COMPUTING A PARTIAL REAL ORDERED GENERALIZED SCHUR FORM USING THE JACOBI-DAVIDSON METHOD.

TYCHO VAN NOORDEN *

Abstract. In this paper, an adapted 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 matrices or real 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 [6]) 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 only real vectors need to be stored, and that complex conjugated pairs of eigenvalues always appear together.

In [2], the author proposes an algorithm, that involves the Jacobi-Davison method, to compute a partial real Schur form of a large matrix. 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. In the proposed algorithm, the inner iteration does not "know" that a real partial Schur form is desired.

In this paper, a variant of the Jacobi-Davidson method is implemented that 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 [1], the author states that he uses a real implementation of the JD method for a comparison with the Ricatti method, but he does not give implementation details and a comparison with the original JD method. Also in [5] (Remark 1), the authors hint at a real version of the JD method, but do not pursue this matter further.

2. The JDQZ Method. In this section the JDQZ method for the construction of a partial generalized Schur form of a matrix pencil is discussed.

2.1. The Jabobi-Davidson Method for the Generalized Eigenvalue Problem. The Jacobi-Davidson (JD) method [8] computes iteratively approximations to eigenvalues and eigenvectors of the generalized eigenvalue problem

$$Ay = \mu By. \tag{2.1}$$

In each step, a search subspace $\mathrm{span}(V)$ and a test subspace $\mathrm{span}(W)$ are constructed. V and W are complex $n \times j$ matrices with $j \ll n$ and have orthonormal columns

*e-mail: noorden@math.uu.nl, Phone: +31 30 2531462, Utrecht University, Budapestlaan 6, Utrecht, The Netherlands.

such that $V^*V = W^*W = I$. An approximation to eigenvectors of the generalized eigenvalues problem (2.1) are obtained form the projected eigenvalue problem

$$W^*AVu = \mu W^*BVu. \tag{2.2}$$

Note that this is an eigenvalue problem of small size, so for example, the QZ method can be used to compute all the eigenvalues and eigenvectors of the eigenvalue problem (2.2).

The next step in the JDQZ method is the expansion of the search and test space. Suppose (μ_k, q_k) is an approximation to an eigenpair, then the residual vector r_k of the approximate eigenpair is defined by

$$r_k := Aq_k - \mu_k Bq_k. \tag{2.3}$$

The search space is expanded by an approximate solution x of the linear equation

$$(I - z_k z_k^*)(A - \mu_k B)(I - q_k q_k^*)x = -r_k. \tag{2.4}$$

This equation is called the Jacobi-Davidson correction equation. Here z_k is the vector $z_k = (\kappa_0 A - \kappa_1 B)q_k$, with e.g., $\kappa_0 = (1 - |\tau|^2)^{-1/2}$ and $\kappa_1 = -\tau(1 - |\tau|^2)^{-1/2}$ (harmonic Petrov value approach [5]). The test space is expanded with the vector $w_k = (\kappa_0 A - \kappa_1 B)x$. This procedure is repeated until $||r_k||$ is small enough.

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)

$$AQ_k = Z_k S_k, \quad BQ_k = Z_k T_k, \tag{2.5}$$

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 is computed already

$$AQ_k = Z_k S_k, \quad BQ_k = Z_k T_k. \tag{2.6}$$

Now one would like to compute the next Schur vector q_{k+1}. It can be shown that this Schur vector satisfies $Q_k^* q_{k+1} = 0$ and

$$(I - Z_k Z_k^*)(A - \mu_k B)(I - Q_k Q_k^*)q_{k+1} = 0. \tag{2.7}$$

Note that this is again a generalized eigenvalue problem, and therefore it can be solved using the JD method. Now the matrices V and W 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

$$W^*(I - Z_k Z_k^*)A(I - Q_k Q_k^*)Vu = \mu_k W^*(I - Z_k Z_k^*)B(I - Q_k Q_k^*)Vu. \tag{2.8}$$

In order to solve this generalized eigenvalue problem using the JD method, one needs to solve the JD correction equation, which can in this case be written as

$$(I - zz^*)(I - Z_k Z_k^*)(A - \mu_k B)(I - Q_k Q_k^*)(I - qq^*)v = -r, \tag{2.9}$$

with $Q_k^* v = 0$ and $q^* v = 0$, and the residual vector r given by $r = (I - Z_k Z_k^*)(A - \mu_k B)(I - Q_k Q_k^*)q$. When this equation 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 only approximately. In [5], 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

$$||r_i||_2 < 2^{-j}||r_0||_2, \tag{2.10}$$

where r_i is the ith residual vector of the Krylov subspace method, and j the iteration number of the JDQZ step. In [5] it is shown that this choice leads to an efficient method.

If P is a preconditioner of the matrix $(A - \mu_k B)$, then the correction equation can be preconditioned as follows.

$$(I - \frac{\tilde{z}_k q_k^*}{q_k^* \tilde{z}_k})P^{-1}(A - \mu_k B)(I - q_k q_k^*)v = -\tilde{r}_k, \tag{2.11}$$

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

In each step of the JDQZ method, a small sized, projected, eigenvalue problem is solved. From the computed eigenvalues of this small sized problem, the so called (harmonic) Petrov values, one has to select the (harmonic) Petrov value that is to be the new approximation of the wanted eigenvalue of the original full sized eigenvalue problem.

3. A JDQZ Method for Real Matrix Pencils.

3.1. The RJDQZ Algorithm. A generalized partial real Schur form of the matrix pencil (A, B) is a decomposition of the following form

$$AQ_k = Z_k S_k, \;\; BQ_k = Z_k T_k, \tag{3.1}$$

where now Q_k and Z_k are *real* matrices with orthogonal 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 [2] the author proposes (in Algorithm 3.1) an adaption of the JDQR algorithm 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 deflated eigenvalue problem (2.7), 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 this procedure, the search and test space do not have to be real, and therefore the Petrov values (being the eigenvalues of the projected eigenvalue problem) do not have to appear in complex conjugated pairs. In this case it is not clear how to decide when a converged Petrov value is real or not, especially when the imaginary part is small.

The way around this problem is to keep the search and test space real. Keeping the search and test space real 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

$$W^* AVu = \mu W^* BVu. \tag{3.2}$$

Since the projected matrices W^*AV and W^*BV are real, the eigenvalues are either real or form a complex conjugated pair. From these eigenvalues one eigenvalue 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 (3.2), a sorted real generalized Schur form is computed. How this form is computed in an efficient and stable way can be found in [2, 10, 12] for the standard eigenvalue problem and in [9, 11, 13] for the generalized eigenvalue problem.

In the JDQZ method, the search space is expanded by an approximate solution of the correction equation

$$(I - zz^*)(I - Z_k Z_k^*)(A - \mu B)(I - Q_k Q_k^*)(I - qq^*)v = -r, \tag{3.3}$$

where $q = Vu$ and $z = (\kappa_0 A - \kappa_1 B)q$, and the residual vector r is given by $r = (I - Z_k Z_k^*)(A - \mu_k B)(I - Q_k Q_k^*)q$. Assume that a Krylov subspace solver is used for an approximate solution. Then it follows that if the current approximation μ is real, then the Krylov subspace build by the solver will also be real, and therefore also to 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.

If now the selected Petrov value μ has non-zero imaginary part, then it is useful to think of the space $U = \text{span}\{V\tilde{u}, \overline{V\tilde{u}}\}$ as an approximation to a two dimensional real generalized invariant subspace. The 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 $\text{span}\{\Re(v), \Im(v)\}$, which contains the vector v.

One could also, instead of solving the correction equation in complex arithmetic, solve a generalized real Sylvester equation for the residual of the two dimensional invariant subspace U. Let X denote an orthonormal basis of U and let Y denote an orthonormal basis of $\text{span}\{BX\}$. Then the Sylvester equation for the update P to X is given by

$$\tilde{A}P - \tilde{B}PM = -(I - YY^*)AX, \tag{3.4}$$

where the matrix M is given by

$$M = (Y^*BX)^{-1}Y^*AX. \tag{3.5}$$

and where the matrices $\tilde{A}$ and $\tilde{B}$ are the deflated matrices

$$\tilde{A} = (I - YY^*)(I - Z_k Z_k^*)A(I - Q_k Q_k^*)(I - XX^*), \tag{3.6}$$
$$\tilde{B} = (I - YY^*)(I - Z_k Z_k^*)B(I - Q_k Q_k^*)(I - XX^*). \tag{3.7}$$

Solving the Sylvester equation is not a straight forward task, certainly if an available preconditioner is to be used. In addition, solving the Sylvester equation will not be much cheaper than solving the JD correction equation in complex arithmetic. Therefore, in this paper, it is chosen to solve the JD correction equation. (It is thinkable that in some cases solving the Sylvester equation may be more numerically stable. This is, however, an issue for further research and is not in the scope of this paper.)

Remark 1: if the selected Petrov value has non-zero imaginary part, then there must also be a Petrov value $\bar{\mu}$. Assuming the target τ is 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 does not matter 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 4.3 for more details).

Remark 2: note that if the target value τ has a non-zero imaginary part then κ_1 (see below equation (2.4)) 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 many of its advantages, although keeping the search space real by expanding it with a two dimensional real space when necessary might still accelerated the convergence of the JD method.

3.2. RJDQZ versus JDQZ.

3.2.1. Differences in Memory Costs. The search and test space for the JDQZ method grow with one complex dimension in each iteration. For the RJDQZ, the search and test space grow with one real dimension if the selected Ritz value is real or with two real dimensions if the selected Ritz value appears in a complex conjugated 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 at most the same, and probably less. In the numerical experiments it will be shown that the larger subspace in the RJDQZ method is beneficial for the convergence.

3.2.2. Differences in Computational Costs.

- The correction equation: When in the RJDQZ method a real Petrov value is selected, the correction equation can be solved using real arithmetic. 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.
- The projected eigenproblem: In the RJDQZ method one needs to compute the real Schur form of real projected eigenproblems, but these may twice as large as the complex projected eigenproblems that appear in the JDQZ method. Assume that that computing a Schur form costs $\mathcal{O}(n^3)$ operations [7] 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 real eigenvalue problem costs about twice as much as computing the Schur form of a complex eigenvalue problem that is twice as small. This is also confirmed by limited experiments in Matlab using the flops count (not reported in this paper).
- Orthogonalization: The only point where the real variant of the JDQZ method may be computationally more expensive than the original JDQZ, is in the orthogonalization procedures. One has to compare these two cases:
  - Orthogonalize a complex vector x against k other complex vectors. This requires $4k$ real innerproducts for the projection plus 2 real innerproducts for the scaling.
  - Orthogonalize two real vectors a and b against $2k$ other real vectors. This requires $2k$ innerproducts for projecting the first vector a plus one innerproduct for scaling. For the next vector b, $2k + 1$ innerproducts

are needed for the projection plus one for the scaling. This adds up to $4k + 3$ innerproducts.

In the worst case the real variant of JDQZ will cost about the work of one innerproduct and one (low dimensional) Schur decomposition more than the original version of JDQZ.

4. Numerical Comparison. The RJDQZ method is compared with two variants of the JDQZ method: the original JDQZ method as described in [5] and a variant that we denote by JDQZd, which is the method described for the standard eigenvalue problem in [2]. This method proceeds just as the original JDQZ method, except when has computed an eigenvalue with non-zero imaginary part. In that case not only the corresponding eigenvector is deflated but also the complex conjugated eigenvector.

4.1. 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 A.1-A.3. The cpu-time results are included, but note that these are not always very reliable.

4.2. Results for the Real JDQR Method. The ideas presented in this paper are easily incorporated in a QR version of the algorithm. In this section, numerical results for the QR version are reported. For all the results presented in this section, the correction equation is solved using at most 10 iterations of the BiCGSTAB method or the GMRES method. Also note that no restart strategy is used, 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. Six eigenvalues with largest real part are computed. The preconditioner that is used for the correction equations is ILU(A-7I,1e-3). In Table 4.1 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 get 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 needed less iterations to build this search space. Note that if only real eigenvalues are computed, the JDQR and JDQRd method perform exactly the same steps. Therefore the results for the RJDQRd 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 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 ILU(A,1e-3). Seven eigenvalues with largest real part are computed. Three of the computed eigenvalues are real and the other four are formed

TABLE 4.1
Results

	BiCGSTAB			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
BWF782	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

by two complex conjugated pairs. Observe that again the RJDQR method needs less 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 bellow in Fig. 4.1. 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 less 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 less matrix-vector products.

-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

FIG. 4.1. *The upper left 10 by 10 block of the matrix CC100.*

TABLE 4.2
Ritz value selection

-1.2973 - 3.1097i	-2.0424	-1.4899 + 3.1120i	-2.1344
-1.4131 - 3.0933i	-2.2331	-1.4982 + 3.1167i	-2.6077
-1.4598 - 3.1031i	-2.6959	-1.5003 + 3.1212i	-2.6250
-1.4822 - 3.1094i	-2.9376	-1.5001 + 3.1222i	-3.2437
-1.4918 - 3.1144i	-3.2460	-1.5000 + 3.1225i	-3.7005
-1.4941 + 3.1095i	-1.9109	-1.5000 + 3.1225i	-4.8570
-1.4976 - 3.1188i	-3.5720	-1.5000 + 3.1225i	-6.2045
-1.4992 - 3.1206i	-3.8480	-1.5000 + 3.1225i	-7.0421
-1.4999 - 3.1218i	-4.1691	-1.5000 + 3.1225i	-8.5431
-1.5000 - 3.1223i	-4.5287	-1.5000 + 3.1225i	-10.1098
-1.5000 - 3.1225i	-5.0167		
-1.4999 + 3.1227i	-3.3964		
-1.5000 + 3.1225i	-4.4841		
-1.5000 + 3.1225i	-5.6165		
-1.5000 + 3.1225i	-6.5374		
-1.5000 + 3.1225i	-7.4940		
-1.5000 + 3.1225i	-8.7653		
-1.5000 + 3.1225i	-10.0890		

4.3. Ritz value selection. A problem from which the Jacobi-Davidson method can suffer is the following. 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 have to 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. This is illustrated in Table 4.2. In the first two columns (a part of) the convergence history of the JD method for a complex eigenvalue is given. The first column contains the selected Ritz values, the second column contains the 10log of the residual. First the iterations converge to the eigenvalue with negative imaginary part, but after a number of iterations the method selects the Ritz value with positive imaginary part. Note that when this happens the residual becomes larger.

In the two last columns of the table, the convergence history of the RJDQR method for the same eigenvalues with the same initial approximation is given.

4.4. 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 using at most 10 iterations of the BiCGSTAB method or the GMRES method. Also note that again no restart strategy is used.

The generalized eigenvalue problem BWF782 (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 corresponding 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 ILU(A,1e-3).

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 [3, 4]. 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 ILU(A,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 less 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 differential equation

$$y'' + \mu^2 y = 0, \tag{4.1}$$

with boundary conditions

$$y(0) = 0 \text{ and } y'(0) + 0.001y'(1) = 0. \tag{4.2}$$

Eigenvalues and corresponding eigenvectors with largest real part are computed. The matrix A is non-symmetric and B is symmetric positive semi definite. Six eigenvalues that are computed with largest real part. The preconditioner that is used for the correction equations is LU(A).

For this matrix pencil, a similar conclusion as for the CC100 matrix is obtained. The computed eigenvalues are all complex, but still the RJDQZ method needs less iterations and less matrix-vector products than the JDQZ method.

5. 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 numerical experiments, the RJDQR and RJDQZ variants needed less iterations, less 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 is due to two reasons: first, the method uses real arithmetic where possible, which results in less (real) matrix-vector products, and second, the search space is usually larger in dimension (while it does not use more storage) which accelerates the convergence, resulting in less iterations.

Appendix A. Matlab-style code for RJDQZ method. In the Figures A.1-A.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 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 [5].

REFERENCES

[1] J. Brandts, *The Riccati method for eigenvalues and invariant subspaces of matrices with inexpensive action*, Linear Algebra and Applications **358** (2003), 333–363.

[2] J. Brandts, *Matlab code for sorted real Schur forms*, Numerical Linear Algebra with Applications, **9** (2002), 249-261.

[3] T. L. Van Noorden, *A Jacobi-Davidson continuation code for systems of differential-algebraic equations*, in preparation.

[4] 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** (1997), 195–232.

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

[6] D. Bindel, J. W. Demmel, and M. J. Friedman, *Continuation of Invariant Subspaces for Large and Sparse Bifurcations Problems*, online proceedings of SIAM Conference on Applied Linear Algebra, Williamsburg (2003).

[7] G. H. Golub and C. F. van Loan, *Matrix Computations* 2nd Ed. Baltimore: The John Hopkins University Press (1989).

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

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

[10] Z. Bai and J. W. Demmel, *On swapping diagonal blocks in real Schur form*, Linear Algebra Appl. **186** (1993), 73–95.

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

[12] A. Bojanczyk and P. Van Dooren, *Reordering diagonal blocks in Real Schur form*, in Linear Algebra for Large-Scale and Real-Time Applications, M. Moonen, G. H. Golub and B. De Moor, ed., Kluwer Academic Publishers, 1993.

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

```
function [Q, Z, R_A, R_B] =RJDQZ(A, B, K, τ, v_0, ε, k_max, j_min, j_max)
Q = []; Z = []; R_A = []; Y = []; H = [];
V = []; V_A = []; V_B = []; W = []; M_A = []; M_B = [];
γ = sqrt(1 + |τ|^2); α_0 = τ/γ; β_0 = 1/γ; 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, complex] = realqzsort(τ, M_A, M_B);
    j = j + 1; found= 1;
    while found
        if complex
            α = S_A(1 : 2, 1 : 2); β = S_B(1 : 2, 1 : 2);
            q = V U_R(:, 1 : 2); z = W U_L(:, 1 : 2);
            r_A = V_A U_R(:, 1 : 2); [r_A, s_A] = mgs(Z, r_A);
            r_B = V_B U_R(:, 1 : 2); [r_B, s_B] = mgs(Z, r_B);
            r = r_A/α − r_B/β;
            found= (||r|| < ε) and (j > 1|k = k_max − 1);
            if ||r|| > ε
                [U_L, U_R, S_A, S_B] = qz(α, β);
                α = S_A(1, 1); β = S_B(1, 1);
                q = q U_R(:, 1); z = z U_L(:, 1);
                r_A = r_A U_R(:, 1); [r_A, s_A] = mgs(Z, r_A);
                r_B = r_B U_R(:, 1); [r_B, s_B] = mgs(Z, r_B);
                r = r_A/α − r_B/β;
            end
            y = K^{-1} z;
            Q~ = [Q, q]; Y~ = [Y, y]; Z~ = [Z, z];
            H~ = [H, Q* y; q* Y, q* y];
        else
            α = S_A(1, 1); β = S_B(1, 1); q = V U_R(:, 1);
            z = W U_L(:, 1); y = K^{-1} z;
            r = V_A U_R(:, 1); [r, s_A] = mgs(Z, r);
            r_B = V_B U_R(:, 1); [r_B, s_B] = mgs(Z, r_B);
            r = r/α − r_B/β;
            found= (||r|| < ε) and (j > 1|k = k_max − 1);
            Q~ = [Q, q]; Y~ = [Y, y]; Z~ = [Z, z];
            H~ = [H, Q* y; q* Y, q* y];
        end
        input Found and restart
    end
end
```

FIG. A.1. *The RJDQZ method.*

```
if isreal(v)
        v = mgs(V,v); v = v/||v||; v_A = Av; v_B = Bv;
        w = (β_0 v_A − α_0 v_B); w = mgs(Z,w);
        w = 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
        ṽ = [real(v), imag(v)];
        for i = 1 : 2
                v = ṽ(:,i); v = mgs(V,v); v = v/||v||; v_A = Av; v_B = Bv;
                w = (β_0 v_A − α_0 v_B); w = mgs(Z,w);
                w = 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
```

FIG. A.2. *Expand*

```
if found
        Q = Q̃; Z = Z̃;
        R_A = [R_A, s_A; zeros(1 + complex, k), α];
        R_B = [R_B, s_B; zeros(1 + complex, k), β];
        k = k + 1 + complex; if k ≥ k_max, break; end
        Y = Ỹ; H = H̃;
        J = [2 + complex : j]; j = j − 1 − complex;
        V = V U_R(:,J); V_A = V_A U_R(:,J); V_B = V_B U_R(:,J);
        W = W U_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, complex] = realqzsort(τ, M_A, M_B);
elseif j ≥ j_max
        j = j_min; J = [1 : j];
        V = V U_R(:,J); V_A = V_A U_R(:,J); V_B = V_B U_R(:,J);
        W = W U_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
```

FIG. A.3. *Found and restart*