Chapter 8

Computing a partial generalized real Schur form using the Jacobi-Davidson method

Abstract. In this chapter, a new variant of the Jacobi-Davidson method is presented that is specifically designed for *real unsymmetric* matrix pencils. Whenever a pencil has a complex conjugate pair of eigenvalues, the method computes the two dimensional real invariant subspace spanned by the two corresponding complex conjugate eigenvectors. This is beneficial for memory costs and in many cases it also accelerates the convergence of the JD method. Both real and complex formulations of the correction equation are considered. In numerical experiments, the RJDQZ variant is compared with the original JDQZ method.

Key words. partial Schur form, real Schur form, ordered Schur form, Jacobi-Davidson, large scale eigenvalue problem

8.1 Introduction

Real unsymmetric matrices or real unsymmetric matrix pencils may have complex eigenvalues and corresponding eigenvectors. Therefore, the (partial generalized) Schur form may consist of complex matrices. In some situations, (e.g., in a continuation context [22]) it is more desirable to compute a real (partial generalized) Schur form. This decomposition consists for a matrix of an orthogonal real matrix and block upper triangular matrix, which has scalars or two by two blocks on the diagonal. The eigenvalues of such a two by two block correspond to two complex conjugate eigenvalues of the matrix (pencil) itself. Advantages of the real Schur form are that it requires less storage, since for every complex conjugate pair of eigenvalues only two real vectors need to be stored instead of two complex vectors,

and that complex conjugate pairs of eigenvalues always appear together.

In this chapter, a variant of the JDQZ method [51] is considered for the computation of a partial generalized real Schur form of a matrix pencil. The original JDQZ method [51] does not use the fact that the pencil is real: (1) it does not exploit the fact that eigenvalues are real or appear in complex conjugate pairs and (2) it needs complex arithmetic, even when only real eigenvalues appear. This is in contrast with other iterative eigenvalue solvers such as the Arnoldi method.

Algorithm 4.1 proposed in [25] computes a partial Schur form in a naive way. This algorithm consists of an outer iteration in which the partial Schur form is expanded by a scalar block whenever the inner iteration, which may consist of the Jacobi-Davidson method applied to a deflated matrix, returns a real eigenvalue, and with a two by two block if the inner iteration returns an eigenvalue with nonzero imaginary part. Thus the Algorithm 4.1 in [25] solves problem (1), but it does not solve problem (2): the inner iteration does not use the fact that the matrix is real.

The variant of the Jacobi-Davidson method, called RJD, that is presented in this chapter, takes advantage of the fact that the pencil is real: it only needs real arithmetic to compute real eigenvalues and computes complex conjugate pairs of eigenvalues by approximating the corresponding two dimensional invariant subspace, by keeping the search and test spaces real. The goal of this chapter is to show that RJD solves problems (1) and (2), and, furthermore, that it converges faster and is more efficient than standard JD, most likely due to the larger search space.

In [26], it is stated that a real implementation of the JD method is used for a comparison with the Riccati method, but implementation details and a comparison with the original JD method are not presented. Also in [51] (Remark 1), the authors hint at a real version of the JD method, but do not pursue this matter further.

The structure of this chapter is as follows. In Section 2, the JDQZ method [51] is discussed. The RJDQZ method is presented in Section 3, and in Section 4, different ways to solve the correction equation in the RJDQZ method are discussed and compared. A comparison of the computational and memory costs between the original JDQZ method and the RJDQZ method is found in Section 5, and a numerical comparison in Section 6. Section 7 concludes.

8.2 A Jacobi-Davidson style QZ method

In this section the JDQZ method [51] for the construction of a partial generalized Schur form of an unsymmetric matrix pencil is discussed.

8.2.1 The Jacobi-Davidson Method for the Generalized Eigenvalue Problem

The Jacobi-Davidson (JD) method [140] iteratively computes approximations to eigenvalues, and their corresponding eigenvectors, that are close to some specified

target τ, of the generalized unsymmetric eigenvalue problem

$$A\mathbf{x} = \lambda B\mathbf{x}, \tag{8.2.1}$$

where A and B are in general unsymmetric $n \times n$ matrices. In each iteration, a search subspace colspan(V) and a test subspace colspan(W) are constructed. V and W are complex $n \times j$ matrices with $j \ll n$ and $V^*V = W^*W = I$. In the first part of an iteration, an approximation to an eigenvector of the generalized eigenvalues problem (8.2.1) is obtained from the projected eigenvalue problem

$$W^*AV\mathbf{u} = \lambda W^*BV\mathbf{u}. \tag{8.2.2}$$

This is a small eigenvalue problem of size $j \times j$, so that a full space method like the QZ method can be used to compute all the eigenvalues and eigenvectors.

Suppose $(\tilde{\lambda}, \mathbf{u})$ is the eigenpair of the projected eigenvalue problem (8.2.2), of which the eigenvalue $\tilde{\lambda}$ is closest to τ. An approximation $(\tilde{\lambda}, \tilde{\mathbf{x}})$ to an eigenpair of the full sized eigenvalue problem (8.2.1) can be constructed by computing $\tilde{\mathbf{x}} = V\mathbf{u}$. The residual vector $\mathbf{r}$ of the approximate eigenpair $(\tilde{\lambda}, \tilde{\mathbf{x}})$ is defined by

$$\mathbf{r} := A\tilde{\mathbf{x}} - \tilde{\lambda}B\tilde{\mathbf{x}}.$$

The second part in a JD iteration is the expansion of the search and test space. The search space V is expanded by an approximate solution $\mathbf{x}$ of the linear equation

$$(I - \tilde{\mathbf{z}}\tilde{\mathbf{z}}^*)(A - \tilde{\lambda}B)(I - \tilde{\mathbf{x}}\tilde{\mathbf{x}}^*)\mathbf{t} = -\mathbf{r}. \tag{8.2.3}$$

This equation is called the Jacobi-Davidson correction equation. Here $\tilde{\mathbf{z}}$ is the vector $\tilde{\mathbf{z}} = (\kappa_0 A + \kappa_1 B)\tilde{\mathbf{x}}$. The test space W is expanded with the vector $\mathbf{w} = (\kappa_0 A + \kappa_1 B)\mathbf{t}$. This procedure is repeated until $||\mathbf{r}||$ is small enough.

There are several possible choices for the complex numbers κ_0 and κ_1. In [51] it is shown that an effective choice for interior eigenvalues is $\kappa_0 = (1 + |\tau|^2)^{-1/2}$ and $\kappa_1 = -\tau(1 + |\tau|^2)^{-1/2}$. This corresponds to the harmonic Petrov value approach, see [51] for more choices.

8.2.2 The JDQZ Method

The JDQZ method is a Jacobi-Davidson style method that is designed to compute an approximation to a partial generalized Schur form of the matrix pair (A, B)

$$AQ_k = Z_k S_k, \quad BQ_k = Z_k T_k, \tag{8.2.4}$$

where Q_k and Z_k are $n \times k$ matrices with orthonormal columns, and S_k and T_k are $k \times k$ upper triangular matrices. Eigenvalues of the pair (S_k, T_k) are also eigenvalues of the pair (A, B).

The first column of Q_k is an eigenvector of the pair (A, B), and can thus be computed with the JD method. Suppose that a partial Schur form (8.2.4) is computed

already. It can be shown [51] that the next Schur vector $\mathbf{q}_{k+1}$ and corresponding eigenvalue λ_{k+1} satisfy $Q_k^*\tilde{\mathbf{q}}_{k+1} = 0$ and

$$(I - Z_k Z_k^*)(A - \lambda_{k+1}B)(I - Q_k Q_k^*)\mathbf{q}_{k+1} = 0. \tag{8.2.5}$$

This deflated generalized eigenvalue problem can be solved with the JD method. The projected generalized eigenvalue problem that has to be solved can be written as

$$W^*(I - Z_k Z_k^*)A(I - Q_k Q_k^*)V\mathbf{u} = \lambda W^*(I - Z_k Z_k^*)B(I - Q_k Q_k^*)V\mathbf{u}, \tag{8.2.6}$$

with $V^*Q_k = W^*Z_k = 0$, where V and W are again search and test spaces, respectively.

Let $(\tilde{\lambda}, \tilde{\mathbf{u}})$ denote an eigenpair of the projected eigenvalue problem (8.2.6). Again an approximation $(\tilde{\lambda}, \tilde{\mathbf{q}})$ to the eigenpair $(\lambda_{k+1}, \mathbf{q}_{k+1})$ of the eigenvalue problem (8.2.5) can be constructed by computing $\tilde{\mathbf{q}} = V\tilde{\mathbf{u}}$. In order to expand the search space V, an approximate solution $\mathbf{t}$ of the correction equation

$$(I - \tilde{\mathbf{z}}\tilde{\mathbf{z}}^*)(I - Z_k Z_k^*)(A - \tilde{\lambda}B)(I - Q_k Q_k^*)(I - \tilde{\mathbf{q}}\tilde{\mathbf{q}}^*)\mathbf{t} = -\mathbf{r}, \tag{8.2.7}$$

is computed, where $\tilde{\mathbf{z}} = (\kappa_0 A + \kappa_1 B)\tilde{\mathbf{q}}$, and $\mathbf{r} = (I - Z_k Z_k^*)(A - \tilde{\lambda}B)(I - Q_k Q_k^*)\tilde{\mathbf{q}}$. The test space W is expanded with the vector $w = (\kappa_0 A + \kappa_1 B)\mathbf{t}$.

Observe that complex arithmetic needs to be used to compute approximations to solutions of equation (8.2.7) whenever $\tilde{\lambda}$ has a nonzero imaginary part, or whenever the matrices Q_k and Z_k contain entries with nonzero imaginary part.

When the JD correction equation (8.2.7) is solved exactly in each step of the JDQZ method, then it can be shown that the method converges asymptotically quadratically to an eigenvector. Solving the correction equation exactly, however, can be expensive. It may be better for the overall performance of the JDQZ method to solve the correction equation (8.2.7) only approximately. See [51] for an effective stopping criterion and also for the use of a preconditioner.

If $B = I$ the generalized eigenvalue problem (8.2.1) reduces to the standard eigenvalue problem $A\mathbf{x} = \lambda\mathbf{x}$, that can be solved by the JDQR method [51], a simplification of the JDQZ method. For the standard eigenvalue problem the eigenvalues of the projected eigenvalue problem (8.2.6) are called (harmonic) Ritz values, instead of (harmonic) Petrov values [51].

8.3 A new JDQZ Method for Real Matrix Pencils

In this section a Jacobi-Davidson style method that is specifically designed for *real unsymmetric* matrix pencils, is discussed.

A partial generalized real Schur form of the real matrix pencil (A, B) is a decomposition of the following form

$$AQ_k = Z_k S_k, \quad BQ_k = Z_k T_k,$$

where now Q_k and Z_k are *real* matrices with orthonormal columns, and S_k and T_k are *real* block upper triangular matrices with scalar or two by two diagonal blocks. The eigenvalues of the two by two diagonal blocks correspond to complex conjugate pairs of eigenvalues of the pencil (A, B).

8.3.1 The JDQRd and JDQZd Algorithms

In [25] an adaptation of the JDQR algorithm for the standard eigenvalue problem is proposed (in Algorithm 4.1) to compute a partial real Schur form. This algorithm is referred to as JDQRd in this chapter.

In order to explain the difference between the JDQR and the JDQRd method, it is convenient to look at the JDQR/JDQZ method as a 3-nested loop. The outer loop (level 1) accumulates the Schur vectors and in this loop the matrices Q_k, Z_k, S_k and T_k, which are needed for the partial Schur form (8.2.4), are constructed. The columns of Q_k and Z_k are computed as left and right eigenvectors of the deflated generalized eigenvalue problem (8.2.5). This problem is solved using the JD method, of which the iterations constitute the level 2-loop. The innermost loop (level 3) is the iterative solution of the correction equation (8.2.7).

The original JDQR/JDQZ methods use complex arithmetic in all three of the levels. The difference between the JDQR and JDQRd method is that the JDQRd method constructs a real partial Schur form by accumulating real vectors in the outermost loop. Thus level 1 in the JDQRd method uses real arithmetic, while levels 2 and 3 use complex arithmetic. This means that the JDQRd algorithm proceeds just as the JDQR method, except that when a complex eigenvalue λ_k, with corresponding eigenvector $\mathbf{q}_k$, is computed, then the partial Schur form is kept real by augmenting it with a real basis of the space spanned by $\mathbf{q}_k$ and $\bar{\mathbf{q}}_k$. In JDQRd, like in JDQR, the JD search space V in level 2 does not have to be real, and, therefore, the Ritz values do not have to appear in complex conjugate pairs. This causes difficulties for the identification of complex pairs of eigenvalues of the original eigenvalue problem, see e.g. Chapter 8 in [132], and it also introduces additional rounding errors when an computed approximate eigenvalue with small imaginary part is replaced by its real part. In Section 8.6.3, also the JDQZd method is considered, which is the QZ version of the JDQRd method.

In the next section, the RJDQZ method is introduced. This method uses real arithmetic in in both levels 1 and 2 and uses real arithmetic in level 3 when possible.

8.3.2 The RJDQZ Algorithm

A more thorough and robust way to compute a partial generalized real Schur form is to keep the JD search and test space in level 2 real. Suppose one has already a real search space V and a real test space W. Then one has to compute the eigenvalues of the projected generalized eigenvalue problem (8.2.2):

$$W^T A V \mathbf{u} = \lambda W^T B V \mathbf{u}.$$

Since the projected matrices W^TAV and W^TBV are real, the eigenvalues are either real or form a complex conjugate pair, and since the projected eigenvalue problem is small, all eigenvalues can be computed accurately and cheaply. From these eigenvalues one eigenvalue (or complex conjugate pair) is selected with a given selection criterion. Denote the selected Petrov value by $\tilde{\lambda}$ and the corresponding eigenvector by $\tilde{\mathbf{u}}$.

In the actual RJDQZ algorithm, instead of an eigenvalue decomposition of the projected eigenvalue problem (8.2.2), a sorted real generalized Schur form is computed. How this form is computed in an efficient and stable way can be found in [9, 23, 25] for the standard eigenvalue problem and in [76, 77, 41] for the generalized eigenvalue problem. As mentioned above, it is in the construction of the sorted real generalized Schur form where it is decided (up to machine precision) whether an eigenvalue is real or appears in a complex conjugate pair.

For a Petrov pair $(\tilde{\lambda}, \tilde{\mathbf{q}} = V\tilde{\mathbf{u}})$, the JDQZ correction equation (8.2.7) is of the following form:

$$(I - ZZ^*)(A - \theta B)(I - QQ^*)\mathbf{t} = -\mathbf{r}, \tag{8.3.1}$$

with $Q = [Q_k, \tilde{\mathbf{q}}]$ and $Z = [Z_k, \tilde{\mathbf{z}}]$. If the selected Petrov value $\tilde{\lambda}$ is real then the matrix and the right hand side in the correction equation (8.3.1) are both real, and an approximate solution can be computed using real arithmetic. Hence the search and test space can be expanded with a real vector.

If the selected Petrov value λ has nonzero imaginary part, then the JD correction equation will have to be solved using complex arithmetic, and one will obtain a complex approximate solution $\mathbf{t}$. In order to keep the search space real, it is expanded with the two dimensional real space $U = \text{span}\{\text{Re}(\mathbf{t}), \text{Im}(\mathbf{t})\}$, which contains the vector $\mathbf{t}$.

Remark 1: It is irrelevant whether a Petrov value $\tilde{\lambda}$ or $\tilde{\lambda}^*$ is selected, since in both cases the search space is expanded with the same space U. This also solves a problem from which the original JD method suffers when the target value is real (see Section 8.6.2 for more details).

Remark 2: Note that if the target value τ has a nonzero imaginary part then κ_1 (see discussion after equation (8.2.3)) will not be real in the harmonic Petrov approach. Thus in the case of a non-real target value combined with the harmonic Petrov approach, the proposed method loses most of its advantages, although keeping the search space real by expanding it with a two dimensional real space, when appropriate, might still accelerate the convergence of the JD method. Note that in this case also other iterative eigenvalue solvers such as the shift and invert Arnoldi method will need complex arithmetic [113].

Considering a complex target for the RJDQZ method may seem to contradict the goal of the method, but the method only aims to exploit the realness of the pencil. The test pencil MHD416 (see Section 8.6.4) is an example of a real pencil for which a complex target is wanted.

8.4 Formulating and solving the correction equation

If a real Petrov pair is selected, the correction equation can be solved using real arithmetic. If a complex Petrov pair is selected, there are three ways to formulate the correction equation for the real variant of Jacobi-Davidson QZ: (1) the correction equation can be formulated as a complex equation (the usual way), (2) the complex correction equation can be made real by defining two coupled equations for the real and imaginary part, or (3) a generalized real Sylvester equation can be formulated for the correction of the approximate two dimensional invariant subspace. In the following it will be shown that these three formulations are equivalent and that approach (3) is the preferred approach from a conceptual point of view, while approach (1) is more efficient in practice.

8.4.1 Complex correction equation

This is the usual formulation of the correction equation as in (8.3.1), solved using complex arithmetic. To keep the JD search space V real, it is expanded with the two dimensional real space $\mathrm{span}(\mathrm{Re}(\mathbf{t}), \mathrm{Im}(\mathbf{t}))$.

8.4.2 Real variant of complex correction equation

Let $\theta = \nu + i\omega$, $\mathbf{t} = \mathbf{u} + i\mathbf{v}$ and $\mathbf{r} = \mathbf{r}_1 + i\mathbf{r}_2$. Then it follows that

$$(A - \theta B)\mathbf{t} = -\mathbf{r} \Longleftrightarrow \begin{bmatrix} A - \nu B & \omega B \\ -\omega B & A - \nu B \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = - \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix}.$$

The matrices with already converged right and left Schur vectors Q_k and Z_k are real. Let $\mathbf{q} = \mathbf{q}_1 + i\mathbf{q}_2$ with $\mathbf{q}_1^*\mathbf{q}_1 + \mathbf{q}_2^*\mathbf{q}_2 = 1$. Then

$$\begin{aligned} (I - \mathbf{q}\mathbf{q}^*)t &= (I - (\mathbf{q}_1\mathbf{q}_1^* + \mathbf{q}_2\mathbf{q}_2^*))\mathbf{u} + (\mathbf{q}_2\mathbf{q}_1^* - \mathbf{q}_1\mathbf{q}_2^*)\mathbf{v} \\ &+ i((I - (\mathbf{q}_1\mathbf{q}_1^* + \mathbf{q}_2\mathbf{q}_2^*))\mathbf{v} - (\mathbf{q}_2\mathbf{q}_1^* - \mathbf{q}_1\mathbf{q}_2^*)\mathbf{u}). \end{aligned}$$

The equivalent real block formulation becomes

$$(I - \mathbf{q}\mathbf{q}^*)\mathbf{t} \Longleftrightarrow P_q \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \equiv \begin{bmatrix} I - (\mathbf{q}_1\mathbf{q}_1^* + \mathbf{q}_2\mathbf{q}_2^*) & \mathbf{q}_2\mathbf{q}_1^* - \mathbf{q}_1\mathbf{q}_2^* \\ -(\mathbf{q}_2\mathbf{q}_1^* - \mathbf{q}_1\mathbf{q}_2^*) & I - (\mathbf{q}_1\mathbf{q}_1^* + \mathbf{q}_2\mathbf{q}_2^*) \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}.$$

By using $\mathbf{q}_1^*\mathbf{q}_1 + \mathbf{q}_2^*\mathbf{q}_2 = 1$ and some basic linear algebra, it can be shown that P_q is indeed a projector. In a similar way, P_z for $\mathbf{z} = \mathbf{z}_1 + i\mathbf{z}_2$ and $\mathbf{z}_1^*\mathbf{z}_1 + \mathbf{z}_2^*\mathbf{z}_2 = 1$ can be defined as

$$P_z \equiv \begin{bmatrix} I - (\mathbf{z}_1\mathbf{z}_1^* + \mathbf{z}_2\mathbf{z}_2^*) & \mathbf{z}_2\mathbf{z}_1^* - \mathbf{z}_1\mathbf{z}_2^* \\ -(\mathbf{z}_2\mathbf{z}_1^* - \mathbf{z}_1\mathbf{z}_2^*) & I - (\mathbf{z}_1\mathbf{z}_1^* + \mathbf{z}_2\mathbf{z}_2^*) \end{bmatrix}.$$

The real equivalent of the correction equation (8.3.1) becomes

$$P_z\mathcal{Z}_k \begin{bmatrix} A - \nu B & \omega B \\ -\omega B & A - \nu B \end{bmatrix} \mathcal{Q}_k P_q \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = - \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix}, \tag{8.4.1}$$

where

$$\mathcal{Z}_k = \begin{bmatrix} I - Z_k Z_k^* & 0 \\ 0 & I - Z_k Z_k^* \end{bmatrix} \quad \text{and} \quad \mathcal{Q}_k = \begin{bmatrix} I - Q_k Q_k^* & 0 \\ 0 & I - Q_k Q_k^* \end{bmatrix}.$$

8.4.3 Real generalized Sylvester equation

An advantage of the RJDQZ algorithm is that approximations to complex conjugate pairs appear in conjugate pairs. The corresponding residual for the approximate two dimensional invariant subspace $\begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 \end{bmatrix}$ is

$$\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 \end{bmatrix} = A \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 \end{bmatrix} - B \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 \end{bmatrix} \begin{bmatrix} \nu & \omega \\ -\omega & \nu \end{bmatrix}.$$

The correction equation for $\begin{bmatrix} \mathbf{u} & \mathbf{v} \end{bmatrix}$ becomes a real generalized Sylvester equation

$$(I - ZZ^*)(A(I - QQ^*) \begin{bmatrix} \mathbf{u} & \mathbf{v} \end{bmatrix} - B(I - QQ^*) \begin{bmatrix} \mathbf{u} & \mathbf{v} \end{bmatrix} \begin{bmatrix} \nu & \omega \\ -\omega & \nu \end{bmatrix}) = - \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 \end{bmatrix}$$

The equivalent block formulation becomes

$$P_z \mathcal{Z}_k \begin{bmatrix} A - \nu B & \omega B \\ -\omega B & A - \nu B \end{bmatrix} \mathcal{Q}_k P_q \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = - \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix}, \tag{8.4.2}$$

which is the same as the one obtained in (8.4.1).

An alternative formulation of the correction equation can be obtained if one considers a sorted real generalized Schur form instead of an eigenvalue decomposition (see also [51]). The selected approximate Schur quartet for the deflated problem is $(\begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 \end{bmatrix}, S, T)$, with $S, T \in \mathbb{R}^{2\times 2}$, T upper triangular, $\begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 \end{bmatrix} \perp Q_k$ and $\begin{bmatrix} \mathbf{z}_1 & \mathbf{z}_2 \end{bmatrix} \perp Z_k$. The residual is computed as

$$\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 \end{bmatrix} = (I - Z_k Z_k^*)(A \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 \end{bmatrix} S^{-1} - B \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 \end{bmatrix} T^{-1}),$$

and the correction equation becomes

$$P_z \mathcal{Z}_k \begin{bmatrix} s_{11}^{-1} A - t_{11}^{-1} B & s_{21}^{-1} A \\ s_{12}^{-1} A - t_{12}^{-1} B & s_{22}^{-1} A - t_{22}^{-1} B \end{bmatrix} \mathcal{Q}_k P_q \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = - \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \end{bmatrix}. \tag{8.4.3}$$

8.4.4 Approximate solution of the correction equation

The real and imaginary part of the exact solution of (8.3.1) and the exact solutions of (8.4.1) and (8.4.2), and (8.4.3) span the same two dimensional real subspace. In practice however, the correction equation is only solved approximately using an iterative linear solver like GMRES. The rate of convergence of linear solvers depends, among others, on the condition number of the operator, the distribution of the eigenvalues, and the quality of the preconditioner. The following proposition states that the eigenvalues of the complex matrix $A - \tilde{\lambda} B$ in equation (8.3.1) are also eigenvalues of the equivalent real block matrix in equations (8.4.1) and (8.4.2), together with their complex conjugates and furthermore that the condition numbers of the matrices are the same.

Proposition 8.4.1. *Let $A, B \in \mathbb{R}^{n\times n}$, $\tilde{\lambda} = \nu + i\omega \in \mathbb{C}$ and $(A-\theta B)\mathbf{x}_j = \lambda_j \mathbf{x}_j, j = 1,\ldots,n$ with $\mathbf{x}_j \in \mathbb{C}^n$ and $\lambda_j \in \mathbb{C}$. Then the eigenpairs of*

$$C = \begin{bmatrix} A-\nu B & \omega B \\ -\omega B & A-\nu B \end{bmatrix} \in \mathbb{R}^{2n\times 2n} \tag{8.4.4}$$

are $(\lambda_j, \left[\mathbf{x}_j^T, (-i\mathbf{x}_j)^T\right]^T)$, $(\bar{\lambda_j}, \left[\mathbf{x}_j^T, (-i\mathbf{x}_j)^T\right]^*)$ for $j = 1,\ldots,n$.
Furthermore Cond(C)=Cond(A − θB).

Proof. For an eigenpair $(\lambda, \mathbf{x})$ of $A - \theta B$ it holds that

$$(A-\nu B)\mathbf{x} - \omega B i\mathbf{x} = \lambda\mathbf{x}, \qquad -(A-\nu B)i\mathbf{x} - \omega B\mathbf{x} = -\lambda i\mathbf{x}.$$

Using this it easily follows that

$$\begin{bmatrix} A-\nu B & \omega B \\ -\omega B & A-\nu B \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ -i\mathbf{x} \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{x} \\ -i\mathbf{x} \end{bmatrix}.$$

The first part of the proposition follows by noting that because matrix (8.4.4) is a real matrix, its eigenpairs appear in complex conjugate pairs. The equality of the condition numbers for C and $A - \theta B$ follows from the fact that for every $\mathbf{z} = \mathbf{v} + i\mathbf{w} \in \mathbb{C}^n$ it holds that

$$\frac{\mathbf{z}^*(A-\theta B)^*(A-\theta B)\mathbf{z}}{\mathbf{z}^*\mathbf{z}} = \frac{\boldsymbol{v}^T C^T C \boldsymbol{v}}{\boldsymbol{v}^T\boldsymbol{v}},$$

where $\boldsymbol{v} = (\mathbf{v}^T\ \mathbf{w}^T)^T$. □

Using similar arguments, this relation can be extended to the operator in equation (8.4.3). From proposition 8.4.1 it follows that no big differences in convergence are to be expected if the approximate solution is computed with a linear solver. This is also confirmed by numerical experiments.

If a preconditioner $K \approx A - \tau B$ is available for a target $\tau \in \mathbb{R}$, it can be used for the block systems as well:

$$\tilde{K} = \begin{bmatrix} K & 0 \\ 0 & K \end{bmatrix}.$$

Using proposition 8.4.1, the condition numbers of $K^{-1}(A-\theta B)$ and $\tilde{K}^{-1}C$ are the same. So the use of a preconditioner also is not expected to cause big differences in speed convergence between the three approaches.

Numerical experiments showed that solving the complex correction equation was the most efficient with respect to the convergence of both the Krylov solver and the JD algorithm. Part of the explanation of why this is the case can be obtained by looking at the computational complexity of the different versions of the correction equations, which is the topic of the next section.

Table 8.1: Costs of operator application for the three approaches. BLAS terms are used: MV(A) is $A\mathbf{x}$, AXPY = $\alpha\mathbf{x} + \mathbf{y}$. The solve of $\mathbf{y}$ from $LU\mathbf{y} = \mathbf{x}$, given LU-factors, is denoted by LU.

Approach	MV(A)	MV(B)	AXPY	MV(Z)	LU
complex (8.3.1)	2	2	4	4	2
real (8.4.1)	2	2	4	4	2
Sylvester (8.4.3)	2	2	4	4	2

8.4.5 Complexity and practical notes

Table 8.1 shows the costs of an application of the operator in equations (8.3.1), (8.4.1) and (8.4.3). Operations are counted in purely real operations: if $\mathbf{x} \in \mathbb{C}^n$, then an MV ($A\mathbf{x}$) costs two real MVs. Furthermore, operations are counted for $n \times n$ matrices and $n \times 1$ vectors, because in a practical situation the $2n \times 2n$ systems are never constructed explicitly.

Operator applications cost the same for all three approaches[1]. The approach in (8.4.3) is the most elegant approach because no complex arithmetic is involved at all for the RJDQZ algorithm. If the correction equation is solved exactly using the LU decomposition, it is more efficient to solve the complex correction equation: the solve of complex linear system of order n costs half the solve of a real linear system of order $2n$. If an iterative solver like GMRES is used, there is convergence within atmost n iterations for the complex correction equations, while the real variants requires at most $2n$ iterations. Therefore, in the RJDQR/RJDQZ implementations used in Sections 8.6.2 and 8.6.3, the complex correction equation is solved. At first sight this may seem to contradict the goal of RJDQZ. Note, however, that the goal of the RJDQZ method is not to avoid all complex arithmetic. The goal of RJDQZ is to exploit the realness of the pencil. This means that the methods aims to approximate real eigenvalues with real Petrov values and complex conjugate pairs of eigenvalues with complex conjugate pairs of Petrov pairs. In approximating real eigenvalues as much real arithmetic as possible is used, and in approximating complex conjugate pairs of eigenvalues complex arithmetic is used where it is needed.

8.5 RJDQZ versus JDQZ

For a precise account of the computational and memory costs of the JDQR and JDQZ methods, see [51]. In this section only the main differences in the costs between the JDQZ(d) and the RJDQZ method are mentioned.

[1] It must be noted that straightforward application of the operator in (8.4.3) costs an additional 3 SCALs ($\alpha\mathbf{x}$). Clever implementation saves these 3 SCALs.

8.5.1 Differences in Memory Costs

The orthonormal bases for the search and test space in the JDQZ method are expanded with one complex vector in each iteration. For the RJDQZ, the bases of the search and test space are expanded with one real vector if the selected Petrov value is real or with two real vectors if the selected Petrov value appears in a complex conjugate pair. This means that, although the dimension of the search and test space for the RJDQZ method can grow twice as fast as for the JDQZ method, the storage requirements are the same at most, and probably less. The numerical experiments give evidence that the larger subspace in the RJDQZ method is beneficial for the convergence.

8.5.2 Differences in Computational Costs

- The correction equation: When in the RJDQZ method a real Petrov value is selected, the correction equation can be solved in real arithmetic. In the original JDQZ method, the chance that a real Petrov value is selected, even in the process of approximating a real eigenvalue of the original real eigenvalue problem, is negligible. This is caused by the fact that all the projected eigenvalue problems in the JDQZ method are in principle complex, and due to rounding errors, all the Petrov values will be complex, also the Petrov values that are approximating real eigenvalues of the original real eigenvalue problem. These latter Petrov values will of course have small imaginary parts, but will be complex nonetheless. This means that when in a RJDQZ iteration a real Petrov value is selected, approximately halve the number of (real) matrix-vector products is needed (depending on how many iterations are used for the approximate solution of the correction equation), when compared to a JDQZ iteration. When in a RJDQZ iteration a Petrov value is selected that appears in a complex conjugate pair, then the JDQZ and the RJDQZ method need the same work for the approximate solution of the correction equation. Note that the RJDQZ method requires two implementations of the solver for the correction equation: a real and a complex version.

- The projected eigenproblem: In the RJDQZ method the real Schur forms of real projected eigenproblems are computed, but these may be twice as large as the complex projected eigenproblems that appear in the JDQZ method. Assume that computing a Schur form costs $\mathcal{O}(k^3)$ operations [64] and that an operation in complex arithmetic costs in average four operations in real arithmetic. Then it is easily deduced that computing the Schur form of a real eigenvalue problem costs about twice as much as computing the Schur form of a complex eigenvalue problem that is twice as small.

- Orthogonalization: The other point where the RJDQZ method may be computationally more expensive than the original JDQZ, is the orthogonalization procedures. One has to compare these two cases:

  – Orthogonalize a complex vector $\mathbf{x}$ against k other complex vectors. This

requires $4k$ real inner products for the projection plus 2 real inner products for the scaling.

– Orthogonalize two real vectors $\mathbf{a}$ and $\mathbf{b}$ against $2k$ other real vectors. This requires $2k$ inner products for projecting the first vector $\mathbf{a}$ plus one inner product for scaling. For the next vector $\mathbf{b}$, $2k+1$ inner products are needed for the projection plus one for the scaling. This adds up to $4k+3$ inner products.

In the worst case, one iteration of the RJDQZ method will cost about the work of one inner product and one (low dimensional) Schur decomposition more than an iteration of the original version of the JDQZ method. If the initial approximation is a real vector, then the cost of the extra inner product in the RJDQZ method is eliminated, and the orthogonalization process in the RJDQZ method will cost at most as much as in the JDQZ method.

8.6 Numerical Comparison

In this section RJDQZ method is compared with the JDQZ and the JDQZd method, using numerical experiments.

8.6.1 Hardware and software

The experiments were performed on a Sunblade 100 workstation using Matlab 6. The Matlab code that was used for the RJDQZ method is given in the appendix. The cpu-time results are included. Note that these cpu-time results do not always give very accurate information, but they at least give an impression of how the methods perform in comparison to each other. If not mentioned otherwise, the target value in each experiment equals 0, and the tolerance is set to 10^{-9}.

For all the results presented in this section, the correction equation is solved approximately using at most 10 iterations of the Bi-CGSTAB method [155] or the GMRES method [133]. No restart strategy is used in the JD part of the algorithm, in order to focus on the differences in the performance of the methods caused by the different strategies for expanding the search and test space.

8.6.2 Results for the Real JDQR Method

The ideas presented in this chapter are easily incorporated in a QR version of the algorithm for the standard eigenvalue problem. In this section, numerical results for the QR version are reported.

The first test matrix is CRY10000 from the NEP collection [1]. This is a large real unsymmetric standard eigenvalue problem with dimension 10000 by 10000. The target value $\tau = 7$ is selected (this corresponds to the rightmost eigenvalues). Six eigenvalues closest to the target are computed. The convergence tolerance is set to 10^{-8}. The preconditioner that is used for the correction equations is the

Table 8.2: Results for QR methods.

	Bi-CGSTAB			GMRES		
CRY10000	JDQR	JDQRd	RJDQR	JDQR	JDQRd	RJDQR
iterations	41		37	71		52
dim. search sp.	36		38	66		63
mat.vec.	1432		751	1532		733
Cpu (secs)	246		116	386		158
AF23560	JDQR	JDQRd	RJDQR	JDQR	JDQRd	RJDQR
iterations	77	70	49	46	37	32
dim. search sp.	71	65	78	40	32	42
mat.vec.	3002	2708	1603	1622	1244	782
Cpu (secs)	4071	4255	2825	2303	1686	1107
CC100	JDQR	JDQRd	RJDQR	JDQR	JDQRd	RJDQR
iterations	37	29	32	57	47	45
dim. search sp.	32	25	39	52	43	57
mat.vec.	1214	878	883	1226	1006	657
Cpu (secs)	5	3	2	17	10	3

incomplete LU (ILU) factorization of the matrix $A - \tau I$ (where A is the CRY10000 matrix) with drop tolerance 10^{-3}. In Table 8.2 the number of iterations, the number of matrix-vector products, and the dimension of the final search space is given. For the JDQR method, the dimension of the search space should be multiplied by two to obtain the number of *real* vectors that are needed to store the basis of the search space. Note that the search space dimension is smaller than the number of iterations for the JDQR method (and also for the JDQRd in later experiments) because every time a Ritz value has converged, its eigenvector is deflated from the search space.

The computed eigenvalues are all real. The selected Ritz values in the intermediate JD iterations need not be, and in fact were not, all real. This explains why both the RJDQR and JDQR constructed a search space of approximately the same dimension, while the RJDQR method required fewer iterations to build this search space. Note that if only real eigenvalues are computed, the JDQR and JDQRd method coincide. Therefore the results for the JDQRd method are not given for the CRY10000 matrix.

The second test problem is the matrix AF23560 [1]. The order of the test matrix is 23560. The preconditioner that is used for the correction equations is the ILU factorization of the AF23560 matrix with drop tolerance 10^{-3}. Seven eigenvalues with largest real part are computed. Three of the computed eigenvalues are real and the other four are formed by two complex conjugate pairs. Observe that again the RJDQR method needs fewer iterations but builds a subspace that is larger than the JDQR method. Note that, although the dimension of the search space for the RJDQR method is larger, it needs far less storage than the JDQR method to store the basis, since it only stores real vectors.

The test matrix CC100 is constructed for the purpose of comparing the JDQR and the RJDQR method in the case that only complex conjugate pairs of eigenvalues are computed. The matrix has order 100, has the numbers -1 to -100 on the

```
-1    1    0    0    0    0    0    0    0    0
-1   -2    1    0    0    0    0    0    0    0
 0    0   -3    1    0    0    0    0    0    0
 0    0   -1   -4    1    0    0    0    0    0
 0    0    0    0   -5    1    0    0    0    0
 0    0    0    0   -1   -6    0    0    0    0
 0    0    0    0    0    0   -7    0    0    0
 0    0    0    0    0    0    0   -8    0    0
 0    0    0    0    0    0    0    0   -9    0
 0    0    0    0    0    0    0    0    0  -10
```

Figure 8.1: The upper left 10 by 10 block of the matrix CC100.

diagonal, and has some off-diagonal elements only in the upper left corner. The upper left 10 by 10 corner is given below in Fig. 8.1. The six eigenvalues with largest real part are computed. No preconditioner is used for the correction equations.

If only complex pairs of eigenvalues are computed, the RJDQR method may need fewer matrix-vector products than the JDQR method. There can be two different reasons: the intermediate selected Ritz values can be real so that real arithmetic can be used in intermediate RJDQR iterations, and secondly, in case the selected Ritz value is not real, then the dimension of the search space grows faster in the RJDQR method, which may result in a faster convergence and thus fewer matrix-vector products.

Ritz value selection

Two complex conjugate eigenvalues have equal distance to a real target τ. In the JDQR method, the approximating Ritz values, however, do not necessarily have the same distance to the target. Therefore it may happen that in one iteration the Ritz value with positive imaginary part is selected and in the next iteration the Ritz value with negative imaginary part. This may hamper the convergence of the method, as illustrated in Fig. 8.2. The dotted line in the left graph shows the convergence of the JDQR method to an eigenvalue with nonzero imaginary part. In the right graph the imaginary part of the selected Ritz value is plotted versus the iteration number. When the sign of the maginary part of the selected Ritz value changes the residual becomes larger. In order to prevent the sign of the imaginary part of the selected Ritz value from changing, the JDQR method can be adapted such that only Ritz values with non-negative imaginary part are selected. The convergence of this adapted method, depicted with dashed lines, is faster. The RJDQR method (solid line) selects complex conjugate Ritz pairs instead and has the fastest convergence, most likely due to the larger search space.

8.6.3 Results for the Real JDQZ Method

The first test for the QZ methods is the generalized eigenvalue problem BFW782 [1]. The matrix A is non-symmetric and B is symmetric positive definite. Six eigenvalues with largest real part are computed. The preconditioner that is used for the correction equations is the ILU factorization of A with drop tolerance 10^{-3}. In Table 8.3 the number of iterations, the number of matrix-vector products, and

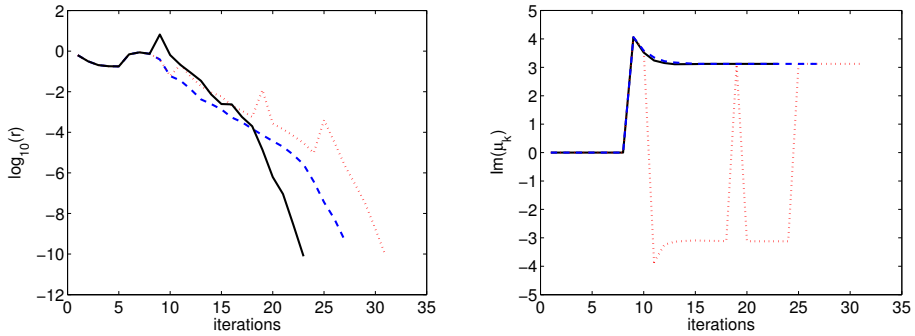

Figure 8.2: Results for matrix CC100. Left graph: the residual for the JDQR, adapted JDQR and RJDQR method versus the number of iterations for the convergence to an eigenvalue with nonzero imaginary part. Right graph: the imaginary part versus the number of iterations. Dotted lines: JDQR, dashed lines: adapted JDQR, solid lines: RJDQR.

the dimension of the final search space is given.

The computed eigenvalues are all real. Observe that the JDQZ and the RJDQZ method both need exactly the same number of iterations and build a search space of the same dimension. From the intermediate iterations (not shown), it can be concluded that the two methods perform exactly the same steps, the only difference being that the RJDQZ method performs the steps in real arithmetic and the JDQZ method performs the steps using complex arithmetic. This explains why the JDQZ method needs twice as many matrix-vector products as the RJDQZ method.

The generalized eigenvalue problem QG6468 arises in the stability analysis of steady states in the finite difference approximation of the QG model described in [40]. The eigenvalues and corresponding eigenvectors of interest are the ones with largest real parts. The matrix A is non-symmetric and B is symmetric positive semi definite. The ten eigenvalues with largest real part are computed. The preconditioner that is used for the correction equations is the ILU factorization of A with drop tolerance 10^{-7}.

For this matrix pencil, two of the computed eigenvalues are real, and the other computed eigenvalues form four complex conjugate pairs. One sees that the RJDQZ method needs fewer iterations, but builds a larger search space than the JDQZ method. The storage requirements for the RJDQZ method are, however, still less than for the JDQZ method.

For the problem ODEP400 [1], the eigenvalues and corresponding eigenvectors with largest real part are wanted. The matrix A is non-symmetric and B is symmetric positive semi definite. The twelve eigenvalues with largest real part are computed. These eigenalues appear in six complex conjugate pairs. The preconditioner that is used for the correction equations is the LU factorization of A.

For this matrix pencil, a similar conclusion as for the CC100 matrix can be drawn. The computed eigenvalues are all complex, but still the RJDQZ method

Table 8.3: Results for QZ methods.

	Bi-CGSTAB			GMRES		
BFW782	JDQZ	JDQZd	RJDQZ	JDQZ	JDQZd	RJDQZ
iterations	23		23	28		28
dim. search sp.	18		18	23		23
mat.vec.	558		279	456		228
Cpu (secs)	13		8	13		8
QG6468	JDQZ	JDQZd	RJDQZ	JDQZ	JDQZd	RJDQZ
iterations	78	59	47	97	67	53
dim. search sp.	69	51	76	88	59	87
mat.vec.	3176	2378	1744	2072	1412	1009
Cpu (secs)	1154	800	548	1090	640	412
ODEP400	JDQZ	JDQZd	RJDQZ	JDQZ	JDQZd	RJDQZ
iterations	33	32	22	31	26	21
dim. search sp.	22	22	24	20	16	29
mat.vec.	1202	1180	570	462	386	243
Cpu (secs)	13	14	4	8	6	3

needs fewer iterations and fewer matrix-vector products than the JDQZ method.

8.6.4 Non-real target

As mentioned in Remark 2, if the target value τ is non-real, the harmonic Petrov approach loses most of its advantages in the RJDQR and RJDQZ methods: when the search space V is expanded with the real vector x, in the harmonic Petrov approach, the test space should be expanded with the vector $w = (\kappa_0 A + \kappa_1 B)x$, where $\kappa_0 = (1+|\tau|^2)^{-1/2}$ and $\kappa_1 = -\tau(1+|\tau|^2)^{-1/2}$. In the case that τ is non-real, κ_1 will also be non-real, and thus the test space has to be expanded with a non-real vector. This spoils the realness of the projected eigenvalue problem and thus the idea behind the RJD methods breaks down.

One could of course abandon the harmonic approach, and choose κ_0 and κ_1 to be real. In this section two such alternatives to the harmonic approach are studied using numerical experiments. For the experiments, the pencil MHD416 [1] is used as an example. The six eigenvalues closest to the target value $\tau = 0.7i$ are computed. As preconditioner the LU factorization of $A-0.7iB$ is used. The results of the experiments in Table 8.4 show that the RJDQZ method is more effective than the JDQZd method.

The results for the JDQZ method are not shown in Table 8.4. The reason is the following. In order to obtain the same set of eigenvalues and eigenvectors as computed using the JDQZd and RJDQZ method, it would be required to run the JDQZ method twice, first with target $\tau = 0.7i$ and then with target $\tau = -0.7i$, since the JDQZ method does not automatically compute complex conjugate eigenvalues and vectors in pairs. This would result in an unfair comparison. One could of course add the complex conjugate of a computed eigenvalue and eigenvector when it is computed to the constructed Schur form, but this would in fact constitute a version of the JDQZd method.

Table 8.4: Results for JDQZd and RJDQZ for the MHD416 pencil.

	Bi-CGSTAB		GMRES	
Harm. Petr.	JDQZd	RJDQZ	JDQZd	RJDQZ
iterations	42		50	
dim. search sp.	32		40	
mat.vec.	1724		1080	
$\kappa_0 = 1$, $\kappa_1 = 0$	JDQZd	RJDQZ	JDQZd	RJDQZ
iterations	50	28	50	41
dim. search sp.	40	46	40	72
mat.vec.	2060	1136	1080	882
$\kappa_0 = 0$, $\kappa_1 = 1$	JDQZd	RJDQZ	JDQZd	RJDQZ
iterations	45	27	55	40
dim. search sp.	35	44	45	70
mat.vec.	1850	1094	1190	860

8.7 Conclusions

In this chapter, an adapted version of the Jacobi-Davidson method is presented, intended for real unsymmetric matrices or pencils.

In all the presented numerical experiments, the RJDQR and RJDQZ variants needed fewer iterations, fewer matrix-vector products and less storage than the original JD methods. The difference is most pronounced for the cases where only real eigenvalues are computed. The better performance of the RJD methods can be attributed to two reasons: first, the method uses real arithmetic where possible, which results in fewer matrix-vector products, and second, the dimension of the search space may grow twice as fast, while it does not use more storage. The limited additional costs for orthogonalization are compensated by far by the faster convergence due to the larger search space.

8.8 Appendix: Matlab-style code for RJDQZ method

In Algorithms 8.1-8.3, the Matlab-style code is presented for the RJDQZ method. The code for the approximate solution of the correction equation is not given. Note that the correction equation can be solved using real arithmetic if the approximate eigenvalue (α, β) is real. Otherwise it must be solved using complex arithmetic.

The routine "realqzsort" computes the ordered generalized real Schur form of the pencil (M_A, M_B), with respect to the target value τ. The logical output variable "complex" should be true if the harmonic Petrov value closest to the target has a nonzero imaginary part, and false if not. The routine "mgs" is a modified Gram-Schmidt routine as described in the appendix of [51].

Addendum

Except for some changes in notation, this chapter is published as [158]

> Tycho van Noorden and Joost Rommes, *Computing a partial generalized real Schur form using the Jacobi-Davidson method*, Numerical Linear Algebra with Applications **14** (2007), no. 3, 197–215.

Algorithm 8.1 The RJDQZ method

function $[Q, Z, R_A, R_B]$ =RJDQZ$(A, B, K, \tau, v_0, \epsilon, k_{max}, j_{min}, j_{max})$
$Q = [\,]$; $Z = [\,]$; $R_A = [\,]$; $Y = [\,]$; $H = [\,]$;
$V = [\,]$; $V_A = [\,]$; $V_B = [\,]$; $W = [\,]$; $M_A = [\,]$; $M_B = [\,]$;
$\gamma = \sqrt{1 + |\tau|^2}$; $\alpha_0 = \tau/\gamma$; $\beta_0 = 1/\gamma$; $k = 0$; $j = 0$;
while $k < k_{max}$
    if $j == 0$, then $v = v_0$, else solve correction equation for v
    Expand (Alg. 8.2)
    $[U_L, U_R, S_A, S_B, \text{complex}] = \text{realqzsort}(\tau, M_A, M_B)$;
    $j = j + 1$; found= 1;
    while found
        if complex
            $\alpha = S_A(1:2, 1:2)$; $\beta = S_B(1:2, 1:2)$;
            $q = VU_R(:, 1:2)$; $z = WU_L(:, 1:2)$;
            $r_A = V_A U_R(:, 1:2)$; $[r_A, s_A] = \text{mgs}(Z, r_A)$;
            $r_B = V_B U_R(:, 1:2)$; $[r_B, s_B] = \text{mgs}(Z, r_B)$;
            $r = r_A/\alpha - r_B/\beta$;
            found= $(||r|| < \epsilon)$ and $(j > 1 | k == k_{max} - 1)$;
            if $||r|| > \epsilon$
                $[U_L, U_R, S_A, S_B] = \text{qz}(\alpha, \beta)$;
                $\alpha = S_A(1,1)$; $\beta = S_B(1,1)$;
                $q = qU_R(:, 1)$; $z = zU_L(:, 1)$;
                $r_A = r_A U_R(:, 1)$; $[r_A, s_A] = \text{mgs}(Z, r_A)$;
                $r_B = r_B U_R(:, 1)$; $[r_B, s_B] = \text{mgs}(Z, r_B)$;
                $r = r_A/\alpha - r_B/\beta$;
            end
            $y = K^{-1}z$; $\tilde{\mathbf{x}} = [Q, q]$; $\tilde{Y} = [Y, y]$; $\tilde{\mathbf{z}} = [Z, z]$;
            $\tilde{H} = [H, Q^*y; q^*Y, q^*y]$;
        else
            $\alpha = S_A(1,1)$; $\beta = S_B(1,1)$; $q = VU_R(:, 1)$;
            $z = WU_L(:, 1)$; $y = K^{-1}z$;
            $r = V_A U_R(:, 1)$; $[r, s_A] = \text{mgs}(Z, r)$;
            $r_B = V_B U_R(:, 1)$; $[r_B, s_B] = \text{mgs}(Z, r_B)$;
            $r = r/\alpha - r_B/\beta$;
            found= $(||r|| < \epsilon)$ and $(j > 1 | k == k_{max} - 1)$;
            $\tilde{\mathbf{x}} = [Q, q]$; $\tilde{Y} = [Y, y]$; $\tilde{\mathbf{z}} = [Z, z]$;
            $\tilde{H} = [H, Q^*y; q^*Y, q^*y]$;
        end
        Found and restart (Alg. 8.3)
    end
end

Algorithm 8.2 RJDQZ: Expand

if isreal(v)
  $v = \text{mgs}(V, v)$; $v = v/||v||$; $v_A = Av$; $v_B = Bv$;
  $w = (\beta_0 v_A - \alpha_0 v_B)$; $w = \text{mgs}(Z, w)$;
  $w = \text{mgs}(W, w)$; $w = w/||w||$;
  $M_A = [M_A, W^* v_A; w^* V_A, w^* v_A]$;
  $M_B = [M_B, W^* v_B; w^* V_B, w^* v_B]$;
  $V = [V, v]$; $V_A = [V_A, v_A]$; $V_B = [V_B, v_B]$; $W = [W, w]$;
else
  $\tilde{v} = [\text{real}(v), \text{imag}(v)]$;
  for $i = 1 : 2$
    $v = \tilde{v}(:, i)$; $v = \text{mgs}(V, v)$; $v = v/||v||$; $v_A = Av$; $v_B = Bv$;
    $w = (\beta_0 v_A - \alpha_0 v_B)$; $w = \text{mgs}(Z, w)$;
    $w = \text{mgs}(W, w)$; $w = w/||w||$;
    $M_A = [M_A, W^* v_A; w^* V_A, w^* v_A]$;
    $M_B = [M_B, W^* v_B; w^* V_B, w^* v_B]$;
    $V = [V, v]$; $V_A = [V_A, v_A]$; $V_B = [V_B, v_B]$; $W = [W, w]$;
  end
end

Algorithm 8.3 RJDQZ: Found and restart

if found
  $Q = \tilde{\mathbf{x}}$; $Z = \tilde{\mathbf{z}}$;
  $R_A = [R_A, s_A; \text{zeros}(1 + \text{complex}, k), \alpha]$;
  $R_B = [R_B, s_B; \text{zeros}(1 + \text{complex}, k), \beta]$;
  $k = k + 1 + \text{complex}$; if $k \geq k_{max}$, break; end
  $Y = \tilde{Y}$; $H = \tilde{H}$;
  $J = [2 + \text{complex} : j]$; $j = j - 1 - \text{complex}$;
  $V = VU_R(:, J)$; $V_A = V_A U_R(:, J)$; $V_B = V_B U_R(:, J)$;
  $W = WU_L(:, J)$; $S_A = S_A(J, J)$; $S_B = S_B(J, J)$;
  $M_A = S_A$; $M_B = S_B$; $U_R = I$; $U_L = I$;
  $[U_L, U_R, S_A, S_B, \text{complex}] = \text{realqzsort}(\tau, M_A, M_B)$;
elseif $j \geq j_{max}$
  $j = j_{min}$; $J = [1 : j]$;
  $V = VU_R(:, J)$; $V_A = V_A U_R(:, J)$; $V_B = V_B U_R(:, J)$;
  $W = WU_L(:, J)$; $S_A = S_A(J, J)$; $S_B = S_B(J, J)$;
  $M_A = S_A$; $M_B = S_B$; $U_R = I$; $U_L = I$;
end