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

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

T.L. van Noorden and J. Rommes

Abstract

In this paper, 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 conjugated pair of eigenvalues, the method computes the two dimensional real invariant subspace spanned by the two corresponding complex conjugated eigenvectors. This is beneficial for memory costs and in many cases it also accelerates the convergence of the JD method. In numerical experiments, the RJDQZ variant is compared with the original JDQZ method.

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 [1]) 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 conjugated eigenvalues of the matrix (pencil) itself. Advantages of the real Schur form are that it requires less storage since for every complex conjugated pair of eigenvalues only two real vectors need to be stored instead of two complex vectors, and that complex conjugated pairs of eigenvalues always appear together.

In this paper, a variant of the JDQZ method [2] is considered for the computation of a partial generalized real Schur form of a matrix pencil. The original JDQZ method [2] 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 conjugated 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.

[*]This research was carried out in the context of the dutch NWO Scientific Computing project "Rapid Changes in Complex Flows" and the BRICKS (MSV1) project of the dutch government.

Algorithm 4.1 proposed in [3] solves the problem (1). 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 non-zero imaginary part. Algorithm 4.1 in [3] does not solve problem (2): the inner iteration still needs complex arithmetic, even when only real eigenvalues are computed.

The variant of the Jacobi-Davidson method that is implemented in this paper does take into account in the inner iteration that either a real eigenvalue and eigenvector are computed, or a two dimensional invariant subspace corresponding to a pair of complex conjugated eigenvalues. It is shown that this approach has several advantages.

In [4], it is stated that a real implementation of the JD method is used for a comparison with the Ricatti method, but implementation details and a comparison with the original JD method are not presented. Also in [2] (Remark 1), the authors hint at a real version of the JD method, but do not pursue this matter further.

The structure of this paper is as follows. In Section 2, the JDQZ method [2] 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.

2 A Jacobi-Davidson style QZ method

In this section the JDQZ method [2] for the construction of a partial generalized Schur form of an unsymmetric matrix pencil is discussed.

2.1 The Jacobi-Davidson Method for the Generalized Eigenvalue Problem

The Jacobi-Davidson (JD) method [5] iteratively computes approximations to eigenvalues, and their corresponding eigenvectors, that are close to some specified target τ, of the generalized unsymmetric eigenvalue problem

$$Aq = \mu Bq, \tag{1}$$

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 have orthonormal columns such that $V^*V = W^*W = I$. In the first part of an iteration, an approximation to an eigenvector of the generalized eigenvalues problem (1) is obtained from the projected eigenvalue problem

$$W^*AVu = \mu W^*BVu. \tag{2}$$

Note that 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 of the eigenvalue problem (2).

Suppose $(\tilde{\mu}, u)$ is the eigenpair of the projected eigenvalue problem (2), of which the eigenvalue $\tilde{\mu}$ is closest to τ. An approximation $(\tilde{\mu}, \tilde{q})$ to an eigenpair of the full sized eigenvalue problem (1) can be constructed by computing $\tilde{q} = Vu$. The residual vector r of the approximate eigenpair $(\tilde{\mu}, \tilde{q})$ is defined by

$$r := A\tilde{q} - \tilde{\mu}B\tilde{q}.$$

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 x of the linear equation

$$(I - \tilde{z}\tilde{z}^*)(A - \tilde{\mu}B)(I - \tilde{q}\tilde{q}^*)x = -r. \tag{3}$$

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

There are several possible choices for the complex numbers κ_0 and κ_1, and the performance of the JDQZ method is affected by this choice. An effective choice of κ_0 and κ_1 depends on whether the target value τ is located near extremal eigenvalues of the pencil (A, B) or whether τ is located in the interior of the spectrum of (A, B). For a detailed discussion on the choice of κ_0 and κ_1, see [2]. An effective choice for interior eigenvalues is $\kappa_0 = (1 + |\tau|^2)^{-1/2}$ and $\kappa_1 = -\tau(1 + |\tau|^2)^{-1/2}$. This choice corresponds to the harmonic Petrov value approach [2].

If P is a preconditioner for the matrix $(A - \tilde{\mu}B)$, then the correction equation (3) can be preconditioned as follows [2]:

$$(I - \frac{\hat{z}\tilde{q}^*}{\tilde{q}^*\hat{z}})P^{-1}(A - \tilde{\mu}B)(I - \frac{\hat{z}\tilde{q}^*}{\tilde{q}^*\hat{z}})x = -\tilde{r}, \tag{4}$$

with $\tilde{r}_k = (I - \frac{\hat{z}\tilde{q}^*}{\tilde{q}^*\hat{z}})P^{-1}r$, and $\hat{z} = P^{-1}\tilde{z}$.

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)

$$AQ_k = Z_k S_k, \quad BQ_k = Z_k T_k, \tag{5}$$

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 (5) is computed already. Now one would like to compute the next Schur vector q_{k+1} and the corresponding eigenvalue μ_{k+1}. It can be shown [2] that this Schur vector satisfies $Q_k^* q_{k+1} = 0$ and

$$(I - Z_k Z_k^*)(A - \mu_{k+1} B)(I - Q_k Q_k^*) q_{k+1} = 0. \tag{6}$$

Note that this is again a generalized eigenvalue problem, and therefore it can be solved using the JD method. The eigenvalue problem (6) shares $n-k$ eigenvalues with the generalized eigenvalue problem (1), and the already computed k eigenvalues of (1) are shifted to zero. The eigenvalue problem (6) is called the *deflated* eigenvalue problem. It is clear that the matrices V, containing the search space, and W, containing the test space in the JD method satisfy the extra condition $V^* Q_k = W^* Z_k = 0$. The projected generalized eigenvalue problem that has to be solved can be written as

$$W^*(I - Z_k Z_k^*) A (I - Q_k Q_k^*) V u = \mu W^*(I - Z_k Z_k^*) B (I - Q_k Q_k^*) V u. \tag{7}$$

Let $(\tilde{\mu}, \tilde{u})$ denote an eigenpair of the projected eigenvalue problem (7). Again an approximation $(\tilde{\mu}, \tilde{q})$ to the eigenpair (μ_{k+1}, q_{k+1}) of the eigenvalue problem (6) can be constructed by computing $\tilde{q} = V\tilde{u}$. In order to expand the search space V, an approximate solution x of the correction equation

$$(I - \tilde{z}\tilde{z}^*)(I - Z_k Z_k^*)(A - \tilde{\mu} B)(I - Q_k Q_k^*)(I - \tilde{q}\tilde{q}^*) x = -r, \tag{8}$$

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

Observe that complex arithmetic needs to be used to compute approximations to solutions of equation (8) whenever $\tilde{\mu}$ has a non-zero imaginary part, or whenever the matrices Q_k and Z_k contain entries with non-zero imaginary part.

When the JD correction equation (8) is solved exactly in each step of the JDQZ method, then it can be shown that the method converges 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) only approximately. In [2], based on an analogy of the JD method with Newton's method, it is suggested to solve the correction equation with a Krylov subspace method, and to stop the iterative solver when

$$||r_i||_2 < 2^{-j} ||r_0||_2,$$

where r_i is the ith residual vector of the Krylov subspace method, and j the iteration number of the JDQZ step. In [2] it is shown that this choice leads to an efficient method.

If $B = I$ the generalized eigenvalue problem (1) reduces to the standard eigenvalue problem $Aq = \mu q$. In this case it is possible to simplify the JDQZ method in order to reduce the memory requirements and the computational costs. This simplified method is called the JDQR method [2], and it computes a partial Schur form of the matrix A. For the standard eigenvalue problem the eigenvalues of the projected eigenvalue problem (7) are called (harmonic) Ritz values, instead of (harmonic) Petrov values [2].

3 A 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.

3.1 The RJDQZ Algorithm

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

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

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 conjugated pairs of eigenvalues of the pencil (A, B).

In [3] an adaptation of the JDQR algorithm for the standard eigenvalue problem is proposed (in Algorithm 4.1) to compute a partial real Schur form. Translated to the generalized case, this procedure proceeds just as the JDQZ method, the only difference being that if a complex eigenvalue μ_k, with corresponding eigenvector q_k, is computed of the eigenvalue problem (6), then the partial generalized Schur form is augmented not only with the eigenvector q_k but with a real basis of the space spanned by q_k and $\bar{q}_k$. In the proposed procedure, the search V and the test space W do not have to be real, and, therefore, the Petrov values (i.e. the eigenvalues of the projected eigenvalue problem) do not have to appear in complex conjugated pairs. This causes difficulties for the identification of complex pairs of eigenvalues of the original eigenvalue problem, see e.g. Chapter 8 in [6], and it also introduces additional rounding errors when an computed approximate eigenvalue with small imaginary part is replaced by its real part.

A way around this problem is to keep the search and test space real, which can be done as follows. 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 (2):

$$W^T AVu = \mu W^T BVu.$$

Since the projected matrices $W^T AV$ and $W^T BV$ are real, the eigenvalues are either real or form a complex conjugated pair, and since the projected eigenvalue problem is small, all eigenvalues can be computed accurately and cheaply. From these eigenvalues one eigenvalue (or complex conjugated pair) is selected with a given selection criterion (closest to a target value, or largest real part, etc.). Denote the selected Petrov value by $\tilde{\mu}$ and the corresponding eigenvector by $\tilde{u}$.

In the actual algorithm, instead of an eigenvalue decomposition of the projected eigenvalue problem (2), a sorted real generalized Schur form is computed. How this form is computed in an efficient and stable way can be found in [7, 8, 3] for the standard eigenvalue problem and in [9, 10, 11] 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 conjugated pair.

If the selected Petrov value $\tilde{\mu}$ is real then the matrix and the right hand side in the correction equation (8):

$$(I - \tilde{z}\tilde{z}^*)(I - Z_k Z_k^*)(A - \tilde{\mu}B)(I - Q_k Q_k^*)(I - \tilde{q}\tilde{q}^*)x = -r,$$

are both real, and, assuming that a Krylov subspace solver is used, the Krylov subspace build by the solver will also be real, and therefore also the approximate solution will be real. In this case the correction equation can be solved using real arithmetic. This also holds in the preconditioned case, as long as the preconditioner is real. This means that if the selected Petrov value is real, then the search and test space are expanded with a real vector.

If the selected Petrov value μ has non-zero imaginary part, then the matrix and the right hand side in the correction equation (8) also have non-zero imaginary parts, and the JD correction equation will have to be solved using complex arithmetic, and one will obtain a complex approximate solution v. In order to keep the search space real, it is expanded with the two dimensional real space $U = \text{span}\{\Re(v), \Im(v)\}$, which contains the vector v. It is easily seen that the space U is also spanned by v and its complex conjugate $\bar{v}$ and it is instructive to think of the space U as an approximation to a two dimensional generalized invariant subspace that can be spanned by two real vectors.

Remark 1: If the selected Petrov value has non-zero imaginary part, then there exists also a Petrov value $\bar{\mu}$. Assuming the target τ to be real, then both μ and $\bar{\mu}$ have an equal distance to τ. In this case selecting the Petrov value might appear to be a problem, but in fact it is irrelevant whether μ or $\bar{\mu}$ is selected, since it is not hard to see that in both cases the space U with which the search space is expanded will be the same. This also solves a problem from which the original JD method suffers when the target value is real (see Section 6.2.1 for more details).

Remark 2: Note that if the target value τ has a non-zero imaginary part then κ_1 (see below equation (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 looses 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 [12].

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.

4.1 Complex correction equation

For a Petrov pair (μ, q), the JDQZ correction equation is of the following form:

$$(I - ZZ^*)(A - \theta B)(I - QQ^*)t = -r, \tag{9}$$

with $Q = [Q_k, q]$ and $Z = [Z_k, z]$. If $\theta \in \mathbb{C}$ and $q \in \mathbb{C}^n$, the correction equation becomes complex and can be solved using complex arithmetic. To keep the search space real, it is expanded with the two dimensional real space $\text{span}(\Re(t), \Im(t))$.

4.2 Real variant of complex correction equation

Let $\theta = \nu + i\omega$, $t = u + iv$ and $r = x + iy$. Then it follows that

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

The matrices with already converged right and left Schur vectors Q_k and Z_k are real. Let $q = q_1 + iq_2$ with $q_1^* q_1 + q_2^* q_2 = 1$. Then

$$(I - qq^*)t = (I - (q_1 q_1^* + q_2 q_2^*))u + (q_2 q_1^* - q_1 q_2^*)v + i((I - (q_1 q_1^* + q_2 q_2^*))v - (q_2 q_1^* - q_1 q_2^*)u).$$

The equivalent real block formulation becomes

$$(I - qq^*)t \Longleftrightarrow P_q \begin{bmatrix} u \\ v \end{bmatrix} \equiv \begin{bmatrix} I - (q_1 q_1^* + q_2 q_2^*) & q_2 q_1^* - q_1 q_2^* \\ -(q_2 q_1^* - q_1 q_2^*) & I - (q_1 q_1^* + q_2 q_2^*) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}.$$

By using $q_1^* q_1 + q_2^* q_2 = 1$ and some basic linear algebra, it can be shown that P_q is indeed a projector. In a similar way, P_z for $z = z_1 + iz_2$ and $z_1^* z_1 + z_2^* z_2 = 1$ can be defined as

$$P_z \equiv \begin{bmatrix} I - (z_1 z_1^* + z_2 z_2^*) & z_2 z_1^* - z_1 z_2^* \\ -(z_2 z_1^* - z_1 z_2^*) & I - (z_1 z_1^* + z_2 z_2^*) \end{bmatrix}.$$

The real equivalent of the correction equation (9) becomes

$$P_z \mathcal{Z}_k \begin{bmatrix} A - \nu B & \omega B \\ -\omega B & A - \nu B \end{bmatrix} \mathcal{Q}_k P_q \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} x \\ y \end{bmatrix}, \tag{10}$$

where

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

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 $\begin{bmatrix} q_1 & q_2 \end{bmatrix}$ is

$$\begin{bmatrix} x & y \end{bmatrix} = A \begin{bmatrix} q_1 & q_2 \end{bmatrix} - B \begin{bmatrix} q_1 & q_2 \end{bmatrix} \begin{bmatrix} \nu & \omega \\ -\omega & \nu \end{bmatrix}.$$

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

$$(I - ZZ^*)(A(I - QQ^*) \begin{bmatrix} u & v \end{bmatrix} - B(I - QQ^*) \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} \nu & \omega \\ -\omega & \nu \end{bmatrix}) = - \begin{bmatrix} x & y \end{bmatrix}$$

The equivalent block formulation becomes

$$P_z \mathcal{Z}_k \begin{bmatrix} A - \nu B & \omega B \\ -\omega B & A - \nu B \end{bmatrix} \mathcal{Q}_k P_q \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} x \\ y \end{bmatrix}, \tag{11}$$

which is the same as the one obtained in (10).

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 [2]). The selected approximate Schur quartet for the deflated problem is $(\begin{bmatrix} q_1 & q_2 \end{bmatrix}, \begin{bmatrix} z_1 & z_2 \end{bmatrix}, S, T)$, with $S, T \in \mathbb{R}^{2\times 2}$, T upper triangular, $\begin{bmatrix} q_1 & q_2 \end{bmatrix} \perp Q_k$ and $\begin{bmatrix} z_1 & z_2 \end{bmatrix} \perp Z_k$. The residual is computed as

$$\begin{bmatrix} x & y \end{bmatrix} = (I - Z_k Z_k^*)(A \begin{bmatrix} q_1 & q_2 \end{bmatrix} S^{-1} - B \begin{bmatrix} q_1 & q_2 \end{bmatrix} T^{-1}),$$

and the correction equation becomes

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

4.4 Approximate solution of the correction equation

The real and imaginary part of the exact solution of (9) and the exact solutions of (10) and (11), and (12) 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 $A - \theta B$ in equation (9) are also eigenvalues of the equivalent real block matrix in equations (10) and (11), together with their complex conjugates and furthermore that the condition numbers of the matrices are the same..

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

$$C = \begin{bmatrix} A - \nu B & \omega B \\ -\omega B & A - \nu B \end{bmatrix} \in \mathbb{R}^{2n\times 2n} \tag{13}$$

are $(\mu_j, \left[v_j^T, (-iv_j)^T\right]^T)$, $(\bar{\mu_j}, \left[v_j^T, (-iv_j)^T\right]^*)$ for $j = 1, \ldots, n$.
Furthermore Cond(C)=Cond($A - \theta B$).

Proof. For an eigenpair (μ, v) of $A - \theta B$ it holds that

$$(A - \nu B)v - \omega Biv = \mu v, \qquad -(A - \nu B)iv - \omega Bv = -\mu iv.$$

Using this it easily follows that

$$\begin{bmatrix} A - \nu B & \omega B \\ -\omega B & A - \nu B \end{bmatrix} \begin{bmatrix} v \\ -iv \end{bmatrix} = \mu \begin{bmatrix} v \\ -iv \end{bmatrix}.$$

The first part of the proposition follows by noting that because matrix (13) 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 $x = v + iw \in \mathbb{C}^n$ it holds that

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

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

Using similar arguments, this relation can be extended to the operator in equation (12). From proposition 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 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.

Table 1: Costs of operator application for the three approaches. BLAS terms are used: MV(A) is Ax, AXPY $= \alpha x + y.\alpha x$. The solve of y from $LUy = x$, given LU-factors, is denoted by LU.

Approach	MV(A)	MV(B)	AXPY	MV(Z)	LU
complex (9)	2	2	4	4	2
real (10)	2	2	4	4	2
Sylvester (12)	2	2	4	4	2

The three approaches, however, may lead to different approximate solutions because the inner product of two complex n-vectors is different from the inner product of the equivalent real $2n$-vectors. Furthermore, it will most likely require more steps of the linear solver to reduce the residual norm to a certain tolerance for a real problem of size $2n$ than for a complex problem of size n. This is confirmed by numerical experiments.

4.5 Complexity and practical notes

Table 1 shows the costs of an application of the operator in equations (9), (10) and (12). Operations are counted in purely real operations: if $x \in \mathbb{C}^n$, then an MV (Ax) 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 (12) is the most elegant approach because no complex arithmetic is involved at all for the RJDQZ algorithm. If the correction equation is solved exactly, 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 $2n$. Furthermore, if an iterative method is used to solve the correction equation approximately, it is expected that within a fixed number of iterations, the approximate solution of the complex correction equation will be most accurate. Therefore, in practice the most efficient approach will be to solve the complex correction equation.

4.6 Numerical examples

The three approaches are equivalent, but in finite arithmetic rounding errors may lead to different results. Also, if an iterative method is used to solve the correction equation, the approximate solutions may differ, as explained above. This effect may be limited, however, because the condition number of the respective operators remains the same. The numerical experiments in this section confirm this. If the correction equation is solved exactly, the differences between

[1]It must be noted that straightforward application of the operator in (12) costs an additional 3 SCALs (αx). Clever implementation saves these 3 SCALs.

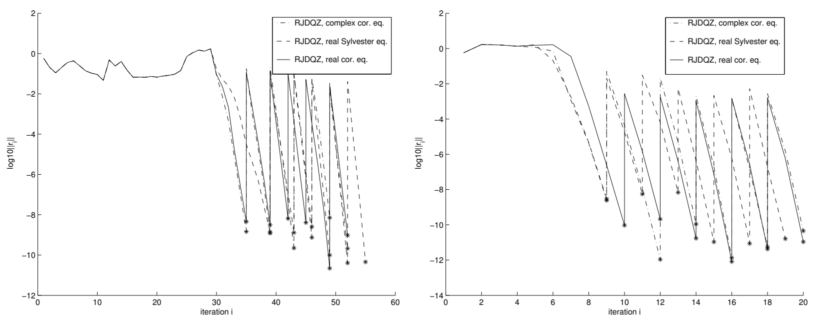

Figure 1: Convergence history for the six rightmost eigenvalues of CC100 with GMRES(10) (left) and GMRES(10) with $LU = A$ preconditioning.

the three approaches are hardly observable, as expected, and therefore they are not shown. If the correction equation is solved with an iterative solver, the differences are more pronounced but not dramatic. Figure 1 shows the convergence history for the problem CC100 ($B = I$) with GMRES(10) as solver (left graph), and with GMRES(10) with $LU = A$ as preconditioner (right graph). Solving the complex correction equation leads to the fastest convergence, but the approaches differ by at most three iterations.

Figure 2 shows the convergence history for the problem ODEP400 with GMRES(5) and GMRES(10). For both processes the preconditioner is $LU = A$. For the GMRES(10) process, the difference is more pronounced: again solving the complex correction equation is the most efficient. It is also clearly observable that this approach takes the most advantage from an increased number of GMRES iterations: apparently it is able to reduce the residual more than the other two approaches in the same number of iterations. Note that the convergence histories for the different ways of solving the correction equation coincide until the first complex Ritz pair is selected.

5 RJDQZ versus JDQZ

For a precise account of the computational and memory costs of the JDQR and JDQZ methods, see [2]. In this section only the main differences in the costs between the JDQZ and the RJDQZ method are mentioned.

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 (depending on the implementation). For the RJDQZ, the bases of the search and test space are expanded with one real vector if the selected Ritz value is real or with two real vectors if the selected Ritz value appears in a complex conjugated pair. This means that,

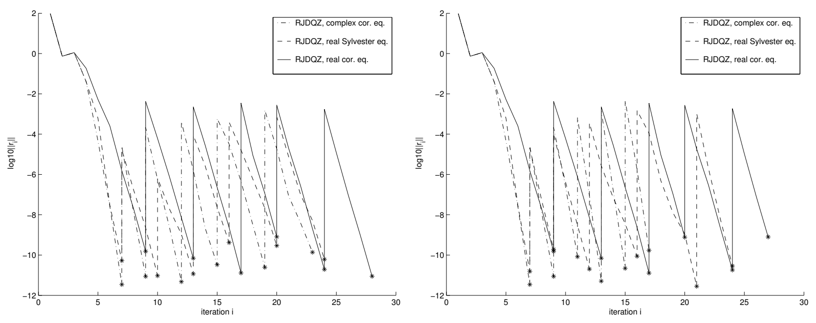

Figure 2: Convergence history for the six rightmost eigenvalues of ODEP400 with $LU = A$ preconditioning for GMRES(5) (left) and GMRES(10).

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.

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 this needs not be the case due to rounding errors: an approximation of a real eigenvalue can be a complex Ritz (Petrov) value with very small imaginary part. This approximately halves the number of (real) matrix-vector products that is needed (depends on how many iterations are used for the approximate solution of the correction equation). When a Petrov value is selected that appears in a complex conjugated pair, then the JDQZ and the RJDQZ method need the same work for the approximate solution of the correction equation. Note that the RJDQZ methods 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 $\mathcal{O}(n^3)$ operations [13] 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 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 a and b against $2k$ other real vectors. This requires $2k$ inner products for projecting the first vector a plus one inner product for scaling. For the next vector 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.

6 Numerical Comparison

The RJDQZ method is compared with two variants of the JDQZ method: the original JDQZ method as described in [2] and a variant that here is denote by JDQZd, which is the method described for the standard eigenvalue problem in [3]. This method proceeds just as the original JDQZ method, except when one has computed an eigenvalue with non-zero imaginary part. In that case the partial generalized Schur form is augmented not only with the corresponding eigenvector, but also with its complex conjugate.

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 Tables 5-7. 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}

6.2 Results for the Real JDQR Method

The ideas presented in this paper 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. For all the results presented in this section, the correction equation is solved approximately using at most 10 iterations of

Table 2: Results for QR methods

	Bi-CGSTAB			GMRES		
CRY10000	JDQR	JDQRd	RJDQR	JDQR	JDQRd	RJDQR
iterations	46		42	76		57
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	83	74	53	52	41	36
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	42	31	34	62	49	47
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

the Bi-CGSTAB method [14] or the GMRES method [15]. 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.

For the first test matrix is the matrix CRY10000 (from the NEP collection at http://math.nist.gov/MatrixMarket/). This is a large real unsymmetric standard eigenvalue problem that arises from the stability analysis of a crystal growth problem. The matrix has 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 incomplete LU (ILU) factorization of the matrix $A - \tau I$ (where A is the CRY10000 matrix) with drop tolerance 1e-3. In Table 2 the number of iterations, the number of matrix-vector products, and the dimension of the final search space is given. For the dimension of the final search space, it is important to note that no restarting is used, and also that in the tables the dimension of the search space is given, which means that for the original, complex JDQR method, this number should be multiplied by two to obtain the number of *real* vectors that are needed to store the basis of 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.

For the second test problem, some of the eigenvalues with largest real part of the matrix AF23560 are computed (also from the NEP collection at MatrixMarket). The test matrix arises from the transient stability analysis of

```
-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 3: The upper left 10 by 10 block of the matrix CC100.

Navier-Stokes solvers. The order of the test matrix is 23560. The eigenvalues and eigenvectors are associated with small perturbation analysis of a finite difference representation of the Navier-Stokes equations for flows over airfoils. The preconditioner that is used for the correction equations is the ILU factorization of the AF23560 matrix with drop tolerance 1e-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 conjugated 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 conjugated pairs of eigenvalues are computed. The matrix has order 100, has the numbers -1 to -100 on the 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. 3. The six eigenvalues with largest real part are computed. No preconditioner is used for the correction equations.

Observe that even in cases when 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.

6.2.1 Ritz value selection

When a real target is supplied, and a complex conjugated pair of eigenvalues is to be computed, then these two eigenvalues have exactly the same distance to the target. The approximating Ritz values do not necessarily have the same distance to the target. Therefore it may happen that in one iteration the Ritz

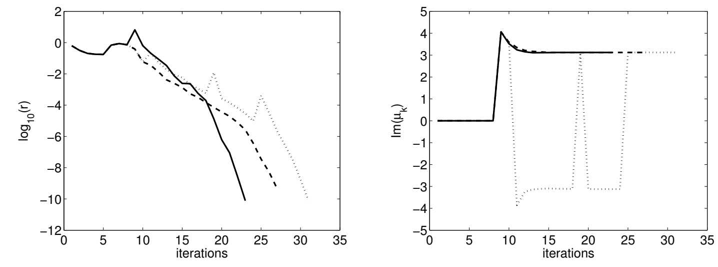

Figure 4: Left graph: the residual for the JDQR, adapted JDQR and RJDQR method versus the number of iterations for the convergence to an eigenvalue with non-zero imaginary part. Right graph: the imaginary part versus the number of iterations. Dotted lines: JDQR, dashed lines: adapted JDQR, solid lines: RJDQR.

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. This is illustrated in Fig. 4. The dotted line in the left and right graph shows the convergence of the JDQR method to an eigenvalue with non-zero imaginary part. In the right graph the imaginary part of the selected Ritz value is plotted versus the iteration number. The sign of the imaginary part of the selected Ritz value changes a number of times. Note that when this happens 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 is depicted with the dashed lines. For the adapted method, the sign of imaginary part of the selected Ritz value does not change, and the convergence is faster. The solid line depicts the convergence of the RJDQR method. For the RJDQR method it does not matter if the sign of the imaginary part of the selected Ritz value changes. For this particular example it does not change. The convergence of the RJDQR method is faster than the convergence of the adapted JDQR method. This is entirely due to the larger search space.

6.3 Results for the Real JDQZ Method

As for the JDQR method, for all the numerical results presented in this section, the correction equation is solved approximately using at most 10 iterations of Bi-CGSTAB [14] or GMRES [15]. No restart strategy is used in the JD part of the algorithm.

The generalized eigenvalue problem BFW782 (from the NEP collection at MatrixMarket) arises in the finite element analysis of Maxwell's equation for finding the propagating modes and magnetic field profiles of a rectangular waveguide filled with dielectric and PEC structures. The eigenvalues and cor-

Table 3: Results for QZ methods

	Bi-CGSTAB			GMRES		
BFW782	JDQZ	JDQZd	RJDQZ	JDQZ	JDQZd	RJDQZ
iterations	28		28	33		33
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	87	64	52	106	72	58
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	44	37	27	42	31	26
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

responding eigenvectors of interest are the ones with positive real parts, which correspond to the propagation modes of a waveguide. 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 1e-3. In Table 3 the number of iterations, the number of matrix-vector products, and 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 [16]. 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. 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 1e-7.

For this matrix pencil, two of the computed eigenvalues are real, and the other computed eigenvalues form four complex conjugated 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 is, however, still less than for the JDQZ method.

The generalized eigenvalue problem ODEP400 (from the NEP collection at MatrixMarket) is obtained as the finite differences approximation of the differ-

Table 4: Restarting JDQR and RJDQR for the CRY10000 matrix, using 10 iterations of Bi-CGSTAB for the correction equation.

		JDQR	RJDQR
$j_{min} = 3$	iterations	>100	>100
$j_{max} = 6$	mat.vec.	?	?
$j_{min} = 6$	iterations	95	56
$j_{max} = 12$	mat.vec.	3640	1387
$j_{min} = 5$	iterations	>100	>100
$j_{max} = 10$	mat.vec.	?	?
$j_{min} = 10$	iterations	49	40
$j_{max} = 20$	mat.vec.	1708	862
no restarts	iterations	46	37
	mat.vec.	1582	841

ential equation

$$y'' + \mu^2 y = 0,$$

with boundary conditions

$$y(0) = 0 \text{ and } y'(0) + 0.001y'(1) = 0.$$

Eigenvalues and corresponding eigenvectors with largest real part are computed. The matrix A is non-symmetric and B is symmetric positive semi definite. Six pairs of complex conjugate eigenvalues that are computed with largest real part. 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 needs fewer iterations and fewer matrix-vector products than the JDQZ method.

6.4 Restarts

In this section, the effect of restarts is studied. In Tables 4-7, the number of iterations and the number of real matrix-vector products for the different methods and for different restart parameters j_{min} and j_{max} are given. For a fair comparison of the different methods, one should compare the methods JDQR and JDQRd with parameters j_{min} and j_{max} with the RJDQR method with parameters $2j_{min}$ and $2j_{max}$. In this case the memory usage and the work per iteration is approximately equal for all the methods. The same holds for the QZ methods. In all the experiments, the RJDQR and RJDQZ variants need fewer iterations and fewer matrix-vector products than the other methods, also with restarts incorporated.

6.5 Non-real target

As mentioned before in Remark 2, if the target value τ is non-real, the harmonic Petrov approach looses most of its advantages in the RJDQR and RJDQZ

Table 5: Restarting JDQR, JDQRd and RJDQR for the CC100 matrix, using 10 iterations of Bi-CGSTAB for the correction equation.

		JDQR	JDQRd	RJDQR
$j_{min} = 3$	iterations	50	37	34
$j_{max} = 6$	mat.vec.	1550	1130	1030
$j_{min} = 6$	iterations	42	32	32
$j_{max} = 12$	mat.vec.	1214	920	883
$j_{min} = 5$	iterations	44	34	33
$j_{max} = 10$	mat.vec.	1298	1004	925
$j_{min} = 10$	iterations	42	33	31
$j_{max} = 20$	mat.vec.	1214	962	820
no restarts	iterations	42	33	31
	mat.vec.	1214	962	820

Table 6: Restarting JDQZ and RJDQZ for the BFW782 pencil, using 10 iterations of GMRES for the correction equation.

		JDQZ	RJDQZ
$j_{min} = 3$	iterations	30	31
$j_{max} = 6$	mat.vec.	530	287
$j_{min} = 6$	iterations	28	28
$j_{max} = 12$	mat.vec.	486	243
$j_{min} = 5$	iterations	28	28
$j_{max} = 10$	mat.vec.	486	243
$j_{min} = 10$	iterations	29	29
$j_{max} = 20$	mat.vec.	508	254
no restarts	iterations	29	29
	mat.vec.	508	254

Table 7: Restarting JDQZ, JDQZd and RJDQZ for the ODEP400 pencil, using 10 iterations of GMRES for the correction equation.

		JDQZ	RJDQZd	RJDQZ
$j_{min} = 3$	iterations	65	52	39
$j_{max} = 6$	mat.vec.	1168	1014	705
$j_{min} = 6$	iterations	49	36	27
$j_{max} = 12$	mat.vec.	816	662	441
$j_{min} = 5$	iterations	56	42	31
$j_{max} = 10$	mat.vec.	970	794	529
$j_{min} = 10$	iterations	41	34	22
$j_{max} = 20$	mat.vec.	640	618	331
no restarts	iterations	41	33	22
	mat.vec.	640	596	331

Table 8: Results for JDQZd and RJDQZ for the MHD416 pencil.

	Bi-CGSTAB		GMRES	
Harm. Petr.	JDQZd	RJDQZ	JDQZd	RJDQZ
iterations	47		55	
dim. search sp.	32		40	
mat.vec.	1724		1080	
$\kappa_0 = 1$, $\kappa_1 = 0$	JDQZd	RJDQZ	JDQZd	RJDQZ
iterations	55	33	55	46
dim. search sp.	40	46	40	72
mat.vec.	2060	1136	1080	882
$\kappa_0 = 0$, $\kappa_1 = 1$	JDQZd	RJDQZ	JDQZd	RJDQZ
iterations	50	32	60	45
dim. search sp.	35	44	45	70
mat.vec.	1850	1094	1190	860

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 (available at MatrixMarket) 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 show that though the harmonic Petrov approach is most effective for the JDQZd method, the RJDQZ method may be more effective even if it is not using the harmonic Petrov approach.

7 Conclusions

In this paper, an adapted version of the Jacobi-Davidson method is presented. This version is 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 (real) matrix-vector products, and second, the dimension of the search space may grow twice as fast (while it does not use more storage) which accelerates the convergence, resulting in fewer iterations.

A Matlab-style code for RJDQZ method

In the Figures 5-7, 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 is 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 leading harmonic Petrov value (i.e., the harmonic Petrov value closest to the target) has a non-zero imaginary part, and should be false is the leading harmonic Petrov value is real.

The routine "mgs" is a modified Gram-Schmidt routine as described in the appendix of [2].

References

[1] D. Bindel, J. W. Demmel, and M. J. Friedman. Continuation of invariant subspaces for large and sparse bifurcations problems. In *online proceedings of SIAM Conference on Applied Linear Algebra, Williamsburg*, 2003. http://www.siam.org/meetings/la03/proceedings/.

[2] Diederik R. Fokkema, Gerard L. G. Sleijpen, and Henk A. Van der Vorst. Jacobi-Davidson style QR and QZ algorithms for the reduction of matrix pencils. *SIAM J. Sci. Comput.*, 20(1):94–125 (electronic), 1998.

[3] J. H. Brandts. Matlab code for sorting real Schur forms. *Numer. Linear Algebra Appl.*, 9(3):249–261, 2002.

[4] J. H. Brandts. The Riccati algorithm for eigenvalues and invariant subspaces of matrices with inexpensive action. *Linear Algebra Appl.*, 358:335–365, 2003. Special issue on accurate solution of eigenvalue problems (Hagen, 2000).

[5] Gerard L. G. Sleijpen and Henk A. Van der Vorst. A Jacobi-Davidson iteration method for linear eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 17(2):401–425, 1996.

[6] Youcef Saad. *Numerical methods for large eigenvalue problems.* Algorithms and Architectures for Advanced Scientific Computing. Manchester University Press, Manchester, 1992.

[7] Zhaojun Bai and James W. Demmel. On swapping diagonal blocks in real Schur form. *Linear Algebra Appl.*, 186:73–95, 1993.

[8] A. Bojanczyk and P. Van Dooren. Reordering diagonal blocks in real schur form. In M. Moonen, G. H. Golub, and B. De Moor, editors, *Linear Algebra*

for Large-Scale and Real-Time Applications. Kluwer Academic Publishers, 1993.

[9] B. Kågström. A direct method for reordering eigenvalues in the generalized real schur form of a regular matrix pair (a, b). In M. Moonen, G. H. Golub, and B. De Moor, editors, *Linear Algebra for Large-Scale and Real-Time Applications.* Kluwer Academic Publishers, 1993.

[10] B. Kågström and P. Poromaa. Computing eigenspaces with specified eigenvalues of a regular matrix pair (a, b) and condition estimation: Theory, algorithms and software. Technical Report UMINF 94.04, Institute for Information Processing, University of Umeå, Umeå, Sweden, 1994. LAPACK Working Note 87.

[11] P. Van Dooren. A generalized eigenvalue approach for solving Riccati equations. *SIAM J. Sci. Statist. Comput.*, 2(2):121–135, 1981.

[12] Beresford N. Parlett and Youcef Saad. Complex shift and invert strategies for real matrices. *Linear Algebra Appl.*, 88/89:575–595, 1987.

[13] Gene H. Golub and Charles F. Van Loan. *Matrix computations*, volume 3 of *Johns Hopkins Series in the Mathematical Sciences.* Johns Hopkins University Press, Baltimore, MD, second edition, 1989.

[14] H. A. van der Vorst. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 13(2):631–644, 1992.

[15] Youcef Saad and Martin H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7(3):856–869, 1986.

[16] H. A. Dijkstra and C. A. Katsman. Temporal variability of the wind-driven quasi-geostrophic double gyre ocean circulation: basic bifurcation diagrams. *Geophys. Astrophys. Fluid Dynamics*, 85:195–232, 1997.

```
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 $j == 0$
        $v = v_0$
    else
        Solve correction equation for $v$
    end
    input(Expand)
    $[U_L, U_R, S_A, S_B, \text{complex}] = \text{realqzsort}(\tau, M_A, M_B)$;
    $j = j + 1$; 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
            $y = K^{-1}z$;
            $\tilde{Q} = [Q, q]$; $\tilde{Y} = [Y, y]$; $\tilde{Z} = [Z, z]$;
            $\tilde{H} = [H, Q^*y; q^*Y, q^*y]$;
        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 $(j > 1 | k = k_{max} - 1)$;
            $\tilde{Q} = [Q, q]$; $\tilde{Y} = [Y, y]$; $\tilde{Z} = [Z, z]$;
            $\tilde{H} = [H, Q^*y; q^*Y, q^*y]$;
        end
        input Found and restart
    end
end
```

Figure 5: The RJDQZ method.

```
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
```

Figure 6: Expand

```
if found
        $Q = \tilde{Q}$; $Z = \tilde{Z}$;
        $R_A = [R_A, s_A; \text{zeros}(1 + \text{complex}, k), \alpha]$;
        $R_B = [R_B, s_B; \text{zeros}(1 + \text{complex}, k), \beta]$;
        $k = k + 1 + \text{complex}$; 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
```

Figure 7: Found and restart