 : j_{\min})$;
23:    $H_{j_{\min}} := T_{j_{\min}}$;   $S_{j_{\min}} := I_{j_{\min}}$;   $J := j_{\min}$
24:  end if
25: end while
```

reasons, the spectral shift in the preconditioner K is always fixed. In this way it has to be computed just once. Notice that $\tilde{K}$ is changing with each correction equation.

Remark 12.2. As long as the shift is held fixed Jacobi–Davidson is actually performing a shift-and-invert Arnoldi iteration. □

Algorithm 12.2 gives the framework for an algorithm to compute the partial Schur decomposition of a matrix A. Q_k stores the converged Schur vectors; V_j stores the 'active' search space. This algorithm *does not* take into account some of the just mentioned issues. In particular the shift is always taken to be the Rayleigh quotient of the most recent approximate $\tilde{\mathbf{q}}$.

12.3 The generalized Hermitian eigenvalue problem

We consider the problem

$$A\mathbf{x} = \lambda M\mathbf{x}, \tag{12.23}$$

with A and M $n\times n$ Hermitian, and M additionally positive definite. Then the eigenvectors can be chosen mutually M-orthogonal,

$$\mathbf{x}_i^* M\mathbf{x}_j = \delta_{ij}, \qquad A\mathbf{x}_i = \lambda_i M\mathbf{x}_i, \quad 1 \le i, j \le n, \tag{12.24}$$

where δ_{ij} denotes the Kronecker delta function. Then it makes sense in the Jacobi–Davidson (as in other algorithms) to keep the iterates M-orthogonal.

Let $V_m = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$ be an M-orthogonal basis of the search space $\mathcal{V}_m$. Then the Galerkin condition

$$AV_m\mathbf{s} - \vartheta MV_m\mathbf{s} \perp \mathbf{v}_1, \ldots, \mathbf{v}_m, \tag{12.25}$$

leads to the eigenvalue problem

$$V_m^* AV_m\mathbf{s} = \vartheta V_m^* MV_m\mathbf{s} = \vartheta\mathbf{s}. \tag{12.26}$$

Let $(\tilde{\vartheta}, \tilde{\mathbf{u}} = V_m\tilde{\mathbf{s}})$ be a solution of (12.26). Then the correction $\mathbf{t}$ to $\tilde{\mathbf{u}}$ must be M-orthogonal,

$$\mathbf{t}^* M\tilde{\mathbf{u}} = 0 \quad \Longleftrightarrow \quad (I - \tilde{\mathbf{u}}\tilde{\mathbf{u}}^* M)\mathbf{t} = \mathbf{t}. \tag{12.27}$$

The correction equation in turn becomes

$$(I - M\tilde{\mathbf{u}}\tilde{\mathbf{u}}^*)(A - \tilde{\vartheta}M)(I - \tilde{\mathbf{u}}\tilde{\mathbf{u}}^* M)\mathbf{t} = -(I - \tilde{\mathbf{u}}\tilde{\mathbf{u}}^* M)\tilde{\mathbf{r}}, = -\tilde{\mathbf{r}}, \qquad \mathbf{t} \perp_M \tilde{\mathbf{u}}, \tag{12.28}$$

where $\tilde{\mathbf{r}} = A\tilde{\mathbf{u}} - \tilde{\vartheta}M\tilde{\mathbf{u}}$. Preconditioners for the secular equation are chosen of the form

$$\tilde{K} = (I - M\tilde{\mathbf{u}}\tilde{\mathbf{u}}^*)K(I - \tilde{\mathbf{u}}\tilde{\mathbf{u}}^* M), \tag{12.29}$$

where $K \approx A - \tau M$ and τ is the target value.

12.4 A numerical example

We give a demonstration on how a full-fledged Jacobi–Davidson algorithm works. The code is a MATLAB implementation of a program from the PhD thesis of Geus [4]. It solves the *generalized symmetric* eigenvalue problem as discussed in the previous section. The command `help jdsym` provides the output given on page 225.

As the numerical example we again consider the accustic behaviour in the interior of a car. We compute the five smallest eigenvalues and associated eigenvectors. The preconditioner is chosen to be the diagonal of A. An eigenpair $(\tilde{\lambda}, \tilde{\mathbf{q}}$ is declared converged if the residual norm $\|A\tilde{\mathbf{q}} - \tilde{\lambda}M\tilde{\mathbf{q}}\| < 10^{-8}\|\tilde{\mathbf{q}}\|$. Most of the components of `options` are explained in the help text. The residual norms for each iteration step are plotted in Fig. 12.1. As soon as an eigenpair has converged a new iteration starts. The residual norm then increases by several orders of magnitude.

```
[Q, lambda, it] = jdsym(n, A, B, K, kmax, tau, options)

jdsym is a MATLAB implementation of the JDQR algorithm for symmetric
matrices.

jdsym returns kmax eigenvalues with corresponding eigenvectors of
the matrix A near the target tau. K is a symmetric preconditioner
for A - tau * B.

The arguments A and B both contain either n-by-n symmetric matrices
or a string containing the name of an M-file which applies a
symmetric linear operator to the columns of a given
matrix. Matrix B must be positive definite.

To solve the specialized eigenvalue problem A * x = lambda * x pass
an empty matrix [] for parameter B. If no preconditioner is used
pass an empty matrix [] for parameter K.

 The options structure specifies certain parameters in the algorithm:

 options.tol      convergence tolerance                           1e-10
 options.jmax     maximal dimension of search subspace V          2*kmax
 options.jmin     dimension of search subspace V after restart    kmax
 options.maxit    maximum number of outer iterations              max(100,2*n/jmax)
 options.clvl     verbosity of output (0 means no output)         1
 options.eps_tr   tracing parameter as described in literature    1e-4
 options.toldecay convergence tolerance for inner iteration is    2
                  toldecay ^ (-solvestep)
 options.cgmaxit  maximum number of iterations in linear solver 100
 options.V0       initial search subspace                         rand(n,1)-.5
                  V0 will be orthonormalized by jdsym
 options.linsolv  solver used for corrections equation            1
                  1 -- CGS
                  2 -- SYMMLQ
                  3 -- CGS_OP
                  4 -- CGS mit SYMOP
                  5 -- MINRES
                  6 -- QMR
                  7 -- QMRS
 options.strategy strategy to avoid computation of zero
                  eigenvalues:
                  0 -- standard JD algorithm                      0
                  1 -- never choose Ritz values that are close
                       to zero as best current approximation.
                       Purge Ritz values that are close
                       to zero when restarting
                  2 -- dynamically adjust tau
                  3 -- see (1) and set tau to last converged
                       eigenvalue if it was bigger than the old
                       tau
                  4 -- set tau to last converged eigenvalue if
                       it was bigger than the old tau

The converged eigenvalues and eigenvectors are stored in Q and lambda. The
number of outer JD iterations performed is returned in it.
```

```
    >> K=diag(diag(A));
    >> options

    options =
         linsolv: 6
        strategy: 0

>> options.tol=1e-8

options =

        tol: 1.0000e-08
       jmax: 20
       jmin: 10
       clvl: 1
     optype: 1
    linsolv: 5

>> [Q, lambda, it] = jdsym(n, A, M, K, 5, -0.01, options);
JDSYM     Solving  A*x = lambda*M*x  with preconditioning

  N=            1095  ITMAX=1.095000e+02
  KMAX=  5  JMIN= 10  JMAX= 20  V0DIM=  1
  TAU=   -1.0000e-02  JDTOL=   1.0000e-08  STRATEGY=        0
  LINSOLVER=  MINRES  OPTYPE=         SYM
  LINITMAX=      100  EPS_TR=   1.000e-04  TOLDECAY= 2.00e+00

  IT   K   J       RES     CGTHET     CGTOL CGIT     CGERR CGFLG    Ritz values 1-5
---------------------------------------------------------------------------------
   0   0   1  4.26e+00
   1   0   2  9.33e-01  -1.00e-02  2.50e-01    1  9.74e-01     0
   2   0   3  7.13e-02  -1.00e-02  1.25e-01    4  6.95e-02     0
   3   0   4  4.14e-03  -1.00e-02  6.25e-02   10  4.04e-03     0
   4   0   5  2.01e-04  -1.00e-02  3.12e-02   33  1.22e-04     0
   5   0   6  4.79e-05  -1.00e-02  1.56e-02   71  3.07e-06     0
   6   0   7  3.66e-07   9.33e-08  7.81e-03   88  3.53e-07     0
   7   0   8  1.70e-09   6.39e-12  3.91e-03   74  1.34e-09     0
   7   1   7  5.94e-03
   8   1   8  4.98e-03  -1.00e-02  5.00e-01    4  2.67e-03     0
   9   1   9  2.53e-03  -1.00e-02  2.50e-01   11  1.19e-03     0
  10   1  10  3.38e-04  -1.00e-02  1.25e-01   18  3.06e-04     0
  11   1  11  4.76e-05  -1.00e-02  6.25e-02   27  2.05e-05     0
  12   1  12  1.45e-06   1.27e-02  3.12e-02   26  1.48e-06     0
  13   1  13  1.87e-08   1.27e-02  1.56e-02   38  2.22e-08     0
  14   1  14  9.87e-11   1.27e-02  7.81e-03   60  1.38e-10     0
  14   2  13  4.75e-03
  15   2  14  3.58e-03  -1.00e-02  5.00e-01    5  2.17e-03     0
  16   2  15  1.16e-03  -1.00e-02  2.50e-01    9  8.93e-04     0
  17   2  16  1.59e-04  -1.00e-02  1.25e-01   10  1.24e-04     0
  18   2  17  1.46e-05  -1.00e-02  6.25e-02   14  8.84e-06     0
  19   2  18  4.41e-07   4.44e-02  3.12e-02   21  4.29e-07     0
  20   2  19  7.01e-09   4.44e-02  1.56e-02   29  6.58e-09     0
  20   3  18  4.82e-03
  21   3  19  3.44e-03  -1.00e-02  5.00e-01    3  2.34e-03     0
  22   3  20  8.25e-04  -1.00e-02  2.50e-01    7  7.08e-04     0
  23    3  11  1.57e-04  -1.00e-02  1.25e-01   11  8.91e-05     0
  24    3  12  1.65e-05  -1.00e-02  6.25e-02   14  9.77e-06     0
  25    3  13  4.77e-07   5.66e-02  3.12e-02   31  4.68e-07     0
  26    3  14  6.51e-09   5.66e-02  1.56e-02   32  7.26e-09     0
  26    4  13  1.28e-02
  27    4  14  1.14e-02  -1.00e-02  5.00e-01    3  6.30e-03     0
  28    4  15  3.54e-03  -1.00e-02  2.50e-01    6  2.45e-03     0
  29    4  16  8.00e-04  -1.00e-02  1.25e-01   10  4.19e-04     0
  30    4  17  1.13e-04  -1.00e-02  6.25e-02   12  4.95e-05     0
  31    4  18  1.67e-05  -1.00e-02  3.12e-02   16  3.22e-06     0
  32    4  19  4.23e-07   1.17e-01  1.56e-02   21  2.49e-07     0
  33    4  20  3.20e-09   1.17e-01  7.81e-03   45  3.21e-09     0

JDSYM

IT_OUTER=33   IT_INNER_TOT=764   IT_INNER_AVG=   23.15

Converged eigensolutions in order of convergence:

  I               LAMBDA(I)      RES(I)
---------------------------------------
  1  9.102733263227557e-16  1.70111e-09
  2  1.269007628846320e-02  9.86670e-11
  3  4.438457596823515e-02  7.01153e-09
  4  5.663501055565738e-02  6.50940e-09
  5  1.166311652214006e-01  3.19504e-09
>>
```

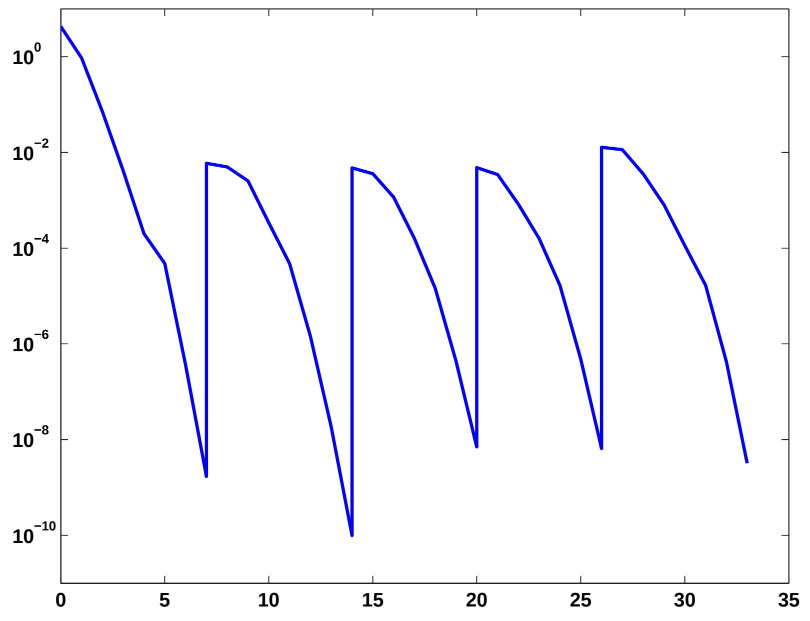

Figure 12.1: Jacobi–Davidson convergence history

12.5 The Jacobi–Davidson algorithm for interior eigenvalues

Interior eigenvalues are eigenvalues that do not lie at the 'border' of the convex hull of the spectrum, cf. Fig. 12.2

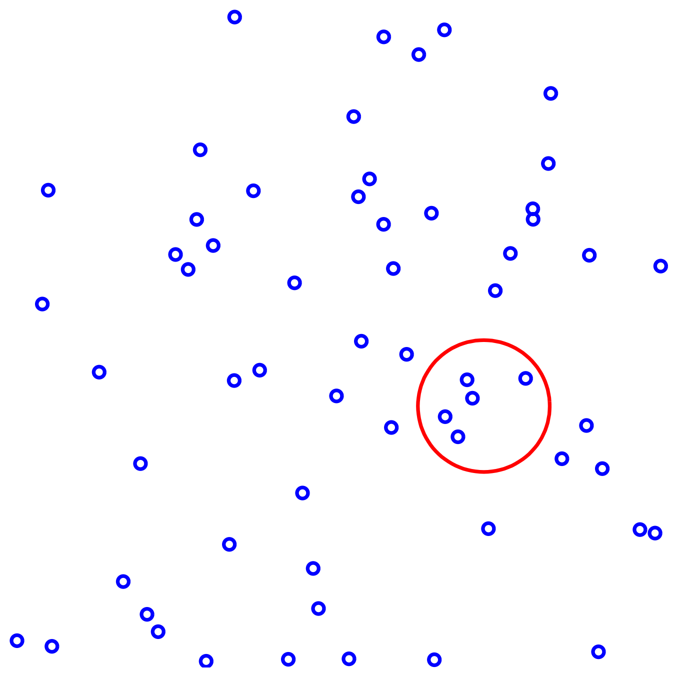

Figure 12.2: View of a spectrum $\sigma(A)$ in the complex plane. The eigenvalues in the red circle are to be computed

The success of the Jacobi–Davidson algorithm depends heavily on the quality of the actual Ritz pair $(\tilde{\vartheta}_j, \tilde{\mathbf{q}})$. However, the Rayleigh–Ritz procedure can lead to problem if it is applied to *interior* eigenvalues. The following simple numerical example shall demonstrate the problem. Let

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \qquad U = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{0.5} \\ 0 & \sqrt{0.5} \end{bmatrix}.$$

Then,

$$U^*AU = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \qquad U^*U = I_2.$$

So, any linear combination of the columns of U is a Ritz vector corresponding to the Ritz value 0, e.g.,

$$U \begin{pmatrix} \sqrt{0.5} \\ \sqrt{0.5} \end{pmatrix} = \begin{pmatrix} \sqrt{0.5} \\ 0.5 \\ 0.5 \end{pmatrix}.$$

Thus, although the basis contains the correct eigenvalue associated with the eigenvalue 0, the Rayleigh–Ritz procedure fails to find it and, instead, returns a very bad eigenvector approximation.

This example may look contrived. So, we conduct a MATLAB experiment with the same A but with a randomly perturbed U.

```
>> rand('state',0)
>> U1=U+1e-4*rand(size(U)); [U1,dummy]=qr(U1,0); U1=-U1
U1 =
    1.0000   -0.0001
    0.0000    0.7071
    0.0001    0.7071

>> B=U1'*A*U1

B =
   1.0e-04 *
   -0.0000   -0.2656
   -0.2656    0.1828

>> [X,L]=eig(B)

X =
   -0.8140   -0.5808
   -0.5808    0.8140
L =
   1.0e-04 *
   -0.1896         0
         0    0.3723

>> x=U1*-X(:,1)

x =
    0.8140
    0.4107
    0.4107

>> theta=L(1,1)

theta =
  -1.8955e-05

>> norm(A*x-x*theta)

ans =
    0.5808
```

We note that ϑ is a reasonable approximation for the eigenvalue 0. However, as the norm of the residual indicates, the Ritz vector is a bad approximation of the eigenvector.

12.6 Harmonic Ritz values and vectors

In the shift-and-invert Arnoldi algorithm the basic operator is $A - \sigma I$ where σ is some shift. The Arnoldi algorithm finds the largest eigenvalues of $A - \sigma I$, i.e., the eigenvalues of A closest to the shift. One of the reasons for inventing the Jacobi-Davidson algorithm is infeasibility of the factorization of $A - \sigma I$. Therefore, a shift-and-invert approach is not possible.

A clever way out of this dilemma works as follows: We apply the Ritz–Galerkin procedure with the matrix $(A - \sigma I)^{-1}$ and some subspace $\mathcal{R}(V) \subset \mathbb{F}^n$. This leads to the eigenvalues problem

$$V^*(A-\sigma I)^{-1}V\mathbf{s} = \mu V^*V\mathbf{s}. \tag{12.30}$$

The largest Ritz values μ_j approximate the largest eigenvalues of $(A-\sigma I)^{-1}$, i.e.,

$$\mu_j \approx \frac{1}{\lambda_j - \sigma} \Longleftrightarrow \lambda_j \approx \sigma + \frac{1}{\mu_j},$$

where λ_j is an eigenvalue of A close to the shift σ.

The trick is in the choice of V. Let us set $V := (A-\sigma I)U$. Then (12.30) becomes

$$U^*(A-\sigma I)^*U\mathbf{s} = \mu U^*(A-\sigma I)^*(A-\sigma I)U\mathbf{s}, \tag{12.31}$$

or, with $\tau = 1/\mu$,

$$U^*(A-\sigma I)^*(A-\sigma I)U\mathbf{s} = \tau U^*(A-\sigma I)^*U\mathbf{s}. \tag{12.32}$$

With $V = (A-\sigma I)U$ this becomes

$$V^*V\mathbf{s} = \tau V^*U\mathbf{s}. \tag{12.33}$$

If A is nonsymmetric, we compute an orthonormal basis $\tilde{V}$ of $V = (A-\sigma I)U$. Then we can write (12.32) in the nonsymmetric form

$$\tilde{V}^*(A-\sigma I)U\mathbf{s} = \tau\tilde{V}^*U\mathbf{s}. \tag{12.34}$$

We make the following

Definition 12.1 Let $(\tau, \mathbf{s})$ be an eigenpair of (12.32)–(12.34). Then the pair $(\sigma+\tau, U\mathbf{s})$ is called a **harmonic Ritz pair** of A with shift σ.

In practice, we are interested only in the harmonic Ritz pair corresponding to the smallest harmonic Ritz values. In the correction equation of the Jacobi–Davidson algorithm the harmonic Ritz vector is used as the latest eigenvector approximation and the harmonic Ritz values as the shift. In the symmetric case the harmonic Ritz value is replaced by the Rayleigh quotient of the harmonic Ritz vector $\mathbf{x}$, since

$$\|A\mathbf{x} - \rho(\mathbf{x})\mathbf{x}\| \le \|A\mathbf{x} - \mu\mathbf{x}\|, \quad \text{for all } \mu.$$

We continue the previous numerical example regarding the computation of the eigenvalue 0 of $A = \text{diag}(0, 1, -1)$

```
>> V=(A-theta*eye(3))*U1;
>> [v,l]=eig(V'*V, V'*U1)
v =
  -1.000000000000000  -1.000000000000000
   0.000059248824925  -0.713473633096137
l =
   1.0e+17 *
   0.000000000000000                   0
                   0  -1.970695224946170

>> theta + l(1,1)     %  Harmonic Ritz value
ans =
   3.722769433847084e-05
```

```
>> x = U1*v(:,1)   % Harmonic Ritz vector
x =
   1.000000001402380
  -0.000018783973233
   0.000018783630008
>> x'*A*x
ans =
   1.289413628670287e-14
```

The above considerations affect the Jacobi–Davidson algorithm in the extraction phase. Steps 11 and 14 in Algorithm 12.2 become

```
11: Compute the smallest eigenpair $(\tilde{\tau}, \tilde{\mathbf{s}})$ of
        $$(W_j^* - \bar{\sigma}V_j^*)(W_j - \sigma V_j)\mathbf{s} = \tau(W_j^* - \bar{\sigma}V_j^*)V_j\mathbf{s}.$$
14: Set $\tilde{\mathbf{q}} = V_j\tilde{\mathbf{s}}$, $\tilde{\mathbf{w}} = W_j\tilde{\mathbf{s}}$.    $\tilde{\vartheta} = \sigma + \tau$ or $\tilde{\vartheta} = \tilde{\mathbf{q}}^* A\tilde{\mathbf{q}}/\tilde{\mathbf{q}}^*\tilde{\mathbf{q}}$.
```

To solve the eigenvalue problem (12.34) the QZ algorithm has to be employed, see section 12.8. In the symmetric case (12.33) the symmetric QR algorithm will suffice in general since the matrix on the left is positive definite.

12.7 Refined Ritz vectors

An alternative to harmonic Ritz vectors are refined Ritz vectors [13]. Again we start from the observation that the Ritz values were of good quality. What we need are improved Ritz vectors. Stewart [13] suggested the following procedure.

Definition 12.2 Let μ_ϑ be a Ritz value of A restricted to U_ϑ. A solution of the minimization problem

$$\min_{\hat{\mathbf{x}} \in \mathbf{U}_\vartheta, \|\hat{\mathbf{x}}\|=\mathbf{1}} \|A\hat{\mathbf{x}} - \mu_\vartheta\hat{\mathbf{x}}\| \tag{12.35}$$

is called a **refined Ritz vector**.

How is this minimization problem solved? We write $\hat{\mathbf{x}} = U_\vartheta\mathbf{z}$. Then (12.35) becomes

$$\min_{\|\mathbf{z}\|=1} \|(A - \mu_\vartheta I)U_\vartheta\mathbf{z}\|. \tag{12.36}$$

This minimization problem is solved by the right singular vector corresponding to the smallest singular value of $(A - \mu_\vartheta I)U_\vartheta$ or, equivalently, the eigenvector corresponding to the smallest eigenvalue of

$$U_\vartheta^*(A - \mu_\vartheta I)^*(A - \mu_\vartheta I)U_\vartheta\mathbf{z} = \tau\mathbf{z}.$$

We continue the example of before.

```
>> [u,s,v]=svd((A - 0*eye(3))*U)

u =
          0    1.0000         0
    -0.7071         0    0.7071
     0.7071         0    0.7071

s =
     1.0000         0
          0         0
          0         0

v =
      0     1
     -1     0

>> U*v(:,2)

ans =
      1
      0
      0

>> [u,s,v]=svd((A - L(1,1)*eye(3))*U1)

u =
    -0.0000    0.5810    0.8139
    -0.7071   -0.5755    0.4108
     0.7071   -0.5755    0.4108

s =
     1.0001         0
          0    0.0000
          0         0

v =
    -0.0001    1.0000
    -1.0000   -0.0001

>> format long
>> U1*v(:,2)

ans =
   1.00009500829405
  -0.00001878470226
   0.00001878647014
```

With the refined Ritz vector approach Steps 11 and 14 in Algorithm 12.2 are replaced by

```
11: Compute the Ritzpair $(\tilde{\vartheta}, \tilde{\mathbf{q}})$ of $A$ closest to the target value.
    Compute the smallest singular vector $\tilde{\mathbf{s}}$ of $AV_j - \tilde{\vartheta}V_j$.
14: Replace $\tilde{\mathbf{q}}$ by $V_j\tilde{\mathbf{s}}$.
```

12.8 The generalized Schur decomposition

The QZ algorithm computes the following generalized Schur decomposition.

Theorem 12.3 ***(Generalized Schur decomposition)*** *If $A, B \in \mathbb{C}^{n\times n}$ then there are unitary matrices $Q, Z \in \mathbb{C}^{n\times n}$ such that*

$$Q^*AZ = T^A, \qquad Q^*BZ = T^B, \tag{12.37}$$

are upper triangular. If for some k, $t^A_{kk} = t^B_{kk} = 0$ then $\sigma(A, B) = \mathbb{C}$. Otherwise

$$\sigma(A, B) = \{t^A_{ii}/t^B_{ii} \mid t^B_{ii} \neq 0\}.$$

Proof. See [5] ■

The algorithm starts out with transforming A and B in Hessenberg and upper triangular form, respectively. After defalting zeros in the lower offdiagonal of the Hessenberg matrix and on the diagonal of the upper triangular matrix, the QR algorithm with implicit shifts is applied to AB^{-1}. For details see [5].

Corresponding to the notion of an invariant subspace for a single matrix we have the notion of a **deflating subspace** for the **pencil** $A - \lambda B$. In particular, we say that a k-dimensional subspace $\mathcal{S} \subset \mathbb{F}^n$ is "deflating" for the pencil $A - \lambda B$ if the subspace $\{A\mathbf{x} + B\mathbf{y} \mid \mathbf{x}, \mathbf{y} \in \mathcal{S}\}$ has dimension k or less. Note that the columns of the matrix Z in the generalized Schur decomposition define a family of deflating subspaces, for if $Q = [\mathbf{q}_1, \ldots, \mathbf{q}_n]$ and $Z = [\mathbf{z}_1, \ldots, \mathbf{z}_n]$ then we have $\text{span}\{A\mathbf{z}_1, \ldots, A\mathbf{z}_k\} \subset \text{span}\{\mathbf{q}_1, \ldots, \mathbf{q}_k\}$ and $\text{span}\{B\mathbf{z}_1, \ldots, B\mathbf{z}_k\} \subset \text{span}\{\mathbf{q}_1, \ldots, \mathbf{q}_k\}$.

12.9 JDQZ: Computing a partial QZ decomposition by the Jacobi–Davidson algorithm

We now consider the generalized eigenvalue problem

$$A\mathbf{x} = \lambda B\mathbf{x}, \tag{12.38}$$

with *arbitrary* A and B. There is a variant of Jacobi–Davidson called JDQZ that computes a *partial* QZ decomposition of the stencil (A, B). This section follows closely the corresponding section in the eigenvalue templates [12]. Further details are found in [3].

With $\lambda = \alpha/\beta$, the generalized eigenproblem (12.38) is equivalent to the eigenproblem

$$(\beta A - \alpha B)\mathbf{x} = 0, \tag{12.39}$$

where we denote a generalized eigenvalue of the matrix pair $\{A, B\}$ as a pair (α, β). The notation (12.39) is preferred over (12.40), because underflow or overflow for $\lambda = \alpha/\beta$ in finite precision arithmetic may occur when α and/or β are zero or close to zero. It also emphazises the symmetry of the roles of A and B.

A **partial generalized Schur** form of dimension k for a matrix pair $\{A, B\}$ is the decomposition

$$AQ_k = Z_k R^A_k, \quad BQ_k = Z_k R^B_k, \tag{12.40}$$

where Q_k and Z_k are unitary $n \times k$ matrices and R^A_k and R^B_k are upper triangular $k \times k$ matrices. A column $\mathbf{q}_i$ of Q_k is referred to as a generalized Schur vector, and we refer to a pair $((\alpha_i, \beta_i), \mathbf{q}_i)$, with $(\alpha_i, \beta_i) = (R_k^A(i,i), R_k^B(i,i))$ as a generalized Schur pair. It follows that if $((\alpha, \beta), \mathbf{y})$ is a generalized eigenpair of (R_k^A, R_k^B) then $((\alpha, \beta), Q_k\mathbf{y})$ is a generalized eigenpair of $\{A, B\}$.

From the relations (12.40) we see that

$$\beta_i A\mathbf{q}_i - \alpha_i B\mathbf{q}_i \perp \mathbf{z}_i.$$

This somewhat resembles the Schur decomposition, where $A\mathbf{q}_i - \lambda_i\mathbf{q}_i \perp \mathbf{q}_i$. The $\mathbf{z}_i$ on the right hand side suggests that we should follow a *Petrov-Galerkin condition* for the construction of reduced systems. In each step the approximate eigenvector $\mathbf{u}$ is selected from a j-dimensional search subspace $\operatorname{span}(V_j) = \operatorname{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_j\}$. We require that the residual $\eta A\mathbf{u} - \zeta B\mathbf{u}$ is orthogonal to some *other* well-chosen test subspace $\operatorname{span}(W_j) = \operatorname{span}\{\mathbf{w}_1, \ldots, \mathbf{w}_j\}$,

$$\eta\, A\mathbf{u} - \zeta\, B\mathbf{u} \perp \operatorname{span}(W_j). \tag{12.41}$$

Equation (12.41) leads to the projected generalized $j \times j$ eigenproblem

$$(\eta\, W_j^* AV_j - \zeta\, W_j^* BV_j)\, \mathbf{s} = 0. \tag{12.42}$$

The j-dimensional pencil $\eta\, W_j^* AV_j - \zeta\, W_j^* BV_j$ can be reduced by the QZ algorithm (see §12.8) to generalized Schur form. This leads to orthogonal $j \times j$ matrices S^R and S^L and upper triangular $j \times j$ matrices T^A and T^B, such that

$$(S^L)^*(W_j^* AV_j)S^R = T^A \quad \text{and} \quad (S^L)^*(W_j^* BV_j)S^R = T^B. \tag{12.43}$$

This decomposition can be reordered such that the first column of S^R and the $(1,1)$-entries of T^A and T^B represent the wanted Petrov solution [3]. With $\mathbf{s} := \mathbf{s}_1^R := S^R\mathbf{e}_1$ and $\zeta := T_{1,1}^A$, $\eta := T_{1,1}^B$, the Petrov vector is defined as

$$\mathbf{u} := V_j\mathbf{s} = V_j\mathbf{s}_1^R$$

for the associated generalized Petrov value (ζ, η). In an analogous way we can define a *left* Petrov vector as

$$\mathbf{p} := W_j\mathbf{s}_1^L \qquad \mathbf{s}_1^L := S^L\mathbf{e}_1$$

If V_j and W_j are unitary, as in Algorithm 12.3, then $\|\mathbf{s}^R\|_2 = \|\mathbf{s}^L\|_2 = 1$ implies $\|\mathbf{u}\|_2 = 1$.

With the decomposition in (12.43), we construct an approximate partial generalized Schur form (cf. (12.40)): V_jS^R approximates a Q_k, and W_jS^L approximates the associated Z_j.

It is not yet clear how to choose the test space W_j. The equations $\operatorname{span}(Z_j) = \operatorname{span}(AQ_j) = \operatorname{span}(BQ_j)$, cf. (12.40), suggest to choose W_j such that $\operatorname{span}(W_j)$ coincides with $\operatorname{span}(\nu_0 AV_j + \mu_0 BV_j)$ for some suitably chosen ν_0 and μ_0. With the weights ν_0 and μ_0 we can influence the convergence of the Petrov values. If we want eigenpair approximations for eigenvalues λ close to a target τ, then the choice

$$\nu_0 = 1/\sqrt{1 + |\tau|^2}, \qquad \mu_0 = -\tau\nu_0$$

is very effective [3], especially if we want to compute eigenvalues in the interior of the spectrum of $A - \lambda B$. We will call the Petrov approximations for this choice the harmonic Petrov eigenpairs. The Jacobi-Davidson correction equation for the component $\mathbf{t} \perp \mathbf{u}$ for the pencil $\eta A - \zeta B$ becomes

$$(I - \mathbf{p}\mathbf{p}^*)\,(\eta A - \zeta B)\,(I - \mathbf{u}\mathbf{u}^*)\,\mathbf{t} = -\mathbf{r}, \qquad \mathbf{r} := \eta A\mathbf{u} - \zeta B\mathbf{u}. \tag{12.44}$$

Sleijpen et al. [10] have shown that if (12.44) is solved exactly, the convergence to the generalized eigenvalue is quadratic. Usually, this correction equation is solved only approximately, for instance, with a (preconditioned) iterative solver. The obtained vector **t** is used for the expansion **v** of V_j and $\nu_0 A\mathbf{v} + \mu_0 B\mathbf{v}$ is used for the expansion of W_j. For both spaces we work with orthonormal bases. Therefore, the new columns are orthonormalized with respect to the current basis by a modified Gram-Schmidt orthogonalization process.

12.9.1 Restart

Suppose that the generalized Schur form (12.43) is ordered with respect to τ such that

$$|T^A_{1,1}/T^B_{1,1} - \tau| \le |T^A_{2,2}/T^B_{2,2} - \tau| \le \cdots \le |T^A_{j,j}/T^B_{j,j} - \tau|,$$

where j is the dimension of span(V_j). Then, for $i < j$, the space span$(V_j\mathbf{s}^R_1, \ldots, V_j\mathbf{s}^R_i)$ spanned by the first i columns of $V_j S^R$ contains the i most promising Petrov vectors. The corresponding test subspace is given by span$(W_j\mathbf{s}^L, \ldots, W\mathbf{s}^L_i)$. Therefore, in order to reduce the dimension of the subspaces ("implicit restart") to $j_{\min}$, $j_{\min} < j$, the columns $\mathbf{v}_{j_{\min}+1}$ through $\mathbf{v}_j$ and $\mathbf{w}_{j_{\min}+1}$ through $\mathbf{w}_j$ can simply be discarded and the Jacobi-Davidson algorithm can be continued with

$$V = [V\mathbf{s}^R_1, \ldots, V\mathbf{s}^R_{j_{\min}}] \quad \text{and} \quad W = [W\mathbf{s}^L_1, \ldots, W\mathbf{s}^L_{j_{\min}}].$$

12.9.2 Deflation

Like in the Jacobi-Davidson algorithm for the standard eigenvalue problem, in the Jacobi-Davidson process for the generalized eigenvalue problem found (converged) Ritz (here Petrov) vectors can be *deflated*.

The partial generalized Schur form can be obtained in a number of successive steps. Suppose that we have already available the partial generalized Schur form $AQ_{k-1} = Z_{k-1}R^A_{k-1}$ and $BQ_{k-1} = Z_{k-1}R^B_{k-1}$. We want to expand this partial generalized Schur form with the new right Schur vector **u** and the left Schur vector **p** to

$$A[Q_{k-1}\mathbf{u}] = [Z_{k-1}\mathbf{p}]\begin{bmatrix} R^A_{k-1} & \mathbf{a} \\ 0 & \alpha \end{bmatrix}$$

and

$$A[Q_{k-1}\mathbf{u}] = [Z_{k-1}\mathbf{p}]\begin{bmatrix} R^B_{k-1} & \mathbf{b} \\ 0 & \beta \end{bmatrix}$$

The new generalized Schur pair $((\alpha, \beta), \mathbf{u})$ satisfies

$$Q^*_{k-1}\mathbf{u} = \mathbf{0} \quad \text{and} \quad (\beta A - \alpha B)\mathbf{u} - Z_{k-1}(\beta\mathbf{a} - \alpha\mathbf{b}) = \mathbf{0},$$

or, since $\beta\mathbf{a} - \alpha\mathbf{b} = Z^*_{k-1}(\beta A - \alpha B)\mathbf{u}$,

$$Q^*_{k-1}\mathbf{u} = \mathbf{0} \quad \text{and} \quad \left(I - Z_{k-1}Z^*_{k-1}\right)(\beta A - \alpha B)\left(I - Q_{k-1}Q^*_{k-1}\right)\mathbf{u} = \mathbf{0}.$$

Hence, the vectors **a** and **b** can be computed from

$$\mathbf{a} = Z^*_{k-1}A\mathbf{u} \quad \text{and} \quad \mathbf{b} = Z^*_{k-1}B\mathbf{u}.$$

Furthermore, the generalized Schur pair $((\alpha,\beta),\mathbf{u})$ is an eigenpair of the deflated matrix pair

$$\left(\left(I-Z_{k-1}Z_{k-1}^*\right)A\left(I-Q_{k-1}Q_{k-1}^*\right),\left(I-Z_{k-1}Z_{k-1}^*\right)B\left(I-Q_{k-1}Q_{k-1}^*\right)\right).$$

This eigenproblem can be solved again with the Jacobi-Davidson QZ process. In that process we construct vectors v_i that are orthogonal to Q_{k-1} and vectors w_i that are orthogonal to Z_{k-1}. This simplifies the computation of the interaction matrices M^A and M^B, associated with the deflated operators

$$\begin{cases} M^A \equiv W^*\left(I-Z_{k-1}Z_{k-1}*\right)A\left(I-Q_{k-1}Q_{k-1}*\right)V = W^*AV, \\ M^A \equiv W^*\left(I-Z_{k-1}Z_{k-1}*\right)B\left(I-Q_{k-1}Q_{k-1}*\right)V = W^*BV, \end{cases}$$

and M^A and M^B can be simply computed as W^*AV and W^*BV, respectively.

12.9.3 Algorithm

The Jacobi-Davidson algorithm to compute a partial QZ decomposition for a general matrix pencil (A,B) is given in Algorithm 12.3 This algorithm attempts to compute the generalized Schur pairs $((\alpha,\beta),q)$, for which the ratio β/α is closest to a specified target value τ in the complex plane. The algorithm includes restart in order to limit the dimension of the search space, and deflation with already converged left and right Schur vectors.

To apply this algorithm we need to specify a starting vector v_0, a tolerance ϵ, a target value τ, and a number $k_{\max}$ that specifies how many eigenpairs near τ should be computed. The value of $j_{\max}$ specifies the maximum dimension of the search subspace. If it is exceeded then a restart takes place with a subspace of dimension $j_{\min}$.

On completion the $k_{\max}$ generalized eigenvalues close to τ are delivered, and the corresponding reduced Schur form $AQ = ZR^A$, $BQ = ZR^B$, where Q and Z are n by $k_{\max}$ orthogonal and R^A, R^B are $k_{\max}$ by $k_{\max}$ upper triangular. The generalized eigenvalues are the on-diagonals of R^A and R^B. The computed form satisfies $\|A\mathbf{q}_j - ZR^Ae_j\|_2 = O(\epsilon)$, $\|B\mathbf{q}_j - ZR^Be_j\|_2 = O(\epsilon)$, where $\mathbf{q}_j$is the jth column of Q.

12.10 Jacobi-Davidson for nonlinear eigenvalue problems

Nonlinear eigenvalue problems have the form

$$T(\lambda)\mathbf{x} = \mathbf{0} \tag{12.45}$$

where the $n\times n$ matrix $T(\lambda)$ has elements that depend on the scalar parameter λ. For the *linear* eigenvalue problem $T(\lambda) = A-\lambda B$. λ is an eigenvalue of (12.45) if $T(\lambda)$ is singular; a nontrivial solution $\mathbf{x}$ of the singular linear system is a corresponding eigenvector.

For small problems, Newton iteration is applicable. Ruhe [9] suggests to proceed as follows. Complement (12.45) by a normalization condition

$$\mathbf{v}^*\mathbf{x} = 1. \tag{12.46}$$

Then, we solve

$$P\begin{pmatrix}\mathbf{x}\\ \lambda\end{pmatrix} = \begin{pmatrix}T(\lambda)\mathbf{x}\\ \mathbf{v}^*\mathbf{x}-1\end{pmatrix} = \begin{pmatrix}\mathbf{0}\\ 0\end{pmatrix}. \tag{12.47}$$

Algorithm 12.3 Jacobi–Davidson QZ method for $k_{\max}$ interior eigenvalues close to τ for the generalized non-Hermitian eigenvalue problem

```
1: Let $A, B \in \mathbb{F}^{n\times n}$ be non-Hermitian. This algorithm computes $k_{\max}$ interior eigenvalues
   of $\alpha A\mathbf{x} = \beta B\mathbf{x}$ close to the target $\tau$.
2: $\mathbf{t} = \mathbf{v}_0$; $k = 0$; $\nu_0 = 1/\sqrt{1+|\tau|^2}$; $\mu_0 = -\tau\nu_0$; $m = 0$;
3: $Q = []$; $Z = []$; $S = []$; $T = []$;
4: while $k < k_{\max}$ do
5:    Orthogonalize $\mathbf{t} := \mathbf{t} - V_m V_m^* \mathbf{t}$
6:    $m = m+1$; $\mathbf{v}_m = \mathbf{t}/\|\mathbf{t}\|$; $\mathbf{v}_m^A := A\mathbf{v}_m$; $\mathbf{v}_m^B := B\mathbf{v}_m$; $\mathbf{w} := \nu_0\mathbf{v}_m^A + \mu_0\mathbf{v}_m^B$;
7:    Orthogonalize $\mathbf{w} := \mathbf{w} - Z_k Z_k^* \mathbf{w}$
8:    Orthogonalize $\mathbf{w} := \mathbf{w} - W_{m-1} W_{m-1}^* \mathbf{w}$
9:    $\mathbf{w}_m = \mathbf{w}/\|\mathbf{w}\|$;
10:
      $$M^A := \begin{bmatrix} M^A & W_{m-1}^*\mathbf{v}_m^A \\ \mathbf{w}_m^* V_{m-1}^A & \mathbf{w}_m^*\mathbf{v}_m^A \end{bmatrix}; \qquad M^B := \begin{bmatrix} M^B & W_{m-1}^*\mathbf{v}_m^B \\ \mathbf{w}_m^* V_{m-1}^B & \mathbf{w}_m^*\mathbf{v}_m^B \end{bmatrix};$$
11:   Compute the QZ decomposition $M^A S^R = S^L T^A$, $M^B S^R = S^L T^B$, such that
      $|T_{i,i}^A/T_{i,i}^B - \tau| \le |T_{i+1,i+1}^A/T_{i+1,i+1}^B - \tau|$          /* Rayleigh Ritz step */
12:   $\mathbf{u} := V\mathbf{s}_1^R$; $\mathbf{p} := W_j\mathbf{s}_1^L$; $\mathbf{u}^A := V^A\mathbf{s}_1^R$; $\mathbf{u}^B := V^B\mathbf{s}_1^R$; $\zeta = T_{1,1}^A$; $\eta = T_{1,1}^B$;
13:   $\mathbf{r} = \eta\mathbf{u}^A - \zeta\mathbf{u}^B$; $\tilde{\mathbf{a}} = Z^*\mathbf{u}^A$; $\tilde{\mathbf{b}} = Z^*\mathbf{u}^B$; $\tilde{\mathbf{r}} = \mathbf{r} - Z(\eta\tilde{\mathbf{a}} - \zeta\tilde{\mathbf{b}})$;
14:   while $\|\tilde{\mathbf{r}}\| < \varepsilon$ do
15:
         $$R^A := \begin{bmatrix} R^A & \tilde{\mathbf{a}} \\ \mathbf{0}^T & \zeta \end{bmatrix}; \qquad R^B := \begin{bmatrix} R^B & \tilde{\mathbf{b}} \\ \mathbf{0}^T & \eta \end{bmatrix};$$
16:      $Q := [Q, \mathbf{u}]$; $Z := [Z, \mathbf{p}]$; $k := k+1$;
17:      if $k = k_{\max}$ then
18:         return $(Q, Z, R^A, R^B)$
19:      end if
20:      $m := m-1$;
21:      for $i = 1, \ldots, m$ do
22:         $\mathbf{v}_i := V\mathbf{s}_{i+1}^R$; $\mathbf{v}_i^A := V^A\mathbf{s}_{i+1}^R$; $\mathbf{v}_i^B := V^B\mathbf{s}_{i+1}^R$;
23:         $\mathbf{w}_i := W\mathbf{s}_{i+1}^L$; $\mathbf{s}_i^R := \mathbf{s}_i^L := \mathbf{e}_i$;
24:      end for
25:      $M^A$, $M^B$ is the lower $m \times m$ block of $T^A$, $T^B$, resp.
26:      $\mathbf{u} := \mathbf{u}_1$; $\mathbf{p} := \mathbf{w}_1$; $\mathbf{u}^A := \mathbf{v}_1^A$; $\mathbf{u}^B := \mathbf{v}_1^b$; $\zeta = T_{1,1}^A$; $\eta = T_{1,1}^B$;
27:      $\mathbf{r} = \eta\mathbf{u}^A - \zeta\mathbf{u}^B$; $\tilde{\mathbf{a}} = Z^*\mathbf{u}^A$; $\tilde{\mathbf{b}} = Z^*\mathbf{u}^B$; $\tilde{\mathbf{r}} = \mathbf{r} - Z(\eta\tilde{\mathbf{a}} - \zeta\tilde{\mathbf{b}})$;
28:   end while
29:   if $m \ge m_{\max}$ then
30:      for $i = 2, \ldots, m_{\min}$ do
31:         $\mathbf{v}_i := V\mathbf{s}_i^R$; $\mathbf{v}_i^A := V^A\mathbf{s}_i^R$; $\mathbf{v}_i^B := V^B\mathbf{s}_i^R$; $\mathbf{w}_i := W\mathbf{s}_i^L$;
32:      end for
33:      $M^A$, $M^B$ is the leading $m_{\min} \times m_{\min}$ block of $T^A$, $T^B$, resp.
34:      $\mathbf{v}_1 := \mathbf{u}$; $\mathbf{v}_1^A := \mathbf{u}^A$; $\mathbf{v}_1^B := \mathbf{u}^B$; $\mathbf{w}_1 := \mathbf{p}$; $m := m_{\min}$
35:   end if
36:   $\tilde{Q} := [Q, \mathbf{u}]$; $\tilde{Z} := [Z, \mathbf{p}]$;
37:   (Approximatively) solve the correction equation for $\mathbf{t} \perp \tilde{Q}$,
38:         $(I - \tilde{Z}\tilde{Z}^*)(\eta A - \zeta B)(I - \tilde{Q}\tilde{Q}^*)$
39: end while
```

For the derivative of P we obtain

$$P' = \begin{bmatrix} T(\lambda) & T'(\lambda)\mathbf{x} \\ \mathbf{v}^* & 0 \end{bmatrix}$$

such that the Newton iteration becomes

$$\begin{pmatrix} \mathbf{x}_{s+1} \\ \lambda_{s+1} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_s \\ \lambda_s \end{pmatrix} - \begin{bmatrix} T(\lambda_s) & T'(\lambda_s)\mathbf{x}_s \\ \mathbf{v}_s^* & 0 \end{bmatrix}^{-1} \begin{pmatrix} T(\lambda_s)\mathbf{x}_s \\ \mathbf{v}_s^*\mathbf{x}_s - 1 \end{pmatrix} \tag{12.48}$$

or

$$\begin{aligned} T(\lambda_s)\mathbf{u}_{s+1} &= T'(\lambda_s)\mathbf{x}_s, \\ \lambda_{s+1} &= \lambda_s - (\mathbf{v}_s^*\mathbf{x}_s)/(\mathbf{v}_s^*\mathbf{x}_{s+1}), \\ \mathbf{x}_{s+1} &= C \cdot \mathbf{u}_{s+1}. \end{aligned} \tag{12.49}$$

Here, C is some normalization constant. The vector $\mathbf{v}_s$ may depend on the iteration step. It can be chosen in a number of ways. It could be constant, e.g., $\mathbf{v}_s = \mathbf{e}_i$. This amounts to keeping one of the entries of $\mathbf{x}_s$ constant. Another choce is

$$\mathbf{v}_s = T(\lambda_s)^*\mathbf{y}_s$$

where $\mathbf{y}_s$ is an approximation to the left eigenvector $\mathbf{y}$.

A Jacobi-Davidson algorithm for large nonlinear eigenvalue problems is given in Algorithm 12.4. This algorithm is by Voss [14]. There are two noteworthy issues.

Algorithm 12.4 Nonlinear Jacobi–Davidson algorithm

```
1:  Start with an initial basis $V$, $V^*V = I$; $m = 1$.
2:  Determine a preconditioner $K \approx T(\sigma)$, $\sigma$ close to the first wanted eigenvalue.
3:  while $m \le$ number of wanted eigenvalues do
4:     Compute an approximation to the $m$-th wanted eigenvalue $\lambda_m$ and corresponding
       eigenvector $\mathbf{s}_m$ of the projected problem $V^*T(\lambda_m)V\mathbf{s} = \mathbf{0}$.
5:     Determine the Ritz vector $\mathbf{u} = V\mathbf{s}_m$ and the residual $\mathbf{r} = T(\lambda_m)\mathbf{u}$
6:     if $\|\mathbf{r}\|/\|\mathbf{u}\| < \varepsilon$ then
7:        Accept approximate eigenpair $(\lambda_m, \mathbf{u})$; $m := m + 1$;
8:        Reduce the search space $V$ if necessary
9:        Choose an approximation $(\lambda_m, \mathbf{u})$ to the next eigenpair.
10:       Compute the residual $\mathbf{r} = T(\lambda_m)\mathbf{u}$
11:    end if
12:    $\mathbf{p} = T'(\lambda_m)\mathbf{x}$;
13:    (Approximatively) solve the correction equation for $\mathbf{t}$,

       (12.50)   $(I - \frac{\mathbf{p}\mathbf{u}^*}{\mathbf{u}^*\mathbf{p}})T(\sigma)(I - \frac{\mathbf{u}\mathbf{u}^*}{\mathbf{u}^*\mathbf{u}})\mathbf{t} = -\mathbf{r}, \qquad \mathbf{t} \perp \mathbf{u}.$

14:    Orthogonalize $\mathbf{t} := \mathbf{t} - VV^*\mathbf{t}$, $\mathbf{v} := \mathbf{t}/\|\mathbf{t}\|$, and expand the subspace $[V, \mathbf{v}]$.
15:    Determine a new preconditioner $K \approx T(\lambda_m)$ if necessary.
16:    Update the projected problem.
17: end while
```

1. The projected problem is the *nonlinear* eigenvalue problem $V^*T(\lambda_m)V\mathbf{s} = \mathbf{0}$ where λ_m is an approximation to the wanted eigenvalue.

2. In the expansion of the search space, it is ensured that the Newton iterate is contained in the expanded search space. To this end, assume that $\mathbf{u}$ is the Ritz vector in $\mathcal{R}(V)$ obtained from the projected problem, $\mathbf{u} = V\mathbf{s}$. Set $\mathbf{p} = T'(\lambda_m)\mathbf{u}$. We now solve the correction equation

   $$(I - \frac{\mathbf{p}\mathbf{u}^*}{\mathbf{u}^*\mathbf{p}})T(\lambda_m)(I - \frac{\mathbf{u}\mathbf{u}^*}{\mathbf{u}^*\mathbf{u}})\mathbf{t} = -\mathbf{r} = -T(\lambda_m)\mathbf{u}, \qquad \mathbf{t} \perp \mathbf{u}. \tag{12.50}$$

   This equation can be written as

   $$T(\lambda_m)\mathbf{t} - \alpha\mathbf{p} = -\mathbf{r}, \qquad \alpha = \frac{1}{\mathbf{u}^*\mathbf{p}}\mathbf{u}^*T(\lambda_m)\mathbf{t}.$$

   Using $T(\lambda_m)\mathbf{u} = \mathbf{r}$ we get

   $$\mathbf{t} = -\mathbf{u} + \alpha T(\lambda_m)^{-1}\mathbf{p} = -\mathbf{u} + \alpha T(\lambda_m)^{-1}T'(\lambda_m)\mathbf{u}.$$

   α is determined such that $\mathbf{t} \perp \mathbf{u}$. Since $\mathbf{u} \in \mathcal{R}(V)$, we must have $T(\lambda_m)^{-1}T'(\lambda_m)\mathbf{u} \in \mathcal{R}([V, \mathbf{t}])$. This ensures the quadratic convergence rate of Newton's method.

The correction equation (12.50) in Algorithm 12.4 is typically solved to low accuracy by a preconditioned GMRES iteration where the preconditioner has the form

$$\tilde{K} = (I - \frac{\mathbf{p}\mathbf{u}^*}{\mathbf{u}^*\mathbf{p}})K(I - \frac{\mathbf{u}\mathbf{u}^*}{\mathbf{u}^*\mathbf{u}}), \qquad K \approx T(\sigma). \tag{12.51}$$

Solving with the preconditioner amounts to solving the equation

$$\tilde{K}\mathbf{t} = \mathbf{g}, \qquad \mathbf{t} \perp \mathbf{u}.$$

Bibliography

[1] R. Barret, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, PA, 1994. (Available from Netlib at URL `http://www.netlib.org/templates/index.html`).

[2] E. R. Davidson, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices*, J. Comp. Phys., 17 (1975), pp. 87–94.

[3] D. R. Fokkema, G. L. G. Sleijpen, and H. A. van der Vorst, *Jacobi–Davidson style QR and QZ algorithms for the partial reduction of matrix pencils*, SIAM J. Sci. Comput., 20 (1998), pp. 94–125.

[4] R. Geus, *The Jacobi–Davidson algorithm for solving large sparse symmetric eigenvalue problems*, PhD Thesis No. 14734, ETH Zürich, 2002. (Available at URL `http://e-collection.ethbib.ethz.ch/show?type=diss&nr=14734`).

[5] G. H. Golub and C. F. van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 2nd ed., 1989.

[6] C. G. J. Jacobi, *Über ein leichtes Verfahren die in der Theorie der Säcularstörungen vorkommenden Gleichungen numerisch aufzulösen*, J. reine angew. Math., 30 (1846), pp. 51–95.

[7] R. B. Morgan, *Davidson's method and preconditioning for generalized eigenvalue problems*, J. Comp. Phys., 89 (1990), pp. 241–245.

[8] R. B. Morgan and D. S. Scott, *Generalizations of Davidson's method for computing eigenvalues of sparse symmetric matrices*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 817–825.

[9] A. Ruhe, *Algorithms for the nonlinear eigenvalue problem*, SIAM J. Numer. Anal., 10 (1973), pp. 674–689.

[10] G. L. G. Sleijpen, A. G. L. Booten, D. R. Fokkema, and H. A. van der Vorst, *Jacobi–Davidson type methods for generalized eigenproblems and polynomial eigenproblems*, BIT, 36 (1996), pp. 595–633.

[11] G. L. G. Sleijpen and H. A. van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.

[12] G. L. G. Sleijpen and H. A. van der Vorst, *Jacobi–Davidson 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. 238–246.

[13] G. W. Stewart, *Matrix Algorithms II: Eigensystems*, SIAM, Philadelphia, PA, 2001.

[14] H. Voss, *A Jacobi–Davidson method for nonlinear eigenproblems*, in Computational Science – ICCS 2004, G. D. van Albada, M. Bubak, P. M. A. Sloot, and J. J. Dongarra, eds., Berlin, 2004, Springer, pp. 34–41. (Lecture Notes in Computer Science, 3037).