RAL-TR-1999-011 (Revised December 1999)

Locking and restarting quadratic eigenvalue solvers

Karl Meerbergen[1]

ABSTRACT

This paper studies the solution of quadratic eigenvalue problems by the quadratic residual iteration method. The focus is on applications arising from finite-element simulations in acoustics. One approach is the shift-invert Arnoldi method applied to the linearized problem. When more than one eigenvalue is wanted, it is advisable to use locking or deflation of converged eigenvectors (or Schur vectors). In order to avoid unlimited growth of the subspace dimension, one can restart the method by purging unwanted eigenvectors (or Schur vectors). Both locking and restarting use the partial Schur form. The disadvantage of this approach is that the dimension of the linearized problem is twice that of the quadratic problem. The quadratic residual iteration and Jacobi-Davidson methods directly solve the quadratic problem. Unfortunately, the Schur form is not defined, nor are locking and restarting. This paper shows a link between methods for solving quadratic eigenvalue problems and the linearized problem. It aims to combine the benefits of the quadratic and the linearized approaches by employing a locking and restarting scheme based on the Schur form of the linearized problem in quadratic residual iteration and Jacobi-Davidson. Numerical experiments illustrate quadratic residual iteration and Jacobi-Davidson for computing the linear Schur form. It also makes a comparison with the shift-invert Arnoldi method.

Keywords: Quadratic eigenvalue problem, linearization, Schur factorization, Davidson, shift-invert Arnoldi, deflation, purging

AMS (MOS) subject classifications: 65F15,65F50

Computational Science and Engineering Department
Atlas Centre
Rutherford Appleton Laboratory
Oxon OX11 0QX

December 3, 1999

Contents

1 Introduction

This paper studies the solution of the quadratic eigenvalue problem

$$Ku + i\omega Cu - \omega^2 Mu = 0 \qquad u \neq 0 \tag{1.1}$$

where K, C and M have dimension $n \times n$ and M is symmetric positive definite. The scalar ω is called an eigenvalue, u is a corresponding eigenvector, and (ω, u) is an eigenpair. This problem arises from the finite-element simulation of damped acoustic problems, where K is the stiffness matrix and is symmetric positive (semi) definite, M is the mass matrix and is symmetric positive definite, and C is the damping matrix and is symmetric and sometimes complex. The condition number of M is usually relatively small, since M is the discretization of the continuous identity operator. Typically, K, C, and M are large (of the order of 10,000 or more unknowns) and sparse. In engineering applications, the eigenvalue ω is complex. The real part is the resonance frequency, while the imaginary part represents the exponential damping of the eigenmode. In applications, all the eigenvalues in a frequency range are wanted, this is a few tens to a few hundreds of eigenpairs.

The problem (1.1) can be 'linearized' into

$$\begin{bmatrix} K & 0 \\ 0 & M \end{bmatrix} \begin{pmatrix} u \\ \omega u \end{pmatrix} = \omega \begin{bmatrix} -iC & M \\ M & 0 \end{bmatrix} \begin{pmatrix} u \\ \omega u \end{pmatrix} \tag{1.2}$$

which we also denote as $Ax = \omega Bx$ with

$$A = \begin{bmatrix} K & 0 \\ 0 & M \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} -iC & M \\ M & 0 \end{bmatrix} . \tag{1.3}$$

(See (Saad 1992, Chapter X) for alternatives.) Note that for acoustic finite-element applications, A is symmetric positive (semi) definite and B is (in general) complex symmetric. Since M is nonsingular, this problem has $2n$ finite eigenvalues. When C is purely imaginary, $Ax = \omega Bx$ is a Hermitian problem, so all eigenvalues are real. In general, C has a real component hence complex eigenvalues are present. When C is real, then the spectrum is symmetric with respect to the imaginary axis : indeed, if (ω, u) satisfies (1.1) and C is real then $(-\bar{\omega}, \bar{u})$ is also an eigenpair. Note that when $C = 0$, the spectral transformation block Lanczos method (Grimes, Lewis and Simon 1994) is a very robust and efficient solver.

In the literature, methods have been proposed for solving (1.1) and (1.2). The linearized problem can be solved by the shift-invert Arnoldi method (Saad 1992, Natarajan 1992). This method computes the eigenpairs of the shift-invert transformation $(A - \sigma B)^{-1}B$ where σ is called the shift. Alternatively, the rational Krylov method (Ruhe 1998) or the Jacobi-Davidson method (Sleijpen and van der Vorst 1996) may be used. The advantage of the linearized approach is that existing methods and software can be used. A disadvantage is that the dimension is doubled.

Methods have been developed for directly tackling (1.1). They solve a sequence of linear systems

$$(K + i\sigma C - \sigma^2 M)y = r \ , \tag{1.4}$$

where σ may change at each iteration. When a direct method is used for solving (1.4), a matrix factorization on each iteration is inevitable. Neumaier (1998) and Huitfeldt and Ruhe (1990) propose methods that use a fixed σ. This allows the same factorization to be used for several iterations. Another approach is the Jacobi-Davidson method (Sleijpen, Booten, Fokkema and van der Vorst 1996*a*, Sleijpen, van der Vorst and van Gijzen 1996*b*, van Gijzen and Raeven 1995) for the quadratic problem, which should not be confused with the Jacobi-Davidson method for the linearized problem. The methods that we study build a subspace. For reasons of computational cost and memory, the subspace dimension is limited. When this limit has been reached without convergence of the sought after eigenvalues, the method needs to be restarted. The concept of restarting eigenvalue solvers is very well understood for linear problems. See the recent work for the Arnoldi method (Sorensen 1992, Lehoucq and Sorensen 1996, Morgan 1996), the Jacobi-Davidson method (Fokkema, Sleijpen and van der Vorst 1999) and the rational Krylov method (Ruhe 1998, De Samblanx, Meerbergen and Bultheel 1997). When more than one eigenvalue is wanted, it is usually advisable to lock the converged eigenpairs. This is proposed for subspace iteration (Stewart 1976), Arnoldi's method (Lehoucq and Sorensen 1996), Jacobi-Davidson (Fokkema et al. 1999) and rational Krylov (Ruhe 1998). Both restarting and locking use the partial Schur form.

The purpose of this paper is the development of reliable deflation and restarting in methods that solve the quadratic problem (1.1). The problem is that the Schur form is not defined for quadratic problems. We give a definition and show that this Schur form does not always exist. Therefore, we propose using the Schur form of the linearized problem (1.2).

We also want to stress that all theory in this paper can be extended to the m degree polynomial case with $m > 2$. The polynomial problem

$$A_0 u + \lambda A_1 u + \cdots + \lambda^m A_m u = 0$$

can be solved by linearization into

$$\begin{bmatrix} A_0 & & & \\ & I & & \\ & & \ddots & \\ & & & I \end{bmatrix} \begin{pmatrix} u \\ \lambda u \\ \lambda^2 u \\ \vdots \\ \lambda^{m-1} u \end{pmatrix} = \lambda \begin{bmatrix} -A_1 & -A_2 & \cdots & -A_m \\ I & & & \\ & \ddots & & \\ & & I & \end{bmatrix} \begin{pmatrix} u \\ \lambda u \\ \lambda^2 u \\ \vdots \\ \lambda^{m-1} u \end{pmatrix} .$$

The paper is organized as follows. In §2, we present the quadratic residual iteration method for solving (1.1) and prove a relationship with a modification of the generalized Davidson method for the solution of the linearized problem (1.2). In §3, we use the theory developed in §2 to link the Jacobi-Davidson methods for (1.1) and (1.2). In §4, the notion of partial Schur form is extended to quadratic eigenvalue problems, and the theory of §2 is used to efficiently compute an approximate partial Schur form of the linearized problem by orthogonal projection of the quadratic problem. In §5 a deflation or locking technique is proposed for the linearized problem, and in §6, we discuss restarting by purging of Schur vectors of the linearized problem. In §7 we explain which vectors we use to expand the subspace when we want to compute a partial Schur form of the linearized problem. Section 8 presents a practical algorithm that is illustrated by numerical examples including one from applications. In §9, we compare quadratic residual iteration and Jacobi-Davidson for computing a partial Schur form. In §10, we compare the shift-invert Arnoldi method with quadratic residual iteration and Jacobi-Davidson. Finally, we summarize the main conclusions in §11. Throughout the paper, $\|\cdot\|$ is used for the 2-norm of matrices and vectors and $\|\cdot\|_F$ for the Frobenius norm.

2 Quadratic residual iteration

In this section, we derive a relationship between methods for solving (1.1) and (1.2). All conclusions assume exact arithmetic. For results on the backward error and condition of the linearized problem we refer to Tisseur (1998). For the linearized problem, we consider the generalized Davidson method (Morgan 1992) (which formally covers the Jacobi-Davidson method (Sleijpen and van der Vorst 1996, Morgan and Meerbergen 1999)) and for the quadratic problem we consider the residual iteration method (Neumaier 1998) with subspace projection. We also discuss the Jacobi-Davidson variant for this problem. This section is devoted to the development of a relationship between the two approaches. Therefore, we also define a modified Davidson technique for the linearized problem (1.2) which is shown to produce the same results as the quadratic residual iteration on (1.1).

The generalized Davidson method for the linearized problem is described by the following algorithm (Morgan 1992).

Algorithm 2.1 (generalized Davidson method)
1. Given $v_1 \in \mathbf{C}^{2n}$ with $\|v_1\| = 1$.
2. For $k = 1, 2, \ldots$ do
2.1. Let $\mathcal{V}_k = [v_1, \ldots, v_k]$.
2.2. Compute the projection matrices $A_k = \mathcal{V}_k^* A \mathcal{V}_k$, and $B_k = \mathcal{V}_k^* B \mathcal{V}_k$.
2.3. Compute the eigenpair (ω_k, z_k) of interest of $A_k z = \omega B_k z$.
2.4. Compute the corresponding Ritz vector $x_k = \mathcal{V}_k z_k$.
2.5. Compute the corresponding residual $r_k = A x_k - \omega_k B x_k$.
2.6. Solve the linear system $(A - \sigma_k B) y_k = r_k$.
2.7. Orthonormalize y_k against $v_1, \ldots, v_k$ into v_{k+1}.

This algorithm consists of a sequence of Cayley transformations

$$y_k = (A - \sigma_k B)^{-1} (A - \omega_k B) x_k \ , \tag{2.1}$$

and a projection step for computing the approximate eigenpair. The Cayley transform aims to improve the approximation of eigenvalues near σ_k. For projection methods, approximate eigenpairs are called Ritz pairs. The approximate eigenvalue is a Ritz value and the approximate eigenvector a Ritz vector. The small eigenvalue problem in Step 2.3 is usually solved by the QZ method (Golub and Van Loan 1996). We select the eigenpair of interest, e.g. corresponding to the eigenvalue nearest σ_k. When the linear system in Step 2.6 is solved by an iterative method, Algorithm 2.1 is the generalized Davidson method. In Davidson's method, one usually employs $\sigma_k = \omega_k$ and Step 2.6 is executed approximately (Crouzeix, Philippe and Sadkane 1994) or one can use the Jacobi-Davidson method (see §3). This choice of σ_k leads to quadratic convergence. The generalized Davidson method does not exploit the special structure of eigenvectors of (1.2). Clearly, it is sufficient to compute the first n components of the eigenvectors and then construct the remaining components. The following algorithm is a modification of the generalized Davidson method that uses the first n components only.

Algorithm 2.2 (modified Davidson method)

1. Given $v_1 \in \mathbf{C}^n$ with $\|v_1\| = 1$.

2. For $k = 1, 2, \ldots$ do

- *2.1. Let $V_k = [v_1, \ldots, v_k]$ and $\mathcal{V}_{2k} = \begin{pmatrix} V_k & 0 \\ 0 & V_k \end{pmatrix}$.*
- *2.2. Compute the projection matrices $A_{2k} = \mathcal{V}_{2k}^* A \mathcal{V}_{2k}$, and $B_{2k} = \mathcal{V}_{2k}^* B \mathcal{V}_{2k}$.*
- *2.3. Compute the eigenpair (ω_k, z_k) of interest of $A_{2k} z = \omega B_{2k} z$.*
- *2.4. Compute the corresponding Ritz vector $x_k = \mathcal{V}_{2k} z_k$.*
- *2.5. Compute the corresponding residual $r_k = A x_k - \omega_k B x_k$.*
- *2.6. Solve the linear system $(A - \sigma_k B) y_k = r_k$.*
- *2.7. Orthonormalize the first n components of y_k against $v_1, \ldots, v_k$ into v_{k+1}.*

Alternatively, one could select the last n components of y_k in Step 2.7. As we shall see, there is no advantage in exact arithmetic. The basis vectors satisfy the projection equation

$$A \mathcal{V}_{2k} - B \mathcal{V}_{2k} H_{2k} = \mathcal{F}_{2k}$$

with projection matrix $H_{2k} = B_{2k}^{-1} A_{2k}$ and with residual term $\mathcal{F}_{2k}$ satisfying $\mathcal{V}_{2k}^* \mathcal{F}_{2k} = 0$. The projection equation is frequently used in §4.

The quadratic residual iteration is now discussed.

Algorithm 2.3 (quadratic residual iteration)

1. Given $v_1 \in \mathbf{C}^n$ with $\|v_1\| = 1$.

2. For $k = 1, 2, \ldots$ do

- *2.1. Let $V_k = [v_1, \ldots, v_k]$.*
- *2.2. Compute the projection matrices $K_k = V_k^* K V_k$, $C_k = V_k^* C V_k$, and $M_k = V_k^* M V_k$.*
- *2.3. Compute the eigenpair (ω_k, z_k) of interest of $K_k z + i\omega C_k z - \omega^2 M_k z = 0$.*
- *2.4. Compute the corresponding Ritz vector $u_k = V_k z_k$.*
- *2.5. Compute the corresponding residual $r_k = K u_k + \omega_k C u_k - \omega_k^2 M u_k$.*
- *2.6. Solve the linear system $(K + i\sigma_k C - \sigma_k^2 M) y_k = r_k$.*
- *2.7. Orthonormalize y_k against $v_1, \ldots, v_k$ into v_{k+1}.*

The algorithm is very similar to Algorithms 2.1 and 2.2, but instead of a regular Cayley transform, a quadratic Cayley transform

$$y_k = (K + i\sigma_k C - \sigma_k^2 M)^{-1} (K + i\omega_k C - \omega_k^2 M) u_k \tag{2.2}$$

is employed. The small eigenvalue problem at Step 2.3 is solved by a method for solving quadratic eigenvalue problems, e.g. Newton's method or by solving the corresponding linearized problem.

In the following, we derive a relationship between Algorithms 2.2 and 2.3. The main operations in these algorithms are the Cayley transform and the projection step.

Cayley transformation On each iteration, the subspace is expanded by the Cayley transformation applied to a Ritz vector. Here we establish a relationship between the Cayley transforms (2.1) and (2.2) applied to a vector with a special structure.

Lemma 2.1 *Let A and B be defined by (1.3) and*

$$\begin{pmatrix} x \\ y \end{pmatrix} = (A - \sigma B)^{-1}(A - \omega B) \begin{pmatrix} u \\ \omega u \end{pmatrix} . \tag{2.3}$$

Then

$$x = (K + i\sigma C - \sigma^2 M)^{-1}(K + i\omega C - \omega^2 M)u \qquad \text{and} \qquad y = \sigma x . \tag{2.4}$$

Proof Write (2.3) as

$$\begin{aligned} \begin{bmatrix} K + i\sigma C & -\sigma M \\ -\sigma M & M \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} &= \begin{bmatrix} K + i\omega C & -\omega M \\ -\omega M & M \end{bmatrix} \begin{pmatrix} u \\ \omega u \end{pmatrix} \\ &= \begin{pmatrix} (K + i\omega C - \omega^2 M)u \\ 0 \end{pmatrix} . \end{aligned}$$

The last row readily produces $y = \sigma x$. Substituting this in the first row leads to the first statement in (2.4). This proves the lemma. □

Projection The projection used in Algorithm 2.2 produces the same Ritz pairs as the projection in Algorithm 2.3 if the V_k's are the same.

Lemma 2.2 *If the projection of (1.1) on* Range(V) *produces the Ritz pair (ω, u) then the projection of (1.2) on*

$$\mathcal{V} = \text{Range}\left(\begin{pmatrix} V & 0 \\ 0 & V \end{pmatrix}\right)$$

produces the Ritz pair

$$\left(\omega, \begin{pmatrix} u \\ \omega u \end{pmatrix}\right) . \tag{2.5}$$

Proof The projection of (1.2) on $\mathcal{V}$ is

$$\begin{aligned} &\begin{pmatrix} V & 0 \\ 0 & V \end{pmatrix}^* \begin{bmatrix} K & 0 \\ 0 & M \end{bmatrix} \begin{pmatrix} V & 0 \\ 0 & V \end{pmatrix} \begin{pmatrix} z \\ t \end{pmatrix} \\ &= \omega \begin{pmatrix} V & 0 \\ 0 & V \end{pmatrix}^* \begin{bmatrix} -iC & M \\ M & 0 \end{bmatrix} \begin{pmatrix} V & 0 \\ 0 & V \end{pmatrix} \begin{pmatrix} z \\ t \end{pmatrix} . \end{aligned}$$

This leads to $V^*KVz = \omega(-iV^*CVz + V^*MVt)$ and $t = \omega z$. This implies that $V^*(K + i\omega C - \omega^2 M)Vz = 0$, which is the projection of (1.1) on Range(V). This completes the proof. □

The theory can now be used for establishing the following relationship.

Theorem 2.3 *Suppose that v_1 is the same for Algorithms 2.2 and 2.3. When $k \leq n$, both algorithms produce the same Ritz pairs.*

Proof The proof is given by induction. Suppose that we use an initial vector v_1 for Algorithms 2.2 and 2.3. Suppose that the theorem is true at the end of iteration $k-1$, i.e. V_k is the same for both algorithms. Following Lemma 2.2, Algorithm 2.3 produces (ω, u) and Algorithm 2.2 $\left(\omega \quad \begin{pmatrix} u \\ \omega u \end{pmatrix} \right)$. Finally, following Lemma 2.1, x_k is the same for both algorithms, so both produce the same v_{k+1}. This implies that V_{k+1} is equal in both algorithms, from which the proof follows. □

An important question is how much better is the subspace generated by Algorithm 2.2 than Algorithm 2.1. Let $\begin{pmatrix} u \\ \omega u \end{pmatrix}$ be a Ritz vector and let $\begin{pmatrix} y \\ \sigma y \end{pmatrix}$ be the result of the Cayley transformation. In Algorithm 2.2, we add both $\begin{pmatrix} y \\ 0 \end{pmatrix}$ and $\begin{pmatrix} 0 \\ y \end{pmatrix}$ to the subspace. When $\sigma \neq 0$, this is mathematically equivalent to adding $\begin{pmatrix} y \\ \sigma y \end{pmatrix}$ and $\begin{pmatrix} y \\ -\sigma y \end{pmatrix}$, since both vector pairs have the same span. The first vector is the result of the Cayley transform

$$\begin{pmatrix} y \\ \sigma y \end{pmatrix} = (A - \sigma B)^{-1}(A - \omega B) \begin{pmatrix} u \\ \omega u \end{pmatrix} .$$

The first vector tries to improve the eigenvector components corresponding to the eigenvalues near σ. When $C = 0$, then the second vector is

$$\begin{pmatrix} y \\ -\sigma y \end{pmatrix} = (A + \sigma B)^{-1}(A + \omega B) \begin{pmatrix} u \\ -\omega u \end{pmatrix}$$

i.e. tries to improve the eigenvectors corresponding to eigenvalues nearest $-\sigma$. In general, however, $\begin{pmatrix} y \\ -\sigma y \end{pmatrix}$ does not have a particular meaning. When C is small, this vector may help finding eigenvalues near $-\sigma$.

Stopping criterion Usually, iterative eigenvalue solvers use a stopping criterion based on the residual. If $Ax - \lambda Bx = r$, then (λ, x) is an exact eigenpair of the perturbed problem $(A + E)x = \lambda Bx$ with $Ex = -r$. So, $\|r\|/\|x\|$ is a measure of the backward error. A relationship between the residual and the forward error on the eigenvalues is given by the Bauer-Fike theorem (Saad 1992, Theorem 3.6).

Theorem 2.4 (Bauer-Fike) *Consider an $n \times n$ matrix G that has n independent eigenvectors. Let λ be an eigenvalue of G and let (μ, x) be an approximate eigenpair with residual $r = Gx - \mu x$ and $\|x\| \neq 0$. Then*

$$|\lambda - \mu| \leq \kappa \|r\|/\|x\|$$

where κ is the condition number of the matrix of eigenvectors.

The residual for the linearized problem with Ritz pair (2.5) becomes

$$r_L = A \begin{pmatrix} u \\ \omega u \end{pmatrix} - \omega B \begin{pmatrix} u \\ \omega u \end{pmatrix} = \begin{pmatrix} r_P \\ 0 \end{pmatrix}$$

with $r_P = Ku + i\omega Cu - \omega^2 Mu$. The first component of r_L is the quadratic residual, so the two-norm of both residuals is equal. So, for quadratic problems, the residual can also be regarded as a backward error. Note, however, that the Ritz vectors, u and $\begin{pmatrix} u \\ \omega u \end{pmatrix}$, have a different norm. In order to use the Bauer-Fike theorem, consider the residual $B^{-1}Ax - \omega x$, which becomes

$$\begin{aligned} B^{-1}A\begin{pmatrix} u \\ \omega u \end{pmatrix} - \omega\begin{pmatrix} u \\ \omega u \end{pmatrix} &= \begin{bmatrix} -iC & M \\ M & 0 \end{bmatrix}^{-1}\begin{pmatrix} Ku + i\omega Cu - \omega^2 Mu \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ M^{-1}Ku + i\omega M^{-1}Cu - \omega^2 u \end{pmatrix} \end{aligned}$$

So, the error between the exact eigenvalue λ and an approximation ω can be bounded by

$$|\omega - \lambda| \leq \kappa \|M^{-1}\| \frac{\|Ku + i\omega Cu - \omega^2 Mu\|}{\|u\|\sqrt{1 + |\omega|^2}} ,$$

where κ is the condition number of the matrix of eigenvectors of $B^{-1}A$, and the denominator is the norm of the Ritz vector.

Accuracy of the Cayley transform A comment is in order on the solution of the linear system

$$(K + i\sigma C - \sigma^2 M)y = (K + i\omega C - \omega^2 M)u \equiv r_P .$$

When we use a linear system solver, we have a residual s so that

$$(K + i\sigma C - \sigma^2 M)y = (K + i\omega C - \omega^2 M)x - s .$$

When a direct method is used, $\|s\|$ is usually of the order of machine precision times $\|r_P\|$ and the quadratic Cayley transform can be considered as exact. When an iterative solver is used, $\|s_k\| \leq \tau \|r_P\|$ with τ the relative residual tolerance. The smaller τ, the more expensive the iterative solver. We therefore want to choose the tolerance not too small. There is a relationship with the linear problem, since

$$\begin{bmatrix} K + i\sigma C & -\sigma M \\ -\sigma M & M \end{bmatrix}\begin{pmatrix} y \\ \sigma y \end{pmatrix} = \begin{bmatrix} K + i\omega C & -\omega M \\ -\omega M & M \end{bmatrix}\begin{pmatrix} u \\ \omega u \end{pmatrix} - \begin{pmatrix} s \\ 0 \end{pmatrix} .$$

When iterative solvers are used for computing the linear Cayley transform in eigenvalue solvers, we talk about the inexact Cayley transform (Meerbergen and Roose 1997, Meerbergen and Roose 1996, Lehoucq and Meerbergen 1998, Morgan and Meerbergen 1999). Two elements play a role in the convergence of an inexact Cayley transform iteration method. The first one is the convergence for an exact Cayley transformation, and the second one is the accuracy of the linear system solver, τ. When τ is not too large, the convergence is as fast as for $\tau = 0$. We give an example in §8.3.

Cayley transform and Shift-and-invert A problem arises when $\sigma = \omega$ in the Cayley transformation. Since $y = (A - \omega B)^{-1}(A - \omega B)x = x$, no new direction is added to the subspace. When the Cayley transform is computed exactly, e.g. with a direct linear solver (in practice, we have a small backward error on the solution of the linear system), then

$$y \equiv (A - \sigma B)^{-1}(A - \omega B)x = x + (\sigma - \omega)(A - \sigma B)^{-1}Bx \tag{2.6}$$

The matrix $(A - \sigma B)^{-1}B$ is called the shift-invert transformation (Saad 1992) or the spectral transformation (Ericsson and Ruhe 1980). The use of the Cayley transform or the spectral transformation in a projection method is equivalent since both add the same direction orthogonal to x to the subspace. The advantage of the shift-invert transform is that the transformation still works when $\sigma = \omega$. When an iterative solver is used it is usually advantageous to use the Cayley transform instead of shift-invert (Lehoucq and Meerbergen 1998, Morgan and Meerbergen 1999).

For the quadratic eigenvalue problem, the shift-invert transformation is defined as follows. The Cayley transform can be rewritten as

$$\begin{aligned} y &\equiv (K + i\sigma C - \sigma^2 M)^{-1}(K + i\omega C - \omega^2 M)u \\ &= u + (\omega - \sigma)(K + i\sigma C - \sigma^2 M)^{-1}(iC - (\omega + \sigma)M)u \end{aligned}$$

The shift-invert transformation is defined as

$$(K + i\sigma C - \sigma^2 M)^{-1}(iC - (\omega + \sigma)M) \tag{2.7}$$

and can also be used when $\sigma = \omega$.

3 The Jacobi-Davidson method

The Jacobi-Davidson method is an alternative to the shift-invert transformation when an iterative linear solver is used and $\sigma = \omega$.

The combination of the Newton method applied to the set of nonlinear equations

$$\begin{aligned} Ax - \omega Bx &= 0 \\ x^* x &= 1 \end{aligned}$$

in ω and x and the generalized Davidson method (Algorithm 2.1) is called the Jacobi-Davidson method (Sleijpen and van der Vorst 1996, Sleijpen et al. 1996*a*). It adds the x component of the solution of the Newton iteration to a subspace and computes Ritz pairs by projection. The algorithm for the linear problem (1.2) is the same as Algorithm 2.1 except for the transformation in Step 2.6. Instead, a 'correction equation' is solved. Let (ω, x) be a Ritz pair, then the subspace is expanded with y satisfying

$$\left(I - \frac{Bxx^*}{x^*Bx}\right)(A - \omega B)\left(I - \frac{xx^*}{x^*x}\right) y = (A - \omega B)x \ . \tag{3.1}$$

The solution y is computed by an iterative method, with a suitable preconditioner if available. For the quadratic problem (1.1), the Jacobi-Davidson method is Algorithm 2.3

where Step 2.6 is replaced by

$$\left(I - \frac{(-iC + 2\omega M)uu^*}{u^*(-iC + 2\omega M)u}\right)(K + i\omega C - \omega^2 M)\left(I - \frac{uu^*}{u^*u}\right)v$$
$$= (K + i\omega C - \omega^2 M)u \ , \tag{3.2}$$

where (ω, u) is a Ritz pair (Sleijpen et al. 1996*a*, van Gijzen and Raeven 1995). This can be derived from one Newton iteration on the equations (1.1) and $u^*u = 1$.

The Jacobi-Davidson method is strongly related to the Davidson method. When we require that $y \perp x$, we can rewrite (3.1) as

$$\begin{aligned} (A - \omega B)y - \epsilon Bx &= (A - \omega B)x \\ \text{or} \quad y &= x + \epsilon(A - \omega B)^{-1}Bx \ , \end{aligned} \tag{3.3}$$

where ϵ is so that $y \perp x$ (Sleijpen and van der Vorst 1996). Note the resemblance with (2.6). When y is added to the subspace, the new direction v_{k+1} is independent of ϵ, since x is in the subspace. We can write y also as

$$y = (A - \omega B)^{-1}(A - (\omega - \epsilon)B)x \ .$$

We immediately see the connection with the generalized Davidson method where the pole (σ) is now ω and the zero (ω) is now $\omega + \epsilon$. It is shown in (Morgan and Meerbergen 1999) that $\omega + \epsilon$ is the harmonic Ritz value with target ω. The value ϵ converges to zero when ω converges to an eigenvalue (Sleijpen and van der Vorst 1996).

Similarly, with $v \perp u$, (3.2) can be rewritten as

$$\begin{aligned} (K + i\omega C - \omega^2 M)v - \eta(-iC + 2\omega M)u &= (K + i\omega C - \omega^2 M)u \\ v &= u + \eta(K + i\omega C - \omega^2 M)^{-1}(-iC + 2\omega M)u \ , \end{aligned} \tag{3.4}$$

where η is so that $v \perp u$. Note the resemblence with (2.7) for $\sigma = \omega$.

In §2, we showed the equivalence of the modified Davidson method in Algorithm 2.2 and quadratic residual iteration in Algorithm 2.3. The following lemma shows that, when Step 2.6 is replaced by the solution of the Newton correction equations (3.1) and (3.2) respectively, the subspace V is expanded in the same way for both Algorithms 2.2 and 2.3.

Lemma 3.1 *Let (ω, u) be a Ritz pair for (1.1) and assume $\epsilon \neq 0$. Let $x = \begin{pmatrix} u \\ \omega u \end{pmatrix}$ and let $y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$ be the solution of (3.1). Then $(\eta/\epsilon)y_1$ is a solution of (3.2).*

Proof Assume that $y \perp x$ and rewrite (3.3) as

$$\begin{bmatrix} K + i\omega C & -\omega M \\ -\omega M & M \end{bmatrix}\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} - \epsilon\begin{bmatrix} -iC & M \\ M & 0 \end{bmatrix}\begin{pmatrix} u \\ \omega u \end{pmatrix}$$
$$= \begin{bmatrix} K + i\omega C & -\omega M \\ -\omega M & M \end{bmatrix}\begin{pmatrix} u \\ \omega u \end{pmatrix}$$

or

$$\begin{aligned}(K+i\omega C)y_1-\omega My_2-\epsilon(-iC+\omega M)u &= Ku+i\omega Cu-\omega^2 Mu\\ -\omega My_1+My_2-\epsilon Mu &= 0\end{aligned}$$

from which $y_2=\omega y_1+\epsilon u$. This gives

$$(K+i\omega C-\omega^2 M)y_1-\epsilon(-iC+2\omega M)u=Ku+i\omega Cu-\omega^2 Mu\ ,$$

which looks like (3.4). The only difference is that $\eta\neq\epsilon$. Since $(\eta/\epsilon)y_1$ and v in (3.4) have the same component orthogonal to u, $(\eta/\epsilon)y_1$ satisfies (3.2). This gives the proof of the lemma. □

Finally it is important to note that the projections used in (3.1) and (3.2) can also be used for $\sigma\neq\omega$. An example is given in (Meerbergen 1996, §3.3.6). However, when the linear systems are solved exactly (e.g. using a direct linear solver), there is no advantage in doing this. The search space V_k will be expanded with the same direction. This is discussed in more detail in (Lehoucq and Meerbergen 1998, Eq. (6.3) and (6.4)) and (Morgan and Meerbergen 1999). When an iterative solver is used, the projections may help speed up the convergence of the iterative solvers, since the projection may remove a small eigenvalue of the matrix in the linear system. An example is given in §9.

4 The Schur factorization

Methods for solving linear eigenvalue problems use the Schur factorization when more than one eigenvalue needs to be computed. The reasons are that this factorization always exists in contrast with the eigendecomposition, Schur bases are orthogonal, and can easily be used for locking and restarting purposes as we will discuss in the coming sections. This section is devoted to the Schur factorization for the linear problem and the quadratic problem. We show some properties of the Schur factorization of the linear problem and define one for the quadratic problem. We also show that the quadratic Schur form may not exist. Therefore, we suggest the use of the Schur form of the linearized problem.

4.1 Linear problems

When B is invertible, the Schur form or Schur factorization of $Ax=\lambda Bx$ is defined by

$$B^{-1}AX=XS \qquad \Leftrightarrow \qquad AX=BXS \tag{4.1}$$

where X is an $n\times n$ unitary matrix and S an $n\times n$ upper triangular matrix with the eigenvalues on the main diagonal. The columns of X are the Schur vectors and S is the Schur matrix. The Schur form (4.1) can be computed by the QR method applied to $B^{-1}A$ or the QZ method applied to the pair (A,B) (Golub and Van Loan 1996). Using the QR method assumes a nonsingular B and the formation of $B^{-1}A$. One could avoid this computation by using the generalized Schur form computed by the QZ method. This complicates the notation, but the concept is very similar. In this paper, we use the Schur form from (4.1) computed by the QR method.

A partial Schur form of order p with $1 \le p \le n$ is defined by

$$AX_p = BX_pS_p$$

where $X_p \in \mathbf{C}^{n\times p}$ is unitary and $S_p \in \mathbf{C}^{p\times p}$ is upper triangular with eigenvalues of $Ax = \lambda Bx$ on the main diagonal.

When A and B are large and sparse, the QR method is not suitable for computing a partial Schur form. One usually employs a projection method, i.e. approximate Schur vectors $\mathcal{U}_k$ are computed in the subspace spanned by the columns of the unitary matrix $\mathcal{V}_k$ so that

$$A\mathcal{U}_k - B\mathcal{U}_kS_k = \mathcal{F}_k \tag{4.2}$$

where $\mathcal{U}_k = \mathcal{V}_kX_k$ and $\mathcal{F}_k$ is a residual term. Using the concept of orthogonal projection, i.e. we force $\mathcal{V}_k^*\mathcal{F}_k = 0$, we find that X_k and S_k satisfy the $k\times k$ Schur problem

$$A_kX_k - B_kX_kS_k = 0 \quad \text{with} \quad A_k = \mathcal{V}_k^*A\mathcal{V}_k \text{ and } B_k = \mathcal{V}_k^*B\mathcal{V}_k\ , \tag{4.3}$$

which can easily be solved by the QR method on $H_k = B_k^{-1}A_k$. We call (4.2) the *approximate partial Schur form.*

We can describe an orthogonal projection method in terms of $\mathcal{U}_k$ instead of $\mathcal{V}_k$ since both form an orthonormal basis for the same subspace. The use of the Schur basis instead of $\mathcal{V}_k$ makes it easier to lock and restart as we shall see later. After the kth iteration we thus have (4.2) with $\mathcal{U}_k^*\mathcal{F}_k = 0$. In the $k+1$ st iteration, a new vector v_{k+1} is added and a new projection matrix H_{k+1} is computed so that

$$A\begin{pmatrix} \mathcal{U}_k & v_{k+1}\end{pmatrix} - B\begin{pmatrix} \mathcal{U}_k & v_{k+1}\end{pmatrix}H_{k+1} = \mathcal{F}_{k+1}$$

with $\begin{pmatrix} \mathcal{U}_k & v_{k+1}\end{pmatrix}^*\mathcal{F}_{k+1} = 0$. Note that the basis $\begin{pmatrix} \mathcal{U}_k & v_{k+1}\end{pmatrix}$ has the Schur vectors in the front which is useful for locking as we shall discuss in §5. In practice, we never compute $\mathcal{U}_k$ at each iteration. Instead, we store $\mathcal{V}_k$ and X_k. Only X_k needs to be computed, which is much cheaper.

4.2 Quadratic problems

The existence of a partial Schur form is guaranteed for the linearized problem (1.2) :

$$\begin{bmatrix} K & 0 \\ 0 & M \end{bmatrix}\begin{pmatrix} U_{2k} \\ Y_{2k}\end{pmatrix} = \begin{bmatrix} -iC & M \\ M & 0 \end{bmatrix}\begin{pmatrix} U_{2k} \\ Y_{2k}\end{pmatrix}S_{2k}\ , \tag{4.4}$$

with $U_{2k}, Y_{2k} \in \mathbf{C}^{n\times 2k}$ and S_{2k} a $2k\times 2k$ upper triangular matrix. Note that U_{2k} does not need to be of full rank. For example, when $C = 0$, and (ω, u) is an eigenpair, $(-\omega, u)$ is also one. The corresponding U matrix is $U_2 = \begin{pmatrix} \alpha u & \beta u\end{pmatrix}$ for some constants α and β which has rank smaller than two.

We define the partial quadratic Schur form as

$$KW_{2k} + iCW_{2k}T_{2k} - MW_{2k}T_{2k}^2 = 0 \tag{4.5}$$

where $W_{2k} \in \mathbf{C}^{n\times 2k}$ is unitary and $T_{2k} \in \mathbf{C}^{2k\times 2k}$ is upper triangular with the eigenvalues on its main diagonal. The following lemma shows a condition for which a partial quadratic Schur form exists.

Lemma 4.1 *If U_{2k} from (4.4) has full rank, then a partial Schur form (4.5) of (1.1) exists.*

Proof From (4.4), it follows that $Y_{2k} = U_{2k}S_{2k}$ and

$$KU_{2k} + iCU_{2k}S_{2k} - MU_{2k}S_{2k}^2 = 0 \ . \tag{4.6}$$

Consider the QR factorization $W_{2k}Z_{2k} = U_{2k}$, with $W_{2k} \in \mathbf{C}^{n\times 2k}$ unitary and $Z_{2k} \in \mathbf{C}^{2k\times 2k}$ upper triangular. Define the upper triangular matrix $T_{2k} = Z_{2k}S_{2k}Z_{2k}^{-1}$. Then (4.5) is satisfied. $\square$

We propose computing (4.4) instead of (4.5), since this Schur form is guaranteed to exist. Due to the relationship between the quadratic problem and its linearization and the corresponding solvers, we can use quadratic residual iteration for building the subspace and computing Ritz pairs.

On the kth iteration, we project the linearized problem (1.2) on the subspace with basis

$$\mathcal{V}_{2k} = \begin{bmatrix} V_k & 0 \\ 0 & V_k \end{bmatrix} \tag{4.7}$$

for computing the Schur basis, where the Schur vectors have the form

$$\mathcal{U}_{2k} = \begin{pmatrix} U_{2k} \\ Y_{2k} \end{pmatrix} = \begin{bmatrix} V_k & 0 \\ 0 & V_k \end{bmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \ . \tag{4.8}$$

With $K_k = V_k^*KV_k$, $C_k = V_k^*CV_k$ and $M_k = V_k^*MV_k$, this projection gives

$$\begin{bmatrix} K_k & \\ & M_k \end{bmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \begin{bmatrix} -iC_k & M_k \\ M_k & \end{bmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} S_{2k} \tag{4.9}$$

from which $X_2 = X_1S_{2k}$. Hence, the approximate partial Schur form is

$$\begin{bmatrix} K & \\ & M \end{bmatrix} \begin{pmatrix} U_{2k} \\ U_{2k}S_{2k} \end{pmatrix} - \begin{bmatrix} -iC & M \\ M & \end{bmatrix} \begin{pmatrix} U_{2k} \\ U_{2k}S_{2k} \end{pmatrix} S_{2k} = \begin{pmatrix} F_{2k} \\ 0 \end{pmatrix}$$

and

$$KU_{2k} + iCU_{2k}S_{2k} - MU_{2k}S_{2k}^2 = F_{2k} \ . \tag{4.10}$$

Note that (even with $F_{2k} = 0$) (4.10) is not a partial quadratic Schur factorization since U_{2k} is not unitary and is not guaranteed to have full rank.

As mentioned before, we can use the Schur basis $\mathcal{U}_{2k}$ instead of $\mathcal{V}_{2k}$. Instead of adding one vector at the $k+1$ st iteration, we now add two vectors to the basis, so the basis at iteration $k+1$ becomes

$$\begin{pmatrix} \mathcal{U}_{2k} & \begin{pmatrix} v_{k+1} & 0 \\ 0 & v_{k+1} \end{pmatrix} \end{pmatrix} \ .$$

Using (4.8), we can rewrite this as

$$\begin{bmatrix} V_k & v_{k+1} & 0 & 0 \\ 0 & 0 & V_k & v_{k+1} \end{bmatrix} \begin{pmatrix} X_1 & 0 & 0 \\ 0 & 1 & 0 \\ X_2 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \ .$$

It is thus sufficient to store V_k, v_{k+1} and X_1, X_2 to represent the basis.

For the computation of the Schur form of the linearized problem, we need $H_{2k} = B_{2k}^{-1} A_{2k}$. This is cheaply computed and in a numerically stable way : let

$$B_{2k} = \begin{bmatrix} -iC_k & M_k \\ M_k & 0 \end{bmatrix} \quad \text{and} \quad A_{2k} = \begin{bmatrix} K_k & \\ & M_k \end{bmatrix} .$$

Then $H_{2k} = B_{2k}^{-1} A_{2k}$ can be computed as follows

Algorithm 4.1 (Computation of $H_{2k} = B_{2k}^{-1} A_{2k}$)
Compute $H_{2k} = \begin{bmatrix} 0 & I \\ M_k^{-1} K_k & iM_k^{-1} C_k \end{bmatrix}$ where the action of M_k^{-1} is performed using the Cholesky factorization $M_k = L_k L_k^*$ with L_k lower triangular.

The matrix $M_k = V_k^* M V_k$ is well conditioned since M is positive definite and has a small condition number.

5 Locking

Locking of converged eigenvalues is widely used in linear eigenvalue calculations (Stewart 1976, Lehoucq and Sorensen 1996, Ruhe 1998, Fokkema et al. 1999). We first explain the idea of locking for linear problems and then apply this to the quadratic problem (1.2).

5.1 Linear problems

The idea of locking is as follows. Suppose that, at the kth iteration, the Schur factorization is reordered so that we have the decomposition

$$A \begin{pmatrix} \mathcal{U}_q & \mathcal{U}_{k-q} \end{pmatrix} - B \begin{pmatrix} \mathcal{U}_q & \mathcal{U}_{k-q} \end{pmatrix} \begin{bmatrix} S_q & S_{q,k-q} \\ 0 & S_{k-q} \end{bmatrix} = \begin{pmatrix} \mathcal{F}_q & \mathcal{F}_{k-q} \end{pmatrix} , \tag{5.1}$$

with $\|\mathcal{F}_q\|$ smaller than the convergence tolerance. We consider the first q Ritz values and corresponding Schur vectors as converged. In fact, we assume that $\mathcal{F}_q = 0$. The first q Schur vectors $\mathcal{U}_q$ and the Schur matrix S_q are fixed or 'locked' in the subsequent iterations, i.e. when new directions are added to the subspace, and the Schur vectors are recomputed, the locked vectors are not changed. On the $k+1$ st iteration, after the addition of v_{k+1}, we have a new basis $\begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix} = \begin{pmatrix} \mathcal{U}_q & \mathcal{U}_{k-q} & v_{k+1} \end{pmatrix}$ and, after projection, we have the projection equation

$$A \begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix} - B \begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix} \begin{bmatrix} H_q & H_{q,k-q+1} \\ H_{k-p+1,q} & H_{k-q+1} \end{bmatrix} = \begin{pmatrix} \mathcal{G}_q & \mathcal{G}_{k-q+1} \end{pmatrix} \tag{5.2}$$

with $\begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix}^* \begin{pmatrix} \mathcal{G}_q & \mathcal{G}_{k-q+1} \end{pmatrix} = 0$, and with the projection matrix

$$\begin{bmatrix} H_q & H_{q,k-q+1} \\ H_{k-q+1,q} & H_{k-q+1} \end{bmatrix} = B_k^{-1} A_k \quad \text{and} \tag{5.3a}$$

$$A_k = \begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix}^* A \begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix} \tag{5.3b}$$

$$B_k = \begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix}^* B \begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix} . \tag{5.3c}$$

Since $\|F_q\|$ is small, we can replace the projection matrix by

$$\begin{bmatrix} S_q & H_{q,k-q+1} \\ & H_{k-q+1} \end{bmatrix}$$

without making a big error. This is deflation or locking. The error made is given by the following theorem.

Theorem 5.1 *Let*

$$\begin{aligned} \mathcal{U}_q^* A \mathcal{U}_q - \mathcal{U}_q^* B \mathcal{U}_q S_q &= 0 & (5.4) \\ A\mathcal{U}_q - B\mathcal{U}_q S_q &= \mathcal{F}_q & (5.5) \end{aligned}$$

and let $\begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix}$ *be unitary. Then, with the definition of (5.3),*

$$\left\| \begin{bmatrix} H_q & H_{q,k-q+1} \\ H_{k-q+1,q} & H_{k-q+1} \end{bmatrix} - \begin{bmatrix} S_q & H_{q,k-q+1} \\ 0 & H_{k-q+1} \end{bmatrix} \right\|_F \leq \|B_k^{-1}\| \|\mathcal{V}_{k-q+1}^* \mathcal{F}_q\|_F \ .$$

Proof We drop the subscripts for $\mathcal{U}_q$ and $\mathcal{V}_{k-q+1}$. From (5.3), it follows that

$$\begin{aligned} \mathcal{U}^* A \mathcal{U} - \mathcal{U}^* B \mathcal{U} H_q - \mathcal{U}^* B \mathcal{V} H_{k-q+1,q} &= 0 \\ \mathcal{V}^* A \mathcal{U} - \mathcal{V}^* B \mathcal{U} H_q - \mathcal{V}^* B \mathcal{V} H_{k-q+1,q} &= 0 \ . \end{aligned}$$

From (5.4) and (5.5), we have that

$$\begin{aligned} \mathcal{U}^* B \mathcal{U}(S_q - H_q) - \mathcal{U}^* B \mathcal{V} H_{k-q+1,q} &= 0 \\ \mathcal{V}^* B \mathcal{U}(S_q - H_q) - \mathcal{V}^* B \mathcal{V} H_{k-q+1,q} &= -\mathcal{V}^* \mathcal{F}_q \ . \end{aligned}$$

This gives the linear system

$$\left(\begin{pmatrix} \mathcal{U} & \mathcal{V} \end{pmatrix}^* B \begin{pmatrix} \mathcal{U} & \mathcal{V} \end{pmatrix} \right) \begin{pmatrix} S_q - H_q \\ H_{k-q+1,q} \end{pmatrix} = \begin{pmatrix} 0 \\ -\mathcal{V}^* \mathcal{F}_q \end{pmatrix} ,$$

which implies that

$$\|S_q - H_q\|_F^2 + \|H_{k-q+1,q}\|_F^2 \leq \left\| \left(\begin{pmatrix} \mathcal{U} & \mathcal{V} \end{pmatrix}^* B \begin{pmatrix} \mathcal{U} & \mathcal{V} \end{pmatrix} \right)^{-1} \right\|_2^2 \|\mathcal{V}^* \mathcal{F}_q\|_F^2 \ .$$

This completes the proof. $\square$

The projection equation can now be decomposed as

$$\begin{aligned} A \begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q} \end{pmatrix} \ &- \ B \begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix} \begin{bmatrix} S_q & H_{q,k-q+1} \\ 0 & H_{k-q+1} \end{bmatrix} & (5.6) \\ &= \begin{pmatrix} \mathcal{F}_q & \mathcal{F}_{k-q+1} \end{pmatrix} \ . \end{aligned}$$

The new Schur vectors must have the form

$$\begin{pmatrix} \mathcal{U}_q & \mathcal{V}_{k-q+1} \end{pmatrix} \begin{bmatrix} I & 0 \\ 0 & Z_{k-q+1} \end{bmatrix} \ .$$

By multiplying (5.6) on the right by $\begin{pmatrix} I & 0 \\ 0 & Z_{k-q+1} \end{pmatrix}$, it follows that with $\mathcal{U}_{k-q+1} = \mathcal{V}_{k-q+1} Z_{k-q+1}$

$$A \begin{pmatrix} \mathcal{U}_q & \mathcal{U}_{k-q+1} \end{pmatrix} - B \begin{pmatrix} \mathcal{U}_q & \mathcal{U}_{k-q+1} \end{pmatrix} \begin{bmatrix} S_q & H_{q,k-q+1} Z_{k-q+1} \\ & S_{k-q+1} \end{bmatrix}$$
$$= \begin{pmatrix} \mathcal{F}_q & \mathcal{F}_{k-q+1} Z_{k-q+1} \end{pmatrix}$$

where $H_{k-q+1} Z_{k-q+1} = Z_{k-q+1} S_{k-q+1}$ is the Schur form of H_{k-q+1}. This allows us to compute Z_{k-q+1} and S_{k-q+1}.

5.2 Quadratic problems

Locking in quadratic residual iteration can be performed via the linearized problem. The mathematics in the last section remains, but two vectors instead of only one are added at each iteration. It is also important to note that it is not necessary to store $\mathcal{U}_{2k}$, but only V_k, X_{2k} and S_{2k}. Locking then corresponds to locking the first q columns of X_{2k} and S_{2k}.

6 Restarting

The number of basis vectors, k, grows as the algorithms proceed. In practice, this number is limited by storage and computational costs for the Gram-Schmidt orthogonalization. It is possible that convergence to the desired eigenpairs has not occurred when this limit is reached. One way to solve this problem is to reduce the dimension of the subspace by throwing away the uninteresting part. This is called purging.

6.1 Linear problems

We can keep the locked Schur vectors as well as the Schur vectors of interest in the subspace and throw away those we are not interested in. This idea was used for restarting the block Arnoldi method (Scott 1995), the implicitly restarted Arnoldi method (Lehoucq and Sorensen 1996, Morgan 1996), the Jacobi-Davidson method (Fokkema et al. 1999), and the rational Krylov method (Ruhe 1998, De Samblanx et al. 1997). The main idea is to reorder the Schur form so that unwanted Schur vectors are moved towards the end of the matrix of Schur vectors and to cut the Schur factorization after the pth column. So,

$$A \begin{pmatrix} \mathcal{U}_p & \mathcal{U}_{k-p} \end{pmatrix} - B \begin{pmatrix} \mathcal{U}_p & \mathcal{U}_{k-p} \end{pmatrix} \begin{bmatrix} S_p & S_{p,k-p} \\ 0 & S_{k-p} \end{bmatrix} = \begin{pmatrix} \mathcal{F}_p & \mathcal{F}_{k-p} \end{pmatrix}$$

is reduced to

$$A\mathcal{U}_p - B\mathcal{U}_p S_p = \mathcal{F}_p$$

by purging the last $k-p$ Schur vectors. This equation can be expanded by adding new vectors.

In practice, the Schur vectors are computed from the projected Schur form (4.3) as $\mathcal{U}_k = \mathcal{V}_k X_k$, by decomposing $X_k = \begin{pmatrix} X_p & X_{k-p} \end{pmatrix}$ and using $\mathcal{V}_p = \mathcal{V}_k X_p$ as the new basis.

6.2 Quadratic problems

We want to reduce the approximate Schur form

$$A\begin{pmatrix} U_{2k} \\ U_{2k}S_{2k} \end{pmatrix} - B\begin{pmatrix} U_{2k} \\ U_{2k}S_{2k} \end{pmatrix} S_{2k} = \begin{pmatrix} F_{2k} \\ 0 \end{pmatrix}$$

to

$$A\begin{pmatrix} U_{2p} \\ U_{2p}S_{2p} \end{pmatrix} - B\begin{pmatrix} U_{2p} \\ U_{2p}S_{2p} \end{pmatrix} S_{2p} = \begin{pmatrix} F_{2p} \\ 0 \end{pmatrix}$$

by chopping off the last $2k-2p$ columns of U_{2k} and S_{2k}. This is not possible since the new basis must have the form

$$\mathcal{Y}_{2p} = \begin{bmatrix} W_p & 0 \\ 0 & W_p \end{bmatrix} \begin{pmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{pmatrix} .$$

We can, however, keep the first p columns of U_{2k}, denoted by U_p, as follows. Consider the QR factorization $U_p = W_p R_p$ with W_p unitary and R_p upper triangular. Let $Z_{11} = R_p$ and $Z_{21} = R_p S_p$ with S_p the upper left $p \times p$ submatrix of S_{2p}. The remaining blocks Z_{12} and Z_{22} must be chosen so that

$$\begin{pmatrix} R_p & Z_{12} \\ R_p S_p & Z_{22} \end{pmatrix} \tag{6.1}$$

is square and unitary. The last p columns do not represent Schur vectors but are necessary to keep the basis in the form (4.7).

In practice, we proceed as follows.

Algorithm 6.1

1. Let the first p columns of $U_{2k} = V_k X_1$ be U_p and let $U_p = V_k X_{11}$.

2. Compute the QR factorization $X_{11} = QR_p$.

3. Decompose

$$\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix}$$

with $X_{11}, X_{21} \in \mathbf{C}^{k\times p}$ and $X_{12}, X_{22} \in \mathbf{C}^{k\times 2k-p}$.

4. Apply Q

$$\begin{pmatrix} Q^*X_1 \\ Q^*X_2 \end{pmatrix} = \begin{pmatrix} R_p & Q^*X_{12} \\ R_pS_p & Q^*X_{22} \end{pmatrix} .$$

Remove the last $2k-2p$ columns of X_1 and X_2.

5. Orthogonalize this matrix to get (6.1).

Note that the first p columns of $\mathcal{Y}_{2p}$ are the same as those of $\mathcal{U}_{2k}$. When the first q columns of $\mathcal{U}_{2k}$ are locked Schur vectors and $q \le p$, then

$$A\mathcal{Y}_{2p} - BY_{2p} \begin{bmatrix} S_q & H_{q,2p-q} \\ 0 & H_{2p-q} \end{bmatrix} = \begin{pmatrix} \mathcal{F}_q & \mathcal{G}_{2p-q} \end{pmatrix} ,$$

where the lower left block in the Schur matrix is forced to be zero. (See Theorem 5.1.)

7 Building the Schur factorization

We have discussed how to compute an approximate partial Schur form by projection on a subspace, how to use the partial Schur form for restarting purposes and for locking eigenvectors (or Schur vectors). We still have to discuss which vectors will be added to the subspace. Instead of building a basis of eigenvectors in the subspace Range(V_k), we can build a partial Schur form. A Schur form always exists, while the eigendecomposition does not. Moreover, Schur vectors are orthogonal which is in favour of the numerical stability of the method. In this section, we discuss how we can expand a partial Schur form.

7.1 Linear problems

The JDQR and JDQZ methods (Fokkema et al. 1999) for solving linear eigenvalue problems compute the partial Schur form $A\mathcal{U}_k = B\mathcal{U}_k S_k$ column by column. Decompose the partial Schur form into

$$A \begin{pmatrix} \mathcal{U}_{k-1} & x \end{pmatrix} = B \begin{pmatrix} \mathcal{U}_{k-1} & x \end{pmatrix} \begin{bmatrix} S_{k-1} & s \\ & \omega \end{bmatrix} .$$

In order to simplify the notation, we use S to denote S_{k-1} and $\mathcal{U}$ for $\mathcal{U}_{k-1}$. Define $\mathcal{Q}$ so that $\mathcal{Q}^* B\mathcal{U} = I$. As a result, $I - B\mathcal{U}\mathcal{Q}^*$ is a projector that maps $B\mathcal{U}$ to zero. Also note that $B\mathcal{U}\mathcal{Q}^*(A - \omega B)x = B\mathcal{U}s$. We will discuss later how to choose $\mathcal{Q}$. Since $(I - \mathcal{U}\mathcal{U}^*)x = x$, we can consider (ω, x) as an eigenpair of

$$(I - B\mathcal{U}\mathcal{Q}^*)A(I - \mathcal{U}\mathcal{U}^*)x = \omega(I - B\mathcal{U}\mathcal{Q}^*)B(I - \mathcal{U}\mathcal{U}^*)x . \tag{7.1}$$

Note the resemblance with the Jacobi-Davidson correction equation in (3.1). Suppose $\mathcal{U}$ and S are already computed and we wish to compute x, s and ω. Suppose an approximation for (ω, x) is available that satisfies the approximate Schur factorization

$$A \begin{pmatrix} \mathcal{U} & x \end{pmatrix} - B \begin{pmatrix} \mathcal{U} & x \end{pmatrix} \begin{bmatrix} S & s \\ & \omega \end{bmatrix} = \begin{pmatrix} 0 & r \end{pmatrix} ,$$

with $\mathcal{Q}^* r = 0$. We want to compute an improvement by using the Cayley transform on (7.1). The Cayley transform y is solved from the system

$$(I - B\mathcal{U}\mathcal{Q}^*)(A - \sigma B)(I - \mathcal{U}\mathcal{U}^*)y = (I - B\mathcal{U}\mathcal{Q}^*)(A - \omega B)(I - \mathcal{U}\mathcal{U}^*)x .$$

Since $\mathcal{U}^* x = 0$ and assuming that $\mathcal{U}^* y = 0$ we get

$$(I - B\mathcal{U}\mathcal{Q}^*)(A - \sigma B)y = (I - B\mathcal{U}\mathcal{Q}^*)r , \tag{7.2}$$

so

$$(A - \sigma B)y = r + B\mathcal{U}z \tag{7.3}$$

where $z = \mathcal{Q}^*(A - \sigma B)y$ is chosen so that $\mathcal{U}^* y = 0$. If $A - \sigma B$ is invertible the solution can be written as

$$y = (A - \sigma B)^{-1} B\mathcal{U}z + (A - \sigma B)^{-1} r .$$

Since $(A-\sigma B)^{-1}B\mathcal{U} = \mathcal{U}(S-\sigma I)^{-1}$,

$$y = \mathcal{U}\tilde{z} + (A-\sigma B)^{-1}r \tag{7.4}$$

where $\tilde{z}$ is such that $y \perp \mathcal{U}$. The second term in the right-hand side can be considered as the Cayley transform for the Schur vector x.

7.2 Quadratic problems

In quadratic residual iteration, we want to expand the subspace by the first n components of y without explicitly solving a linear system with $A-\sigma B$. With U denoting U_{k-1}, recall from $\mathcal{U}_k = \begin{pmatrix} U_k \\ U_k S_k \end{pmatrix}$ that $\mathcal{U} = \begin{pmatrix} U \\ US \end{pmatrix}$ and $x = \begin{pmatrix} u \\ Us+\omega u \end{pmatrix}$. Decompose $r = \begin{pmatrix} r_1 \\ r_2 \end{pmatrix}$, then from $r = (A-\omega B)x - B\mathcal{U}s$, we derive

$$\begin{aligned} r_1 &= Ku + i\omega Cu - \omega^2 Mu + iCUs - \omega MUs - MUSs \\ r_2 &= 0\ . \end{aligned}$$

Note that r_1 is the last column of the approximate partial Schur factorization

$$\begin{aligned} &K\begin{pmatrix} U & u \end{pmatrix} + iC\begin{pmatrix} U & u \end{pmatrix}\begin{bmatrix} S & s \\ & \omega \end{bmatrix} \\ &-M\begin{pmatrix} U & u \end{pmatrix}\begin{bmatrix} S & s \\ & \omega \end{bmatrix}^2 = \begin{pmatrix} 0 & r_1 \end{pmatrix}\ . \end{aligned}$$

With $y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$, (7.3) becomes

$$\begin{aligned} (K+i\sigma C)y_1 - \sigma My_2 &= r_1 - iCUz + MUSz && (7.5a) \\ -\sigma My_1 + My_2 &= MUz\ . && (7.5b) \end{aligned}$$

Decompose $\mathcal{Q} = \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix}$, then $\mathcal{Q}^* r = 0 = Q_1^* r_1$ and

$$z = \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix}^* \begin{pmatrix} (K+i\sigma C)y_1 - \sigma My_2 \\ -\sigma My_1 + My_2 \end{pmatrix}\ .$$

By multiplying (7.5b) by σ and adding to (7.5a), we get

$$\begin{aligned} (K+i\sigma C-\sigma^2 M)y_1 &= r_1 + (-iCU + MU(S+\sigma I))z && (7.6) \\ -\sigma My_1 + My_2 &= MUz \end{aligned}$$

with similarly

$$z = \begin{pmatrix} Q_1 \\ Q_2 - \sigma Q_1 \end{pmatrix}^* \begin{pmatrix} (K+i\sigma C-\sigma^2 M)y_1 \\ -\sigma My_1 + My_2 \end{pmatrix}\ .$$

The dependence on y_2 disappears when $Q_2 = \sigma Q_1$. The equation (7.4) becomes

$$\begin{aligned} y_1 &= U\tilde{z} + (K+i\sigma C-\sigma^2 M)^{-1} r_1 \\ y_2 &= US\tilde{z} + \sigma(K+i\sigma C-\sigma^2 M)^{-1} r_1 \end{aligned}$$

and shows that y_1 and y_2 point in the same direction orthogonal to U. In a projection method, it is only this direction that matters, so it is sufficient to add y_1 to the subspace. By rearranging (7.6) we obtain

$$(I - (-iCU + MU(S + \sigma I))Q_1^*)(K + i\sigma C - \sigma^2 M)y_1 = r_1$$

where we still have the condition that $\mathcal{U}^* y = 0$. Since $\tilde{z}$ is determined by this condition, but since $\mathcal{U}\tilde{z}$ does not contribute to the expansion of the subspace, we can as well use the orthogonality condition $U^* y_1 = 0$, which produces another $\tilde{z}$, but produces the same component of y_1 orthogonal to U. This also produces y_2 with the same direction orthogonal to U.

In practice we solve the equation

$$(I - PQ^*)(K + i\sigma C - \sigma^2 M)(I - ZZ^*)y_1 = r_1 \ , \tag{7.7}$$

by an iterative method with $P = iCU - MU(S + \sigma I)$ and $ZT = U$ the QR factorization of U. We still have to make a choice of Q. When U and S are computed by orthogonal projection within quadratic residual iteration or Jacobi-Davidson, the obvious choice is $Q = ZR$, since $Z^* r_1 = 0$ where R is chosen so that $Q^* P = I$. The problem is that PQ^* is an oblique projector and is not guaranteed to exist. A more robust choice is $Q = PR$, but then we first must orthogonalize r_1 against the columns of Q. In our numerical experiments we used $Q = ZR$.

8 Algorithm and numerical examples

The first two examples are easily generated and are included to allow the reader to reproduce the results. The last example is the numerical simulation of poro-elastic material and is less easily reproduced. This is added to illustrate the method on a more realistic problem.

The algorithm that we use for the computation of Ritz pairs is the following one. It uses locking of converged Schur vectors and purging for restarting purposes. On each iteration, the Schur basis of the projected linearized problem is computed. The basis vectors are obtained by Gram-Schmidt orthogonalization with reorthogonalization (Daniel, Gragg, Kaufman and Stewart 1976). The major work lies in the solution of the linear system in Step 2.10 and the Gram-Schmidt orthogonalization in Step 2.11. When the number of vectors becomes large, the operations on the small scale linearized problem may also become significant. We allow σ to be chosen differently at each iteration. When a direct linear solver is user, however, we prefer to keep σ unchanged for a number of iterations in order to avoid a factorization on each iteration.

Algorithm 8.1 In this algorithm, k is the actual dimension of the subspace, q is the number of locked Schur vectors, satisfying $\|KU_q e_j + iCU_q S_q e_j - MU_q S_q^2 e_j\| \leq \mathrm{TOL}$ for $j = 1, \ldots, q$, where $\left\| \begin{pmatrix} U_q \\ U_q S_q \end{pmatrix} \right\| = 1$, p is the dimension after a restart, and m is the maximum subspace dimension.

1. Given $v_1 \in \mathbf{C}^n$ with $\|v_1\| = 1$.

Let $k = 1$ and $q = 0$.

2. Until $q \geq$ number of wanted eigenvalues :
   2.1. Compute the kth row and column of $K_k = V_k^* K V_k$, $C_k = V_k^* C V_k$, and $M_k = V_k^* M V_k$.
   2.2. Compute H_{2k} by Algorithm 4.1.
   2.3. Map H_{2k} in the Schur basis :

$$H_{2k} = \begin{pmatrix} X_1 & 0 & 0 \\ 0 & 1 & 0 \\ X_2 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}^* H_{2k} \begin{pmatrix} X_1 & 0 & 0 \\ 0 & 1 & 0 \\ X_2 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

   2.4. Compute the Schur form $H_{2k} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} S_{2k}$ where the first q Schur vectors are locked.
   2.5. Order the Schur form so that the main diagonal elements of S_{2k} nearest σ are in the upper left position.
   2.6. For $j = q + 1$ to k :
      2.6.1 Compute the jth column of $R_k = KV_k X_1 + iCV_k X_1 S_{2k} - MV_k X_1 S_{2k}^2$.
      2.6.3. If $\|R_k e_j\| \leq$ TOL then $q = q + 1$.
      Else Go to 2.7.

      End for
   2.7. If $k \geq m$ then
      2.7.2. Compute the QR factorization $X_1 = Q_k R_k$.
      2.7.3. Compute the basis vectors $V_p = V_k Q_p$.
      2.7.4. Update the Schur vectors : $X_1 = R_p$ and $X_2 = Q_p^* X_2$.
      2.7.5. Add columns to X_1 and X_2 so that they have appropriate size and form an orthonormal basis (see Algorithm 6.1).
      2.7.6. Update : $K_p = Q_p^* K_k Q_p$, $C_p = Q_p^* C_k Q_p$, $M_p = Q_p^* M_k Q_p$.
   2.8. Increase the subspace dimension $k = k + 1$.
   2.9. Select a pole σ.
   2.10. Solve the linear system $(K + i\sigma C - \sigma^2 M)w = R_k e_{q+1}$.
   2.11. Orthogonalize w against $v_1, \ldots, v_k$ by Gram-Schmidt into v_{k+1}.

For our examples, K, C and M are real symmetric matrices, so, in Step 2.1 it is sufficient to compute the kth column. The subspace is expanded by a quadratic Cayley transform applied to a Schur vector. When an iterative solver is used, we can expand the subspace by solving the correction equation (3.2) instead of the Cayley linear system in Step 2.10, and we also take into account the locked Schur vectors by employing (7.7) where Z also contains the last (unlocked) Schur vector and Q is the corresponding left hand-side basis. (See §9 and §10 for examples.)

8.1 A 'linear' problem

For the first example, M is the identity matrix, C is zero, and K is the diagonal matrix with $1^2, 2^2, \ldots, n^2$ on its main diagonal for $n = 1000$. The 10 eigenvalues nearest zero are $\pm j$ for $j = 1, \ldots, 5$ and the corresponding eigenvectors are e_j for the eigenvalue $\pm j$ where e_j is the jth identity vector. Note that the Schur form, defined by (4.5) does not exist, but it does exist for the linearized problem. Incidentally, the linearized problem has $2n$ eigenvectors. We can solve this problem by the spectral transformation Lanczos method (Ericsson and Ruhe 1980), since $C = 0$, but we want to illustrate that the algorithm is able to compute linearly dependent eigenvectors of the quadratic eigenvalue problem.

We computed the 10 eigenvalues nearest zero by the use of Algorithm 8.1 with a fixed pole $\sigma = 0$ and $m = 30$ iteration vectors. The initial vector v_1 was $1/\sqrt{n}$ everywhere. All the eigenvalues were computed within 20 iterations to a tolerance (TOL) of 10^{-7}. When only $m = 17$ vectors are used and we compress the dimension to $p = 15$, then before the first restart, 8 eigenvalues have converged. For the convergence of the remaining two eigenvalues, two restarts are required. In total, this corresponds to 20 iterations, which is exactly the same number as for the run with $m = 30$ vectors. This shows that the purging of the Ritz values far away from the pole σ does not necessarily slow down the convergence. A similar result was shown for the implicitly restarted Arnoldi method (Morgan 1996) and for the implicitly filtered rational Krylov method (De Samblanx et al. 1997).

8.2 A simple quadratic problem

We use the same K and M as for the previous example, but now $C = \alpha I$ with $\alpha = 0.1$. We again computed the 10 eigenvalues nearest zero. We used the same parameters as for the previous example, i.e. $m = 17$, $p = 15$, $\sigma = 0$ and a tolerance of 10^{-7}. The eigenvalues of this example and the previous one are related as follows.

Lemma 8.1 *Let $u \neq 0$ and $Ku + i\omega Cu - \omega^2 Mu = 0$ then there is an eigenvalue λ of $Kv = \lambda^2 Mv$, so that*

$$|\lambda - \omega| \leq |\omega| \|M^{-1}\|^{1/2} \frac{\|Cu\|}{u^* Mu} .$$

Proof Let L be so that $M = L^* L$. Define $v = Lu$ so that

$$\begin{aligned} Ku - \omega^2 Mu &= -i\omega Cu \\ L^{-*} K L^{-1} v - \omega^2 v &= -i\omega L^{-*} Cu . \end{aligned}$$

The proof follows from the Bauer-Fike Theorem (Theorem 2.4) with $\kappa = 1$, since $L^{-*} K L^{-1}$ is Hermitian. Note that $\|L^{-*}\| = \|M^{-1}\|^{1/2}$. □

Since $\|C\| = 0.1$ the eigenvalues lie close to those of the previous example. Therefore, the convergence is very similar. It took 20 iterations to compute the ten Ritz values. The first ten Ritz values are given by Table 8.1.

Table 8.1: Ritz values for the example in §8.2

real part	imaginary part	residual norm
0.99874921	0.0500000	$6.94\,10^{-8}$
−0.99874921	0.0500000	$6.94\,10^{-8}$
1.9993749	0.0500000	$9.58\,10^{-8}$
−1.9993749	0.0500000	$9.57\,10^{-8}$
−2.9995833	0.0500000	$2.88\,10^{-8}$
2.9995833	0.0500000	$2.88\,10^{-8}$
3.9996875	0.0500000	$9.75\,10^{-9}$
−3.9996875	0.0500000	$9.87\,10^{-9}$
−4.9997500	0.0500000	$2.84\,10^{-8}$
4.9997500	0.0500000	$2.85\,10^{-8}$

8.3 Acoustic simulation of poro-elastic material

This example is related to the acoustic simulation of a $0.4m \times 0.4m \times 0.06m$ sample made of a poro-elastic material. The material is modelled using a two-phase Biot model accounting for kinematic and mechanical interactions between the (elastic) skeleton and the pore (acoustic) fluid (Sandhu and Pister 1970, Simon, Wu, Zienkiewicz and Paul 1986). The following material properties have been selected : for the skeleton, the Young modulus is $140000N/m^2$, the Poisson ratio $0.35(-)$, and the density $1300kg/m^3$. The pore fluid has density $1.225kg/m^3$, the sound speed is $340m/s$, the porosity $0.95(-)$, the flow resistivity is $5000Ns/m^4$, the Biot factor is $1(-)$, the fluid bulk modulus $141600N/m2$, and the tortuosity is $1.2(-)$. The discrete finite-element model relies on a u-w formulation (Simon et al. 1986) where skeleton displacement components (u) and relative fluid displacement components (weighted by the local porosity) (w) are selected as nodal variables. The finite-element mesh has 324 nodes and 192 HEXA8 elements. The total number of degrees of freedom is 1944.

We used Algorithm 8.1 with $m = 30$, $p = 15$ and the number of wanted Ritz values 10. The first pole was 300. A new pole σ was selected after every 10 iterations, as

$$\sigma = \min_{j \geq q+1} \{\mu = \mathrm{Re}((\omega_j + \omega_{j+1})/2), |\omega_j - \mu| \geq 10\} \ ,$$

where q is the number of converged Ritz values so that the pole is never close to a Ritz value. This prevents the matrix factorization of almost singular matrices. The tolerance for the eigenvalue problem was $\mathrm{TOL} = 10^{-5}$. (This tolerance is relatively large, since the initial residual norm $\|r_P\|$ is of the order 1.)

The results are obtained by a direct and an iterative method for solving the linear system in Step 2.12 of Algorithm 8.1. The direct solver is the package ME47 from the Harwell Subroutine Library (HSL 1996). The iterative solver is GMRES (Saad and Schultz 1986). The linear systems are solved with a relative residual tolerance $\tau = 10^{-2}$ and $\tau = 10^{-4}$, i.e. $(K + i\sigma C - \sigma^2 M)w = r_P + s$ and $\|s\| \leq \tau \|r_P\|$ where s is the residual of the linear system and $r_P = R_k e_{q+1}$ is the quadratic residual of the Ritz pair. The real part $K - \sigma^2 M$ was used as preconditioner, where the HSL package MA47 was used for

Table 8.2: Number of converged Ritz pairs for some iteration numbers when a direct and iterative linear solver are used. τ is the relative residual tolerance of the iterative method.

iteration	number of converged Ritz values		
	direct	iterative $\tau = 10^{-2}$	iterative $\tau = 10^{-4}$
10	1	1	1
20	3	3	4
30	6	5	6
40	8	7	8
46	10		10
50		9	
59		10	

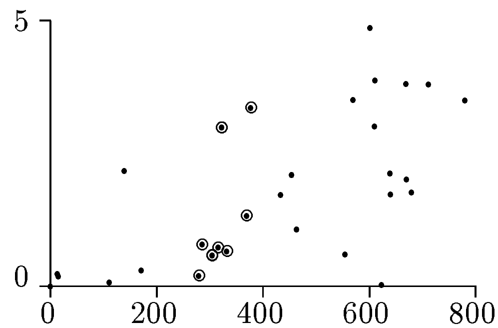

Figure 8.1: The eigenvalues of the acoustic simulation of a poro-elastic material between 0 and 800 Hz are shown as dots. The Ritz values computed by Algorithm 8.1 are denoted as circles. Only 8 circles are shown, since there are two pairs of double eigenvalues. Notice the different scales for real and imaginary axes.

the solution of the corresponding linear system. The preconditioner is perhaps not very advisable for practical computations, but this example shows that an iterative method can be used. Recall from §2 that the quadratic Cayley transform need not be computed very accurately for fast convergence.

The eigenvalues and computed Ritz values are displayed in Figure 8.1. The iterative solver for $\tau = 10^{-2}$ required between 21 and 27 iterations to attain the required residual tolerance. From the numerical results in Table 8.2, it can be seen that about the same number of Ritz values have converged after 40 iterations, independent of the choice of linear solver. From the 40th iteration on, there is a significantly different convergence behaviour. Quadratic residual iteration requires 46 iterations when a direct linear solver is used and 59 iterations when the iterative solver with $\tau = 10^{-2}$ is used. Note that the relative residual tolerance $\tau = 10^{-2}$ for GMRES is quite large. Accurate solves are not necessary. With $\tau = 10^{-4}$, the convergence speed is the same as when a direct linear system solver is used, but the cost per iteration is higher than when $\tau = 10^{-2}$.

9 Jacobi-Davidson for the quadratic eigenvalue problem

In this section, we compare the Jacobi-Davidson method with quadratic residual iteration. The only difference with Algorithm 8.1 is in Step 2.12, where we solve the correction equation (3.2) instead of computing a Cayley transformation. Since we compute more than one eigenpair, we also add the locked Schur vectors to this basis, i.e. we solve a problem of the form (7.7), where Z spans the $q+1$ first columns of $V_k X_1$ and Q the first $q+1$ columns of $iC(V_k X_1) - M(V_k X_1)(S_{2k} + \sigma I)$.

We now compare the following methods for the application from §8.3. The purpose is to compute the ten eigenvalues nearest 300.

QRI-D Quadratic residual iteration with a direct linear solver. The solver used is ME47 from the HSL library. The pole changes every 10 iterations. The first pole is 300.

QRI-G Quadratic residual iteration with GMRES(30) as linear solver, preconditioned by the direct solver MA47 from the HSL library for $K - \sigma^2 M$. The pole σ changes every 10 iterations. The first pole is 300.

QRI-GP Quadratic residual iteration with GMRES(30) applied to (7.7), with Z and Q defined as above, preconditioned by MA47. The pole σ changes every 10 iterations. The first pole is 300.

QJD Jacobi-Davidson where the linear systems are solved by GMRES(30) preconditioned by MA47. The first ten iterations, the pole was kept constant to 300. From the eleventh iteration on, the pole $\sigma = \omega$ and changes each iteration. The preconditioner changes every ten iterations, so we factorize only a few times.

Table 9.1 contains results for the methods listed above for different values of the tolerance of the linear solver. The bold values give the minimum number of linear solves for each method. Clearly, QRI-D is far much faster than any other method, but this is because the linear systems with $K + i\sigma C - \sigma^2 M$ are solved exactly. Note, however, that a direct solver for complex linear systems is used while the other methods use a direct solver for real matrices. Let us concentrate on the results obtained with GMRES. The preconditioner is the real part of $K + i\sigma C - \sigma^2 M$, which is in this case $K - \sigma^2 M$. Note that the factorization cost is about the same for all methods since factorization is only performed every ten outer iterations. For QRI-G and QRI-GP, the linear systems need not be solved very accurately to obtain fast convergence. It is remarkable that the number of outer iterations is much larger for the optimal cases. For $\tau = 10^{-6}$ and $\tau = 10^{-8}$ the linear systems are solved more accurately than necessary, since the number of outer iterations does not change significantly. Using a larger τ gives roughly the same number of outer iterations, but less global work. Also note that QRI-GP is more efficient than QRI-G when τ is smaller. We also tried QRI-G with $\sigma \equiv \omega$ as for QJD, but then GMRES stagnates rather quickly. Without the projections in the correction equation (3.2), the matrix of the linear system has an eigenvalue near zero that hinders the convergence. The projections filter away this eigenvalue. For small τ, QJD is definitely faster than any other algorithm. For large τ, QJD stagnates after the locking of the first Ritz value.

Table 9.1: The number of linear solves for the different algorithms.

Method	Linear solver tolerance τ	Direct linear solves	Eigensolver iterations
QRI-D	—	**45**	46
QRI-G	10^{-2}	**1412**	58
QRI-G	10^{-3}	1545	48
QRI-G	10^{-4}	1976	45
QRI-G	10^{-6}	2998	47
QRI-G	10^{-8}	4130	47
QRI-GP	10^{-2}	**1432**	57
QRI-GP	10^{-3}	1499	47
QRI-GP	10^{-4}	1739	46
QRI-GP	10^{-6}	2318	46
QRI-GP	10^{-8}	3503	45
QJD	10^{-2}	—	—
QJD	10^{-3}	—	—
QJD	10^{-4}	**1469**	33
QJD	10^{-6}	2208	27
QJD	10^{-8}	2918	25

The first and second eigenvalue form a double one at $303.624 + 0.589i$ and it seems to be difficult to find the second of the double eigenvalue when τ is large. A more important danger arises in the following situation. When we use $\tau = 10^{-8}$ and change the pole after the 6th iteration instead of after the 10th, the Ritz value is $\omega \simeq 303.624 + 0.589i$ and QJD tries to improve the corresponding Ritz vector by solving (7.7). The problem is that there is an eigenvalue at $303.624 + 0.589i$ with multiplicity two, while only one eigenvector is filtered away by the projector $I - ZZ^*$. This means that the linear system is still nearly singular and GMRES stagnates.

10 Shift-invert Arnoldi for the linearized problem

The most widely used method in applications is probably the shift-invert Arnoldi method applied to the linearized problem (1.2). This method is criticized because it doubles the size of the problem, leading to an increase in storage cost for the iteration vectors $\mathcal{V}_k$ and in the overall computational cost. The shift-invert transformation, $y = (A - \sigma B)^{-1}Bx$, however, can be efficiently computed by the solution of the block structured linear system

$$\begin{aligned} (A - \sigma B)y &= Bx \\ \begin{bmatrix} K + i\sigma C & -\sigma M \\ -\sigma M & M \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} &= \begin{pmatrix} -iCx_1 + Mx_2 \\ Mx_1 \end{pmatrix} . \end{aligned}$$

By multiplying the last row by σ and adding to the first row, we get

$$\begin{bmatrix} K + i\sigma C - \sigma^2 M & 0 \\ -\sigma M & M \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} -iCx_1 + M(x_2 + \sigma x_1) \\ Mx_1 \end{pmatrix} ,$$

from which

$$\begin{aligned} y_1 &= (K + i\sigma C - \sigma^2 M)^{-1}(-iCx_1 + M(x_2 + \sigma x_1)) \\ y_2 &= x_1 + \sigma y_1 . \end{aligned}$$

This requires the matrix factorization of $K+i\sigma C-\sigma^2 M$ as in the solution of the quadratic problem.

The Arnoldi method does not produce Ritz vectors that satisfy (2.5) exactly, so the relationship between Algorithms 2.2 and 2.3 is lost. On the other hand, there is a link with Algorithm 2.1, since in exact arithmetic and with the exact linear solves, the Davidson method produces the same subspace as the shift-invert Arnoldi method when σ is fixed. The major difference between Algorithms 2.1 and Algorithms 2.3 lies in the construction of the subspaces. In the Algorithms 2.2 and 2.3, the subspace is expanded so that the two components from the Cayley transform applied to a Ritz vector are added. Both algorithms are designed to improve one Ritz vector at a time. In the Arnoldi method, all Ritz vectors converge together.

This is illustrated by the following example. Consider the example from §8.3. We performed 30 steps of Arnoldi's method applied to $(A-\sigma B)^{-1}B$ starting with a random initial vector. The left-hand picture in Figure 10.1 shows the residual norms $\rho_j = \|Ax_j - BX_jS_j\|$ of the six Ritz values nearest $\sigma = 300$ as a function of the iteration number. The Ritz values nearest σ converge faster, where $X_j = [x_1, \ldots, x_j]$ denote the first j Schur vectors and S_j the corresponding Schur matrix. Most of the Ritz values have decreasing residuals from the first to the last iteration. The central picture shows the results of 30 iterations of quadratic residual iteration. We used Algorithm 8.1 without a restart. We consider a Ritz value as converged when the residual norm $\rho_j = \|r_P\| = \|(KU_j + iCU_jS_j - MU_jS_j^2)e_j\|$ is smaller than the convergence tolerance, 10^{-7}. The horizontal dashed line indicates the tolerance TOL used by Algorithm 8.1 in Step 2.3.6. Two different kinds of convergence behaviour can be observed. The residual norm of the third Ritz value (dotted line) makes a significant decrease during the convergence of the first and the second. This looks like the convergence behaviour of the Arnoldi method. The other Ritz values show a completely different behaviour. Each time a Ritz value has converged it is locked and another Ritz value is targeted and starts converging. This is very clear from the dashed convergence curves. The residuals decrease at the beginning, but most of them stagnate until a new eigenpair is targeted. Peaks in the curves indicate a Ritz value that starts converging to another eigenvalue.

The tolerance TOL plays an important role in the overall convergence speed. In the example mentioned above, Arnoldi requires 21 iterations for finding six eigenvalues with the required accuracy when TOL$= 10^{-7}$ and quadratic residual iteration 28 iterations. When TOL$= 10^{-4}$ instead, Arnoldi's method requires only 18 iterations, but quadratic residual no more than 14. This illustrates that we cannot make a decision on which method is the best.

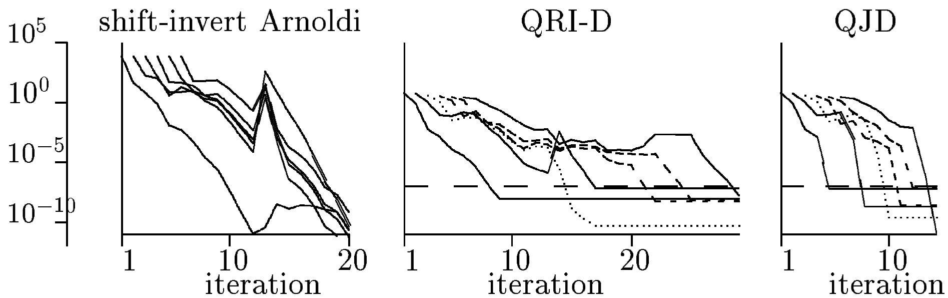

Figure 10.1: Convergence behaviour of shift-invert Arnoldi, quadratic residual iteration (QRI-D) and Jacobi-Davidson (QJD) for the problem from §8.3. The lines show the evolution of $\rho_j = \|(KU_j + iCU_jS_j - MU_jS_j^2)e_j\|$ of the six Ritz values nearest σ.

The right-hand picture shows the results for the Jacobi-Davidson method preconditioned with $K + i\sigma C - \sigma^2 M$ with $\sigma = 300$ and linear solver tolerance $\tau = 10^{-5}$. The first three iterations used a fixed pole $\sigma = 300$, and then $\sigma = \omega$, while the preconditioner is computed once for $\sigma = 300$. The picture shows quadratic convergence for the Ritz values. The algorithm converges in 13 iterations, but requires 118 complex linear solves, while less than 30 for the other two algorithms. Stagnation is not as pronounced as for the QRI-D method.

The quadratic residual iteration method has some advantages. First, the modified Davidson framework (Algorithm 2.2) uses a larger subspace than the Davidson approach (Algorithm 2.1) constructed by adding two vectors per iteration step to the subspace. This may lead to some minor gain in convergence speed. The potential gain is in the storage of the iteration vectors. Often, however, one stores the matrices KV_k, MV_k and CV_k in order to calculate the residuals more efficiently or the projection matrices K_k, M_k and C_k and then the advantage is lost. In our implementation, these additional vectors are not stored.

11 Conclusions

In this paper, we have shown there is a close connection between solution methods for the quadratic eigenvalue problem and its linearized form. Solution methods that solve the quadratic problem without linearization appear to be equivalent to methods that solve the linearized problem by projection on a larger subspace. The Jacobi-Davidson correction equation for the linearized problem is connected to the correction equation of the quadratic problem.

If direct linear system solvers are used for the Cayley transform, the gain of the residual iteration method is small compared to the shift-invert Arnoldi method. The Gram-Schmidt orthogonalization cost is significantly lower due to the smaller dimension. When the vectors KV_k, CV_k and MV_k are not stored, about half of the memory is needed, but otherwise the memory consumption is higher than for the shift-invert Arnoldi method. The convergence of the Arnoldi method is not focused on a single eigenvalue, but all eigenvalues start converging from in the beginning. The quadratic method targets one eigenvalue, which may lead to a fast local convergence, but a slow global convergence. Therefore, we suggest the use of the shift-invert Arnoldi method when direct linear system solvers are used. However, as the last example illustrates, when the convergence tolerance is large, quadratic residual iteration is faster. If iterative linear system solvers are used, we suggest the use of quadratic residual iteration or the Jacobi-Davidson method for the quadratic problem in conjunction with a deflation and restarting scheme based on the linearized case. It is important to note that none of the discussed methods can be flagged as optimal : there are always situations for which the one method outperforms the other.

Finally, we want to stress that a practical code should use a block version, i.e. a number of Cayley transforms is applied to more than one Ritz vector at a time. This allows us to compute multiple or clustered eigenvalues more efficiently and may improve the reliability of the method since more than one eigenvalue is targeted simultaneously. When Jacobi-Davidson is used some care with the choice of poles σ is in order when multiple eigenvalues are expected, since this may lead to the solution of nearly singular linear systems.

12 Acknowledgements

The author thanks Jennifer Scott, Nick Gould and Iain Duff from the Rutherford Appleton Laboratory for helpful comments that improved the presentation of the paper. We are also grateful Jean-Pierre Coyette from Free Field Technologies for the matrices of the acoustic simulation of a poro-elastic material. The author also thanks the referee for suggestions that improved the comparisons between the various methods.

References

Crouzeix, M., Philippe, B. and Sadkane, M. (1994), 'The Davidson method', *SIAM J. Sci. Comput.* **15**, 62–76.

Daniel, J., Gragg, W., Kaufman, L. and Stewart, G. (1976), 'Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization', *Math. Comp.* **30**, 772–795.

De Samblanx, G., Meerbergen, K. and Bultheel, A. (1997), 'The implicit application of a rational filter in the RKS method', *BIT* **37**, 925–947.

Ericsson, T. and Ruhe, A. (1980), 'The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems', *Math. Comp.* **35**, 1251–1268.

Fokkema, D., Sleijpen, G. and van der Vorst, H. (1999), 'Jacobi-Davidson style QR and QZ algorithms for the reduction of matrix pencils', *SIAM J. Sci. Comput.* **20**(1), pp.94–125.

Golub, G. and Van Loan, C. (1996), *Matrix computations*, 3rd edn, The Johns Hopkins University Press.

Grimes, R., Lewis, J. and Simon, H. (1994), 'A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems', *SIAM J. Matrix Anal. Applic.* **15**, 228–272.

HSL (1996), 'Harwell Subroutine Library. A Catalogue of Subroutines (Release 12)'. Information: Nick Brealey, AEA Technology, 477 Harwell, Didcot, Oxon, OX11 0RA, UK, E-mail: hsl@aeat.co.uk.

Huitfeldt, J. and Ruhe, A. (1990), 'A new algorithm for numerical path following applied to an example from hydrodynamical flow', *SIAM J. Sci. Statist. Comput.* **11**(6), 1181–1192.

Lehoucq, R. and Meerbergen, K. (1998), 'Using generalized Cayley transformations within an inexact rational Krylov sequence method', *SIAM J. Matrix Anal. Applic.* **20**(1), 131–148.

Lehoucq, R. and Sorensen, D. (1996), 'Deflation techniques within an implicitly restarted Arnoldi iteration', *SIAM J. Matrix Anal. Applic.* **17**, 789–821.

Meerbergen, K. (1996), Robust methods for the calculation of rightmost eigenvalues of large eigenvalue problems, PhD thesis, Department of Computer Science, K.U.Leuven, Leuven, Belgium.

Meerbergen, K. and Roose, D. (1996), 'Matrix transformations for computing rightmost eigenvalues of real nonsymmetric matrices', *IMA J. Numer. Anal.* **16**, 297–346.

Meerbergen, K. and Roose, D. (1997), 'The restarted Arnoldi method applied to iterative linear system solvers for the computation of rightmost eigenvalues', *SIAM J. Matrix Anal. Applic.* **18**, 1–20.

Morgan, R. (1992), 'Generalisations of Davidson's method for computing eigenvalues of large nonsymmetric matrices', *J. Comput. Phys.* **101**, 287–291.

Morgan, R. (1996), 'On restarting the Arnoldi method for large nonsymmetric eigenvalue problems', *Math. Comp.* **65**, 1213–1230.

Morgan, R. and Meerbergen, K. (1999), §11.2. Inexact methods, *in* Z. Bai, J. Demmel, J. Dongarra, A. Ruhe and H. van der Vorst, eds, 'Templates for the solution of algebraic eigenvalue problems: a practical guide', SIAM, Philadelphia, USA. In press.

Natarajan, R. (1992), 'An Arnoldi-based iterative scheme for nonsymmetric matrix pencils arising in finite element stability problems', *J. Comput. Phys.* **100**, 128–142.

Neumaier, A. (1998), 'Residual inverse iteration for the nonlinear eigenvalue problem', *SIAM J. Numer. Anal.* **22**, 914–923.

Ruhe, A. (1998), 'Rational Krylov: a practical algorithm for large sparse nonsymmetric matrix pencils', *SIAM J. Sci. Comput.* **19**(5), 1535–1551.

Saad, Y. (1992), *Numerical methods for large eigenvalue problems*, Algorithms and Architectures for Advanced Scientific Computing, Manchester University Press, Manchester, UK.

Saad, Y. and Schultz, M. (1986), 'GMRES : a generalized minimal residual algorithm for solving nonsymmetric linear systems', *SIAM J. Sci. Statist. Comput.* **7**, 856–869.

Sandhu, R. and Pister, K. (1970), 'A variational principle for linear coupled field problems in continuum mechanics', *Int. J. Eng. Sci.* **8**, pp. 989–999.

Scott, J. (1995), 'An Arnoldi code for computing selected eigenvalues of sparse unsymmetric matrices', *ACM Trans. Math. Softw.* **21**, 432–475.

Simon, B., Wu, J., Zienkiewicz, O. and Paul, D. (1986), 'Evaluation of u-w and u-p finite element methods for the dynamic response of saturated porous media using one-dimensional models', *Int. J. Numer. Anal. Methods Geomech.* **10**, pp. 461–482.

Sleijpen, G. and van der Vorst, H. (1996), 'A Jacobi-Davidson iteration method for linear eigenvalue problems', *SIAM J. Matrix Anal. Applic.* **17**, 401–425.

Sleijpen, G., Booten, G., Fokkema, D. and van der Vorst, H. (1996*a*), 'Jacobi Davidson type methods for generalized eigenproblems and polynomial eigenproblems', *BIT* **36**, 595–633.

Sleijpen, G., van der Vorst, H. and van Gijzen, M. (1996*b*), 'Quadratic eigenproblems are no problem', *SIAM News* **29**(7), pp.8–9.

Sorensen, D. (1992), 'Implicit application of polynomial filters in a k-step Arnoldi method', *SIAM J. Matrix Anal. Applic.* **13**, 357–385.

Stewart, G. (1976), 'Simultaneous iteration for computing invariant subspaces of non-Hermitian matrices', *Numer. Math.* **25**, 123–136.

Tisseur, F. (1998), Backward error and condition of polynomial eigenvalue problems, NA Report 332, Revised, 1999, Department of Mathematics, University of Manchester, M13 9PL, UK. To appear in Linear Algebra and its Applications.

van Gijzen, M. and Raeven, F. (1995), The parallel computation of the smallest eigenpair of an acoustic problem with damping, Preprint 924, Utrecht University, Department of Mathematics, P.O.Box 80.010, 3508 TA Utrecht, The Netherlands.