A Jacobi-Davidson method for solving complex symmetric eigenvalue problems

SIAM J. SCI. COMPUT.
Vol. 25, No. 5, pp. 1655–1673

A JACOBI–DAVIDSON METHOD FOR SOLVING COMPLEX SYMMETRIC EIGENVALUE PROBLEMS*

PETER ARBENZ† AND MICHIEL E. HOCHSTENBACH‡

Abstract. We discuss variants of the Jacobi–Davidson method for solving the generalized complex symmetric eigenvalue problem. The Jacobi–Davidson algorithm can be considered as an accelerated inexact Rayleigh quotient iteration. We show that it is appropriate to replace the Euclidean inner product in $\mathbb{C}^n$ with an indefinite inner product. The Rayleigh quotient based on this indefinite inner product leads to an asymptotically cubically convergent Rayleigh quotient iteration. Advantages of the method are illustrated by numerical examples. We deal with problems from electromagnetics that require the computation of interior eigenvalues. The main drawback that we experience in these particular examples is the lack of efficient preconditioners.

Key words. generalized complex symmetric eigenvalue problem, interior eigenvalues, Jacobi–Davidson algorithm, Rayleigh quotient iteration, cubic convergence

AMS subject classifications. 65F15, 65F50

DOI. 10.1137/S1064827502410992

1. Introduction.

In this paper we consider variants of the Jacobi–Davidson (JD) algorithm [27] for computing a few eigenpairs of

$$A\mathbf{x} = \lambda\mathbf{x}, \tag{1.1}$$

where the large, sparse matrix A is *complex symmetric*: $A = A^T \in \mathbb{C}^{n\times n}$, where A^T denotes the transpose of A. Eigenvalue problems of this type, and of the related generalized complex symmetric eigenvalue problem

$$A\mathbf{x} = \lambda B\mathbf{x}, \qquad B \text{ invertible}, \tag{1.2}$$

where both A and B are complex symmetric, are becoming increasingly important in applications, most notably in the field of electromagnetic simulations. High-quality particle accelerators can be modeled by the time-independent Maxwell equations, assuming perfectly conducting cavity walls. This approach leads to a generalized real symmetric eigenvalue problem [2]. However, in cases where the damping of higher modes is more important than the high efficiency of a cavity, and for cavities with ferrite inserts for tuning purposes, the currents produced in the walls or in the ferrite lead to a damping of the eigenmodes. In this situation these surfaces are treated as lossy material, which introduces a complex permittivity that in turn leads to complex symmetric matrices in the form of (1.1) or (1.2).

Open cavities are often modeled on bounded domains. Lossy perfectly matched layers (PMLs) along the boundary are introduced to prevent reflection of waves. PMLs, also called absorbing boundary conditions, are again modeled by complex

*Received by the editors July 12, 2002; accepted for publication (in revised form) September 15, 2003; published electronically March 5, 2004.
http://www.siam.org/journals/sisc/25-5/41099.html

†Institute of Computational Science, Swiss Federal Institute of Technology (ETH), CH-8092 Zürich, Switzerland (arbenz@inf.ethz.ch). Some of the work of this author was done during his visit to the Department of Mathematics at Utrecht University.

‡Department of Mathematics, Utrecht University, NL-3508 TA Utrecht, The Netherlands. Present address: Department of Mathematics, Case Western Reserve University, 10900 Euclid Avenue, Cleveland, OH 44106 (hochstenbach@cwru.edu).

permittivities [32]. The PML scheme has the potential to extend the range of applications for these eigenvalue solvers to the wide field of antenna design.

Notice that complex symmetric matrices are not Hermitian. So, they do not possess the favorable properties of Hermitian matrices. In particular, complex symmetric matrices have complex eigenvalues in general and can be arbitrarily nonnormal. In fact, *every* matrix is similar to a complex symmetric matrix [10, 16], whence it may be arbitrarily difficult to solve (1.1) or (1.2), respectively. Nevertheless, complex symmetric matrices do have special properties. If $\mathbf{x}$ is a right eigenvector of A, $A\mathbf{x} = \lambda\mathbf{x}$, then it is also a left eigenvector, in the sense that $\mathbf{x}^T A = \lambda\mathbf{x}^T$. Eigenvectors $\mathbf{x}, \mathbf{y}$ corresponding to different eigenvalues $\lambda \neq \mu$ are *complex orthogonal*; i.e., they satisfy $(\mathbf{x}, \mathbf{y})_T = 0$, where

$$(\mathbf{x}, \mathbf{y})_T := \mathbf{y}^T\mathbf{x} \tag{1.3}$$

is called an *indefinite inner product* [12]. If A is diagonalizable, then the diagonalization can be realized by a complex orthogonal matrix Q, $Q^TQ = I$ [10, 16]. A vector $\mathbf{x}$ is called *quasi-null* if $(\mathbf{x}, \mathbf{x})_T = 0$.

When treating the generalized eigenvalue problem (1.2), it is natural to use the indefinite bilinear form

$$[\mathbf{x}, \mathbf{y}]_T := (\mathbf{x}, B\mathbf{y})_T = \mathbf{y}^T B\mathbf{x}. \tag{1.4}$$

The matrix $B^{-1}A$ is then complex symmetric with respect to $[\mathbf{x}, \mathbf{y}]_T$, as A is complex symmetric with respect to $(\mathbf{x}, \mathbf{y})_T$. We therefore restrict ourselves to the special eigenvalue problem (1.1) whenever there is no loss of generality. The numerical examples that we will discuss later are all generalized eigenvalue problems of the form (1.2).

A number of algorithms have been designed for solving complex symmetric linear systems of equations. Van der Vorst and Melissen [33] modified the biconjugate gradient algorithm to obtain the complex conjugate gradient algorithm COCG. The crucial idea here is to set the initial shadow vector equal to the initial residual. (If one works with the Euclidean inner product, the shadow vector has to be the complex conjugate of the initial residual; see [33].) With regard to the relation between right and left eigenvectors mentioned before, this choice of the shadow vector is very natural. Freund used the same idea to adapt the quasi-minimal residual (QMR) algorithm to the complex symmetric case [9]. In COCG and QMR, the same Krylov subspaces are generated. However, the approximate solutions are extracted differently from these subspaces. A comparison of the two methods can be found in [11], where the authors are in favor of QMR for its smooth behavior.

A few algorithms for solving complex symmetric eigenvalue problems have been proposed. Eberlein adapted the classical Jacobi algorithm (for full matrices). Cullum and Willoughby [6, Chapter 6] proposed a Lanczos-type eigensolver employing the bilinear form (1.3). The same authors suggested a complex symmetric tridiagonal QR algorithm [7]. Recently, Luk and Qiao [20] introduced a fast $\mathcal{O}(n^2 \log n)$ eigensolver for complex Hankel matrices that is based on the works of Cullum and Willoughby and on the fast Fourier transform.

In this paper we present a JD-type algorithm for computing a few eigenpairs of a complex symmetric matrix that exploits this structure. For the original JD algorithm see [27, 25, 8]. Our method is a modification of the two-sided JD algorithm [15] with properly chosen left and right vectors, using the bilinear form (1.3). In contrast to the complex symmetric methods mentioned before, our JD algorithm can be transcribed quite easily into a solver for the generalized eigenvalue problem (1.2).

The paper is organized as follows. In section 2 we show that for complex symmetric eigenvalue problems it is a good idea to replace the usual Rayleigh quotient with the Rayleigh quotient based on the indefinite inner product (1.3). In section 3 we adapt a variant of the two-sided JD algorithm to this problem and discuss its application to the generalized complex symmetric eigenvalue problem. The convergence behavior of exact and inexact variants of the JD algorithm is investigated in section 4. Numerical experiments are presented in section 5. A discussion and some conclusions can be found in section 6.

2. A Rayleigh quotient for complex symmetric matrices. Let us first introduce some notation. Throughout the paper, λ denotes a simple eigenvalue of the complex symmetric $n \times n$ matrix A, with $\mathbf{x}$ its corresponding eigenvector. Since λ is simple, it has a finite condition $\kappa(\lambda)$. Because

$$\infty > \kappa(\lambda) = |\mathbf{x}^T\mathbf{x}|^{-1} = |(\mathbf{x},\mathbf{x})_T|^{-1}, \tag{2.1}$$

an eigenvector corresponding to a simple eigenvalue is not quasi-null, whence we can assume that it is "normalized" such that $(\mathbf{x},\mathbf{x})_T = 1$. Let $\mathbf{u} \approx \mathbf{x}$ be an approximate eigenvector. If $\mathbf{u}$ is close enough to $\mathbf{x}$, then $\mathbf{u}$ also is not quasi-null, and we "normalize" $\mathbf{u}$ such that $(\mathbf{u},\mathbf{u})_T = 1$.

Given $\mathbf{u}$, the corresponding eigenvalue is usually approximated by the Rayleigh quotient

$$\rho = \rho(\mathbf{u}) := \frac{\mathbf{u}^*A\mathbf{u}}{\mathbf{u}^*\mathbf{u}}. \tag{2.2}$$

Alternatively, with regard to the indefinite inner product (1.3), we also can define the Rayleigh quotient by

$$\theta = \theta(\mathbf{u}) := \frac{\mathbf{u}^TA\mathbf{u}}{\mathbf{u}^T\mathbf{u}}. \tag{2.3}$$

One may check that, for complex symmetric A, the latter definition has the desirable property (cf. [23, p. 688] and [15])

$$\theta(\mathbf{u}) \text{ is stationary} \iff \mathbf{u} \text{ is an eigenvector of } A. \tag{2.4}$$

(Recall that *stationary* means that all directional derivatives are zero.) By writing

$$\mathbf{u} = (\mathbf{x}\mathbf{x}^T)\mathbf{u} + \left(I - \mathbf{x}\mathbf{x}^T\right)\mathbf{u},$$

we see that $\mathbf{u}$ can be written in the form

$$\mathbf{u} = \alpha\mathbf{x} + \delta\mathbf{d}, \tag{2.5}$$

where $\alpha^2 + \delta^2 = 1$, $(\mathbf{d},\mathbf{d})_T = 1$, and $\mathbf{x} \perp_T \mathbf{d} = 0$. Direct computation shows that

$$\lambda - \theta = \delta^2\,\mathbf{d}^T(\lambda I - A)\mathbf{d}.$$

So, we conclude that

$$|\lambda - \theta| = \mathcal{O}(\delta^2), \tag{2.6}$$

while $|\lambda - \rho|$ is in general "only" $\mathcal{O}(\delta)$. (The reason for the last statement is that in general the eigenvectors are no stationary points of $\rho(\mathbf{u})$.) Therefore, the Rayleigh quotient θ is asymptotically (i.e., when $\mathbf{u}$ converges to $\mathbf{x}$) more accurate than the usual Rayleigh quotient ρ.

3. JD for complex symmetric matrices. In this section we introduce a JD method for complex symmetric matrices that we denominate JDCS. A subspace method typically consists of two ingredients: *extraction* and *expansion*. Suppose we have a k-dimensional search space $\mathcal{U}$, where typically $k \ll n$. The crucial observation is that if $\mathcal{U}$ is the search space for the (right) eigenvector, then, with regard to the indefinite inner product (1.3), $\mathcal{U}$ forms a search space for the left eigenvector of equal quality. So, the fundamental difference with the two-sided JD algorithm [15] is that, as we build up a right search space (i.e., a search space for the right eigenvector), we get a good left search space for free. We do not have to (approximately) solve a left correction equation as in the two-sided JD algorithm. This saves roughly half the computational and storage costs.

3.1. Extraction. We first study the subspace extraction for complex symmetric matrices. Given a search space $\mathcal{U}$, we would like to get an approximate eigenpair $(\theta, \mathbf{u})$, where $\mathbf{u} \in \mathcal{U}$. Let the columns of U form a basis for $\mathcal{U}$, and define the *residual* $\mathbf{r}$ by

$$\mathbf{r} := A\mathbf{u} - \theta\mathbf{u}.$$

In view of (2.4) and (2.6), we take, instead of the usual Ritz–Galerkin condition on the residual $\mathbf{r} = A\mathbf{u} - \theta\mathbf{u} \perp \mathcal{U}$, the same condition but with respect to the indefinite inner product (1.3)

$$\mathbf{r} = A\mathbf{u} - \theta\mathbf{u} \perp_T \mathcal{U}. \tag{3.1}$$

Writing $\mathbf{u} = U\mathbf{c}$, $\mathbf{c} \in \mathbb{C}^k$, we find that $(\theta, \mathbf{c})$ must be a solution of the projected eigenproblem

$$U^T AU\mathbf{c} = \theta\, U^T U\mathbf{c}. \tag{3.2}$$

Thus, a *Ritz pair* $(\theta, \mathbf{u}) = (\theta, U\mathbf{c})$ is obtained by backtransforming an eigenpair of the *projected pencil* $(U^T AU, U^T U)$. In particular, if $(\theta, \mathbf{u})$ is a Ritz pair, we have

$$\theta = \theta(\mathbf{u}) := \frac{\mathbf{u}^T A\mathbf{u}}{\mathbf{u}^T\mathbf{u}} \qquad \text{and} \qquad \mathbf{r} \perp_T \mathbf{u}.$$

3.2. Expansion. Let us now examine the subspace expansion for JDCS: Having an approximate eigenpair $(\theta, \mathbf{u})$ to $(\lambda, \mathbf{x})$, how do we expand the search space $\mathcal{U}$ in order to get an even better approximation? JD-type methods look for a correction $\mathbf{s}$ such that

$$A(\mathbf{u} + \mathbf{s}) = \lambda(\mathbf{u} + \mathbf{s}), \tag{3.3}$$

i.e., such that $\mathbf{u} + \mathbf{s}$ is a multiple of the eigenvector $\mathbf{x}$. This equation can be rewritten in two different ways, depending on whether we wish that $\mathbf{s} \perp_T \mathbf{u}$ or $\mathbf{s} \perp \mathbf{u}$. Let us start with $\mathbf{s} \perp_T \mathbf{u}$. Write (3.3) as

$$(A - \theta I)\mathbf{s} = -(A - \theta I)\mathbf{u} + (\lambda - \theta)\mathbf{u} + (\lambda - \theta)\mathbf{s}. \tag{3.4}$$

In view of (2.6), the term $(\lambda - \theta)\mathbf{s}$ is asymptotically of third order. When we neglect this term, we still have cubic convergence; see Theorem 4.1. During the process, λ and hence also $(\lambda - \theta)\mathbf{u}$ are unknown. Therefore it is interesting to consider the projection of this equation that maps $\mathbf{u}$ to $\mathbf{0}$ and keeps $\mathbf{r} = (A - \theta I)\mathbf{u}$ fixed. Because $\mathbf{r} \perp_T \mathbf{u}$, this

projector is $I - \mathbf{u}\mathbf{u}^T$, the oblique projection onto $\mathbf{u}^{\perp_T}$. The result of projecting (3.4) is

$$\left(I - \mathbf{u}\mathbf{u}^T\right)(A - \theta I)\mathbf{s} = -\mathbf{r}.$$

Using the constraint

$$\left(I - \mathbf{u}\mathbf{u}^T\right)\mathbf{s} = \mathbf{s},$$

we derive the first possibility for the JDCS correction equation:

$$\left(I - \mathbf{u}\mathbf{u}^T\right)(A - \theta I)\left(I - \mathbf{u}\mathbf{u}^T\right)\mathbf{s} = -\mathbf{r}, \qquad \text{where } \mathbf{s} \perp_T \mathbf{u}. \tag{3.5}$$

The operator in this equation is complex symmetric. First, we try to solve (3.5) by a linear solver that is especially designed for complex symmetric systems, such as complex symmetric QMR [9], COCG [33], or CSYM [5]. We remark that CSYM is not a Krylov subspace method, which may explain the disappointing convergence behavior often experienced in practice; see [5]. Therefore, we do not use this algorithm for our numerical examples.

Second, we investigate the situation where we wish to have an orthogonal update $\mathbf{s} \perp \mathbf{u}$. We rewrite (3.3) as

$$(A - \theta I)\mathbf{s} = -(A - \rho I)\mathbf{u} + (\lambda - \rho)\mathbf{u} + (\lambda - \theta)\mathbf{s}.$$

Again neglecting the last term and noting that

$$\widetilde{\mathbf{r}} := (A - \rho I)\mathbf{u} \perp \mathbf{u},$$

this leads to an alternative JDCS correction equation:

$$\left(I - \frac{\mathbf{u}\mathbf{u}^*}{\mathbf{u}^*\mathbf{u}}\right)(A - \theta I)\left(I - \frac{\mathbf{u}\mathbf{u}^*}{\mathbf{u}^*\mathbf{u}}\right)\mathbf{s} = -\widetilde{\mathbf{r}}, \qquad \text{where } \mathbf{s} \perp \mathbf{u}. \tag{3.6}$$

Unless A is Hermitian, this operator does not have any particular properties; we could try to (approximately) solve the equation by, for instance, GMRES, standard QMR, or BiCGSTAB. In practice, a correction equation is often solved only *approximately* (or *inexactly*). The approximate solution is used to expand the search space $\mathcal{U}$; this is called *subspace acceleration*.

Next, we mention that JDCS can be viewed as an *accelerated inexact Newton method* for the eigenvalue problem. For the correction equation (3.6) such a result has been given in [28]. For the correction equation (3.5), we define

$$F(\mathbf{u}) = A\mathbf{u} - \frac{\mathbf{a}^T A\mathbf{u}}{\mathbf{a}^T\mathbf{u}}\mathbf{u};$$

then one may check that a Newton step $DF(\mathbf{u})\mathbf{s} = -F(\mathbf{u})$, with $\mathbf{a} = \mathbf{u}$, becomes

$$\left(I - \mathbf{u}\mathbf{u}^T\right)(A - \theta I)\mathbf{s} = -\mathbf{r}.$$

Algorithm 3.1 summarizes our algorithm JDCS as developed so far. In step 2, MGS-CS stands for any numerically stable form of Gram–Schmidt used to expand a complex orthogonal basis for the search space: the last column of U_k is a multiple of $(I - U_{k-1}U_{k-1}^T)s$, computed in a stable way. This implies that the matrix on the right-hand side of (3.2) is the identity, whence we only have to solve a standard eigenvalue problem in step 4. In step 5 and elsewhere in the paper, $\|\cdot\|$ denotes the Euclidean norm. In step 8 of the algorithm the correction equation (3.5) could be replaced with (3.6). The following three practical issues have been omitted in Algorithm 3.1 for ease of presentation:

ALGORITHM 3.1. **The JDCS algorithm for the computation of an eigenpair of a complex symmetric matrix closest to a target τ.**

Input: *a device to compute* $A\mathbf{x}$ *for arbitrary* $\mathbf{x}$*, a starting vector* $\mathbf{u}_1$*, and a tolerance* ε*.*

Output: *an approximation* $(\theta, \mathbf{u})$ *to an eigenpair satisfying* $\|A\mathbf{u} - \theta\mathbf{u}\| \leq \varepsilon\|\mathbf{u}\|$*.*

1. $s = \mathbf{u}_1$.
   for $k = 1, 2, \ldots$
2. $U_k = MGS\text{-}CS\ (U_{k-1}, \mathbf{s})$.
3. *Compute* k*th column of* $W_k = AU_k$.
   Compute k*th row and column of* $H_k = U_k^T W_k$.
4. *Compute the eigenpair* $(\theta, \mathbf{c})$ *of* $U_k^T AU_k$ *that is closest to the target* τ*, where* $(\mathbf{c}, \mathbf{c})_T = 1$.
5. $\mathbf{u} = U_k\mathbf{c}$.
6. $\mathbf{r} = (A - \theta I)\mathbf{u} \,/\, \|\mathbf{u}\| = (W_k\mathbf{c} - \theta\mathbf{u}) \,/\, \|\mathbf{u}\|$.
7. *Stop if* $\|\mathbf{r}\| \leq \varepsilon$.
8. *Solve (approximately) for* $\mathbf{s} \perp_T \mathbf{u}$
   $\left(I - \mathbf{u}\mathbf{u}^T\right)(A - \theta I)\left(I - \mathbf{u}\mathbf{u}^T\right)\mathbf{s} = -\mathbf{r}$.

(1) In our actual computations we replace the shift θ in the correction equation of step 8 in the first couple of iterations with a fixed target τ, which we know (or hope) is close to the desired eigenvalue. This is reasonable, as the correction equation, if solved exactly, amounts to one step of shift-and-invert. As the Ritz value θ initially is far from the desired eigenvalue, using θ as shift does not give any benefit. We switch the shift from τ to θ as soon as the residual $\|\mathbf{r}\|$ is below some threshold, such as 0.1.

(2) To restrict the memory consumption of our algorithm, we limit the dimension of the search space $\mathcal{U}$. If this limit is reached, we *restart*, i.e., we replace $\mathcal{U}$ with a given number of the "best" Ritz vectors contained in $\mathcal{U}$ (for instance, those with Rayleigh quotient closest to the target, or refined Ritz vectors; see section 3.3).

(3) If we need to compute several eigenpairs, we apply the algorithm repeatedly. Hereby, we use the search space of the previous iteration as our initial search space. Furthermore, the correction equation (3.5) is replaced with

$$(I - \widetilde{U}\widetilde{U}^T)(A - \theta I)(I - \widetilde{U}\widetilde{U}^T)\mathbf{s} = -\mathbf{r}, \qquad \widetilde{U}^T\mathbf{s} = \mathbf{0}. \tag{3.7}$$

Here, $\widetilde{U} = [\mathbf{u}, \mathbf{x}_1, \ldots]$, scaled such that $\widetilde{U}$ has complex orthogonal columns, contains, in addition to the Ritz vector $\mathbf{u}$, the eigenvectors $\mathbf{x}_i$ that have been computed previously. Notice that $\widetilde{U}$ may contain some further orthogonality constraints; see section 5.1.

Now we consider the correction equation for the generalized eigenproblem. In this case, (3.4) becomes

$$(A - \theta B)\mathbf{s} = -(A - \theta B)\mathbf{u} + (\lambda - \theta)B\mathbf{u} + (\lambda - \theta)B\mathbf{s}. \tag{3.8}$$

One may check that with the Galerkin condition $\mathbf{r} = A\mathbf{u} - \theta B\mathbf{u} \perp_T \mathcal{U}$, leading to

$$\theta = \frac{\mathbf{u}^T A\mathbf{u}}{\mathbf{u}^T B\mathbf{u}},$$

the last term on the right-hand side of (3.8) is of third order. The projector $I - B\mathbf{u}(\mathbf{u}^T B\mathbf{u})^{-1}\mathbf{u}^T$ annihilates the term $(\lambda - \theta)B\mathbf{u}$. So, when $\mathbf{u}$ is scaled such that $\mathbf{u}^T B\mathbf{u} = 1$, the correction equation for the generalized eigenvalue problem corresponding to (3.5) is

$$(I - B\mathbf{u}\mathbf{u}^T)(A - \theta B)(I - \mathbf{u}\mathbf{u}^T B)\mathbf{s} = -(A - \theta B)\mathbf{u}, \qquad \mathbf{s} \perp_T B\mathbf{u}. \tag{3.9}$$

Notice that the operator is complex symmetric. By analogous manipulations, (3.7) becomes

$$(I - B\widetilde{U}\widetilde{U}^T)(A - \theta B)(I - \widetilde{U}\widetilde{U}^T B)\mathbf{s} = -\mathbf{r}, \qquad \mathbf{s} \perp_T B\widetilde{U}, \tag{3.10}$$

with $\widetilde{U}^T B\widetilde{U} = I$.

3.3. Harmonic Ritz vectors. It is well known that Ritz–Galerkin extraction (see section 3.1) works out nicely for exterior eigenvalues but may give poor approximations to interior eigenvalues. For these eigenvalues, we can apply a harmonic Ritz approach, just as in the standard JD method [22, 4]. Suppose that we are interested in one or more interior eigenpairs near the target τ. One idea is to consider a ("complex symmetric") Galerkin condition on $(A - \tau I)^{-1}$:

$$(A - \tau I)^{-1}\widetilde{\mathbf{u}} - (\widetilde{\theta} - \tau)^{-1}\widetilde{\mathbf{u}} \perp_T \widetilde{\mathcal{U}}, \quad \widetilde{\mathbf{u}} \in \widetilde{\mathcal{U}}. \tag{3.11}$$

With $\widetilde{\mathcal{U}} := (A - \tau I)\mathcal{U}$ and $\widetilde{\mathbf{u}} = \widetilde{U}\widetilde{\mathbf{c}}$ this condition becomes

$$U^T(A - \tau I)^2 U\widetilde{\mathbf{c}} = (\widetilde{\theta} - \tau)U^T(A - \tau I)U\widetilde{\mathbf{c}}. \tag{3.12}$$

The solutions $(\widetilde{\theta}, U\widetilde{\mathbf{c}})$ to this small complex symmetric eigenvalue problem are called harmonic Ritz pairs. If we are to compute interior eigenvalues of A, then the common procedure is to replace the eigenvalue problem in step 4 of Algorithm 3.1 with (3.12) and extract the harmonic Ritz pair closest to the target value τ. We can multiply (3.12) from the left by $\widetilde{\mathbf{c}}^T$ to obtain [26]

$$((A - \tau I)U\widetilde{\mathbf{c}}, (A - \tau I)U\widetilde{\mathbf{c}})_T = (\widetilde{\theta} - \tau)((A - \tau I)U\widetilde{\mathbf{c}}, U\widetilde{\mathbf{c}})_T.$$

In contrast to the case where A is Hermitian, the expression on the left is not a residual norm, whence a small value of $\widetilde{\theta} - \tau$ does not necessarily imply that $U\widetilde{\mathbf{c}}$ is a good eigenvector approximation; the harmonic Ritz vector does not necessarily have a small residual norm.

Therefore, it is more promising to use the harmonic approach that is based on the usual Euclidean inner product. This approach leads to the generalized eigenproblem (see, for instance, [31, p. 296])

$$U^*(A - \tau B)^*(A - \tau B)U\widetilde{\mathbf{c}} = (\widetilde{\theta} - \tau)U^*(A - \tau B)^* BU\widetilde{\mathbf{c}}. \tag{3.13}$$

This extraction has the mathematical justification that

$$\|(A - \tau B)U\widetilde{\mathbf{c}}\| \leq |\widetilde{\theta} - \tau|\|BU\|,$$

but the reduced system (3.13) is not complex symmetric.

3.4. Refined Ritz vectors. A second approach to computing interior eigenvalues is through refined Ritz vectors. Let $(\theta, \mathbf{u})$ be a Ritz pair, i.e., a solution of (3.1). The Ritz value θ may "by coincidence" be close to an interior eigenvalue of A although the corresponding Ritz vector is a linear combination of eigenvectors that are not close to the particular eigenvector. In this situation, which is common when computing interior eigenvalues, the computed Ritz vectors are of no use as approximate eigenvectors in the correction equation. A remedy suggested in [17] (see also [31, p. 289]) is to *refine* the Ritz vectors. A refined Ritz vector is defined to be a solution of the problem

$$\text{minimize } \|A\widehat{\mathbf{x}} - \theta\widehat{\mathbf{x}}\| \quad \text{subject to} \quad \widehat{\mathbf{x}} \in \mathcal{U},\ \|\widehat{\mathbf{x}}\| = 1. \tag{3.14}$$

Let $\widehat{\mathbf{x}} = U\widehat{\mathbf{c}}$ be a solution of (3.14). Then $\widehat{\mathbf{c}}$ is a right singular vector corresponding to the smallest singular value of the matrix

$$(A - \theta I)U. \tag{3.15}$$

In order to extract a refined Ritz vector we modified steps 4 and 5 of Algorithm 3.1 in the following way:

4.	Compute the eigenpair $(\theta, \mathbf{c})$ of $U_k^T A U_k$ that is closest to the target τ. Determine a right singular vector $\widehat{\mathbf{c}}$ corresponding to the smallest singular value of $(A - \theta I)U$, where $(\widehat{\mathbf{c}}, \widehat{\mathbf{c}})_T = 1$.
5.	$\mathbf{u} = U_k\widehat{\mathbf{c}}, \quad \theta = \mathbf{u}^T A\mathbf{u}/\mathbf{u}^T\mathbf{u}$.

It is straightforward to adapt (3.14)–(3.15) to the generalized eigenvalue problem.

4. Convergence of (inexact) JD for complex symmetric matrices. The convergence theory developed in this section is an adaptation of the theory for the two-sided Rayleigh quotient and the two-sided JD in [15] to the complex symmetric situation.

When we solve either of the two correction equations (3.5) or (3.6) exactly, we find (see, e.g., [27])

$$\mathbf{s} = -\mathbf{u} + \alpha\,(A - \theta I)^{-1}\,\mathbf{u},$$

where α is such that $\mathbf{s} \perp_T \mathbf{u}$ or $\mathbf{s} \perp \mathbf{u}$. JDCS uses $\mathbf{s}$ to expand the search space $\mathcal{U}$. Since already $\mathbf{u} \in \mathcal{U}$, we get the same subspace expansion using $\widetilde{\mathbf{s}} = (A - \theta I)^{-1}\,\mathbf{u}$. Here we recognize a step of the Raleigh quotient iteration (RQI), and we conclude that *exact* JDCS (i.e., JDCS where we solve the correction equation exactly) can also be interpreted as (subspace) accelerated RQI.

Therefore, we first define an RQI for complex symmetric matrices and show that this RQI has asymptotically cubic convergence for eigenpairs of which the vector is not quasi-null; see Algorithm 4.2.

THEOREM 4.1 (locally cubic convergence of the RQI for complex symmetric matrices). *Suppose that* $\mathbf{u}_k = \alpha_k\mathbf{x} + \delta_k\mathbf{d}_k$ *(cf.* (2.5)*) converges to* $\mathbf{x}$*, where* $\mathbf{x}$ *is not quasi-null, as* $k \to \infty$*. Then* $\theta_k \to \lambda$*, and we have*

$$\delta_{k+1} = \mathcal{O}(\delta_k^3).$$

Proof. The proof goes along the same lines as the proof of [15, Theorem 3.1]; see also [23, p. 689]. The main argument is that

$$\mathbf{u}_{k+1} = \alpha_{k+1}(\mathbf{x} + \delta_k(\lambda - \theta_k)\,(A - \theta_k I)^{-1}\,\mathbf{d}_k).$$

ALGORITHM 4.2. **Rayleigh quotient iteration for complex symmetric matrices.**

Input: *An initial vector* $\mathbf{u}_1$*, not quasi-null.*
Output: *an eigenpair of* A *(or failure).*
for $k = 1, 2, \ldots$
1. *Compute* $\theta_k := \theta_k(\mathbf{u}_k) = \frac{\mathbf{u}_k^T A \mathbf{u}_k}{\mathbf{u}_k^T \mathbf{u}_k}$.
2. **If** $A - \theta_k I$ *is singular,* **then** *solve* $(A - \theta_k I)\mathbf{x} = 0$.
3. *Solve* $(A - \theta_k I)\mathbf{u}_{k+1} = \mathbf{u}_k$.
4. **If** $(\mathbf{u}_{k+1}, \mathbf{u}_{k+1})_T = 0$, **then** *method fails.*
   else *"normalize"* $\mathbf{u}_{k+1}$ *such that* $(\mathbf{u}_{k+1}, \mathbf{u}_{k+1})_T = 1$.

Now, a combination of (2.6) and the fact that $(A - \lambda I)^{-1}$ exists on $\mathbf{x}^{\perp_T}$ ($A - \lambda I : \mathbf{x}^{\perp_T} \to \mathbf{x}^{\perp_T}$ is a bijection) proves the theorem. ∎

Now, as a corollary, the asymptotically cubic convergence of JDCS follows. (See [30, p. 652] for a discussion about the term "asymptotic convergence" for subspace methods.) As noted in [15], the constant of $\mathcal{O}(\delta_k^3)$ may be considerable in practice, which reduces the significance of the cubic convergence.

Apart from this, JD- and RQI-type methods are in practice often very expensive when we accurately solve the linear systems occurring in the methods. We therefore consider inexact variants, where the linear systems are solved to a certain precision (minimal residual approach).

First consider the situation where we solve the linear system of the complex symmetric RQI method inexactly, by which we mean that we are satisfied with a $\mathbf{u}_{k+1}$ if

$$\|(A - \theta_k I)\mathbf{u}_{k+1} - \mathbf{u}_k\| \leq \xi < 1. \tag{4.1}$$

Notice that it may become increasingly difficult to satisfy (4.1) as θ_k tends to λ because $A - \theta_k I$ is almost singular.

The following two theorems can be proved by similar methods as in [15, Lemma 5.1, Theorems 5.2 and 5.3], exploiting the fact that the right eigenvector $\mathbf{x}$ is a left eigenvector as well.

THEOREM 4.2 (locally quadratic convergence of inexact RQI for complex symmetric matrices). *Let the iterates of the inexact RQI* $\mathbf{u}_k = \alpha_k \mathbf{x} + \delta_k \mathbf{d}_k$ *satisfy* (4.1) *with an accuracy* ξ *such that* $\xi \cdot |(\mathbf{x}, \mathbf{x})_T| < 1$. *Then*

$$\delta_{k+1} = \mathcal{O}(\delta_k^2).$$

Consider the situation where we inexactly solve the correction equation (3.5) or (3.6) of the complex symmetric JD method, by which we mean that we are satisfied with $\widetilde{\mathbf{s}} \perp_T \mathbf{u}_k$, where

$$\left\| \left(I - \mathbf{u}_k \mathbf{u}_k{}^T\right) (A - \theta I)\widetilde{\mathbf{s}} + \mathbf{r}_k \right\| \leq \eta \, \|\mathbf{r}_k\| \tag{4.2}$$

for some $0 < \eta < 1$.

THEOREM 4.3 (locally linear convergence of inexact JD for complex symmetric matrices). *Let* $\mathbf{u}_k = \alpha_k \mathbf{x} + \delta_k \mathbf{d}_k$ *be the iterates of inexact JDCS satisfying* (4.2). *Then*

$$\delta_{k+1} \leq \gamma \delta_k + \mathcal{O}(\delta_k^2),$$

where $\gamma = \eta \cdot \kappa((A - \lambda I)|_{\mathbf{x}^{\perp_T} \to \mathbf{x}^{\perp_T}})$.

We remark that for the variant where $\widetilde{\mathbf{s}} \perp \mathbf{u}_k$ we have a similar statement, now with $\gamma = \kappa((A - \lambda I)|_{\mathbf{x}^{\perp} \to \mathbf{x}^{\perp_T}})$. So, for η small enough, we expect linear convergence. However, in practice we may have one of the following situations:

- Although the condition number γ in Theorem 4.3 is bounded, it can be large, for instance, if the gap between λ and the other eigenvalues is small. This is a situation that we encounter in our numerical examples.
- We may choose η not small, say 0.5.
- For the correction equation, we may perform a fixed number of iterations with a linear solver, rather than focusing on a fixed norm reduction.

In all these cases, the previous theorem has little or nothing to say. Yet, we often experience linear convergence in practice.

As also discussed in [15], we cannot conclude from the theorems that inexact RQI is faster than inexact JD. The theorems only say something on the local, not the global, rate of convergence (note that JD has subspace acceleration); moreover, (4.2) is potentially easier to satisfy than (4.1), as $\left(I - \mathbf{u}_k \mathbf{u}_k{}^T\right)(A - \theta I)$ considered as a mapping from $\mathbf{u}^{\perp_T}$ into itself has a bounded inverse.

5. Numerical experiments. In this section we discuss the applicability of our algorithm to three test problems from electromagnetics. All tests have been executed with MATLAB 6.5 (Release 13).

5.1. A dispersive waveguide structure. We consider first the generalized eigenvalue problem `dwg961` that is available from the Matrix Market [21]. It originates from a finite element discretization of a waveguide problem with conductors of finite cross section in a lossy dielectric medium. The eigenvalue problem has the form

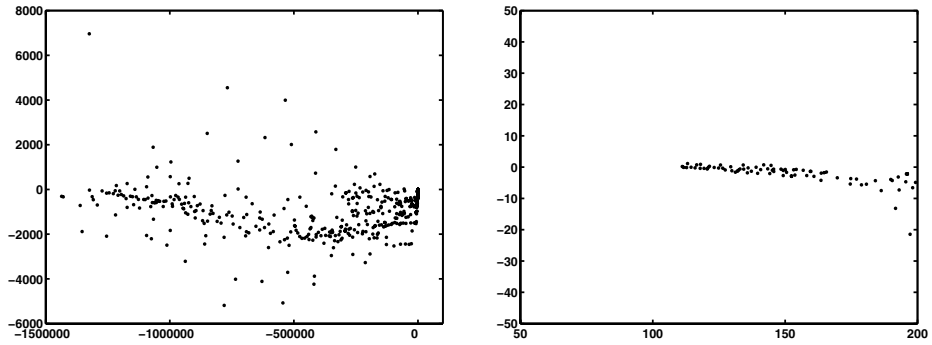
$$A\mathbf{x} = \begin{bmatrix} A_{11} & O \\ O & O \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \lambda \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \lambda B\mathbf{x}. \tag{5.1}$$

These matrix structures are obtained if the Maxwell equations are discretized by finite edge elements. The order of the overall problem is 961. The order of A_{11} is 705. The matrix B, as well as the submatrix A_{11}, is nonsingular. Thus, (5.1) has 256 zero eigenvalues. The corresponding eigenspace is

$$\mathcal{N}(A) = \mathcal{R}(Y), \qquad Y = \begin{bmatrix} O \\ I_{256} \end{bmatrix}.$$

Here, $\mathcal{N}(\cdot)$ and $\mathcal{R}(\cdot)$ denote nullspace and range of the respective matrices. Similarly, as in the real symmetric case [1] it may be advantageous to prevent the iterates from converging to eigenvectors corresponding to the zero eigenvalue by forcing them to be complex orthogonal to $\mathcal{R}(BY)$. Technically, this can be achieved by incorporating Y into the set of found eigenvectors $\widetilde{U}$ in the correction equation (3.10). In the examples discussed in this paper this precaution was unnecessary.

In Figure 5.1 the spectrum of the matrix pencil $(A; B)$ of (5.1) is plotted. In addition to the 256 zero eigenvalues, there are 362 eigenvalues with negative real part and 343 with positive real part. Although there is no eigenvalue with real part between -2500 and 100, the two sets are, in a relative sense, not well separated. From Figure 5.1(a) we see that the eigenvalues with positive real part are much more clustered than those with negative real part. The smallest of the former are about 0.5 to 1 apart. The largest eigenvalue (in modulus) is about $1.4 \cdot 10^6$. Thus, the relative gap is quite small and the condition number of the correction equation is about 10^6.



(a) (b)

FIG. 5.1. (a) *complete spectrum of* `dwg961` *and* (b) *portion of the spectrum close to* $\tau = 100$. *The plot shows imaginary versus real parts of the eigenvalues.*

We determine the six smallest eigenvalues with positive real part by the JD algorithm, Algorithm 3.1, for computing interior eigenvalues close to the shift $\tau = 100$ employing refined Ritz vectors. The practical considerations (1)–(3) mentioned after Algorithm 3.1 apply.

The efficient solution of the correction equation (3.10) requires preconditioning. We employ complex symmetric preconditioners of the form

$$(I - B\widetilde{U}\widetilde{U}^T)M(I - \widetilde{U}\widetilde{U}^T B),$$

where M approximates $A - \tau B$, and we hope that solving systems with M will be (much) cheaper than with $A - \tau B$. The computation of the preconditioned residual $\mathbf{t}$ from the residual $\mathbf{r}$ amounts to solving

$$(I - B\widetilde{U}\widetilde{U}^T)M(I - \widetilde{U}\widetilde{U}^T B)\mathbf{t} = \mathbf{r}, \qquad \widetilde{U}^T B\mathbf{t} = \mathbf{0}. \tag{5.2}$$

The solution of (5.2) is given by

$$\mathbf{t} = (I - M^{-1}B\widetilde{U}(\widetilde{U}^T BM^{-1}B\widetilde{U})^{-1}\widetilde{U}^T B)M^{-1}\mathbf{r}. \tag{5.3}$$

Note that we keep the preconditioner fixed throughout the computation. As we compute only a few close eigenvalues, a constant preconditioner turns out to be good enough. In the course of the computation, the quality of the preconditioner may deteriorate if many eigenpairs are desired. Then the preconditioner may have to be updated by techniques like those suggested in [29].

The special solvers discussed in the previous sections can be employed for solving (3.10) only if the preconditioner in (5.2), and thus M, is complex symmetric. We assume M to be of the form $M = LDL^T$, where D is a diagonal and L is a unit lower triangular matrix. We show experiments with the following:

- *LDL^T factorization preconditioning.* Here, $M = A - \tau B$. L and D are obtained by ordinary Gaussian elimination of $A - \tau B$ with pivoting suppressed, i.e., with diagonal pivoting. This is the preconditioner that engineers often use because of the bad condition of the correction equation [32]. As real and imaginary parts of $A - \tau B$ are not positive definite, this factorization is potentially unstable [14].
- *Incomplete (complex symmetric) LDL^T factorization preconditioning with drop tolerance ζ, $ILDL^T(\zeta)$.* This factorization is computed in the same manner as

the LDL^T factorization except that, after each column of L has been calculated, all entries in that column which are smaller in magnitude than ζ times the norm of this column are dropped from L [24, section 10.4]. Again, pivoting is suppressed.

Diagonal and symmetric Gauss–Seidel (SSOR($\omega = 1$)) preconditioners were found to be insufficient. Notice that we always permute the matrices according to the minimum degree algorithm applied to $A - \tau B$ to reduce fill-in. Using symmetric approximate minimum degree permutations hardly made any difference. Reordering the matrices considerably reduces consumption of memory and time to compute the (incomplete) factorizations and enhances the quality of the resulting preconditioners.

We compute the six eigenvalues closest to the target $\tau = 100$,

$$\begin{aligned} \lambda_1 &\approx 111.153 + 0.188i, & \lambda_4 &\approx 113.158 + 1.135i, \\ \lambda_2 &\approx 111.717 - 0.084i, & \lambda_5 &\approx 114.440 - 0.016i, \\ \lambda_3 &\approx 112.787 - 0.150i, & \lambda_6 &\approx 115.963 - 0.231i. \end{aligned}$$

The convergence criterion for the outer iteration (JDCS) is $\|(A - \theta(\mathbf{x})B)\mathbf{x}\| < \varepsilon\|\mathbf{x}\|$ with $\varepsilon = 10^{-5}$. The JD algorithm is restarted as soon as the search space has dimension $j_{\max} = 20$. The iteration is continued (restarted) with the $j_{\min} = 10$ best refined Ritz vectors available.

The inner iteration (QMR, COCG, or GMRES) is considered converged if the norm of the relative residual is smaller than $\max\{2^{-1-j}, \varepsilon\}$, where j is the iteration count of the JD algorithm. As suggested in [8], j is set to zero at the beginning of the computation and when an eigenpair has been found. The strategy to only gradually enforce high accuracy in the inner iteration is one of the major reasons for the efficiency and success of the JD algorithm. The maximally allowed steps for the inner iteration was set to 100. This limit was hardly ever hit except for $\zeta \geq 10^{-3}$.

The shift in the correction equation is set equal to the target τ as long as the residual norm $\|(A - \theta(\mathbf{x}_j)B)\mathbf{x}_j\| \geq 0.1\|\mathbf{x}_j\|$. If the relative residual norm drops below 0.1, then the shift $\theta(\mathbf{x}_j)$ is chosen. The value 0.1 was found to be satisfactory by experimentation.

TABLE 5.1
Statistics for the test problem `dwg961` *of order* $n = 961$.

Preconditioner	nnz(L)	QMR		JDCS/ COCG		GMRES		JDQZ/ GMRES	
		$nit_{\rm jd}$	$n_{\rm prec}$	$nit_{\rm jd}$	$n_{\rm prec}$	$nit_{\rm jd}$	$n_{\rm prec}$	$nit_{\rm jd}$	$n_{\rm prec}$
LDL^T	20740	36	237	34	223	35	239	35	218
$ILDL^T(10^{-5})$	16396	36	223	36	222	36	213	47	278
$ILDL^T(10^{-4})$	13590	37	353	37	326	39	315	43	304
$ILDL^T(10^{-3})$	10444	43	2517	42	2469	38	1611	74	986
$ILDL^T(10^{-2})$	7871	—	—	—	—	(43)	(2660)	52	1657

In Table 5.1 we give some statistics on our experiments with the smallest test problem `dwg961`. Ten tests were run with varying random starting vectors. The numbers given in Table 5.1 are averages of these tests. The correction equations were solved with QMR, COCG, or GMRES using the LDL^T or incomplete LDL^T preconditioner with varying drop tolerances ζ. The numbers for $ILDL^T(10^{-2})$ in column JDCS/GMRES are given in parentheses, as only eight out of the ten runs computed all six desired eigenpairs. These numbers are the averages of the eight best runs. (In the two unsuccessful runs only four eigenpairs were found within a reasonable number of steps.)

The column "nnz(L)" indicates the number of nonzero elements of the triangular factor L of the respective preconditioner. Of course, the bigger ζ is, the sparser L is. The number of nonzeros of the lower triangle of $A - \tau B$ is 5776.

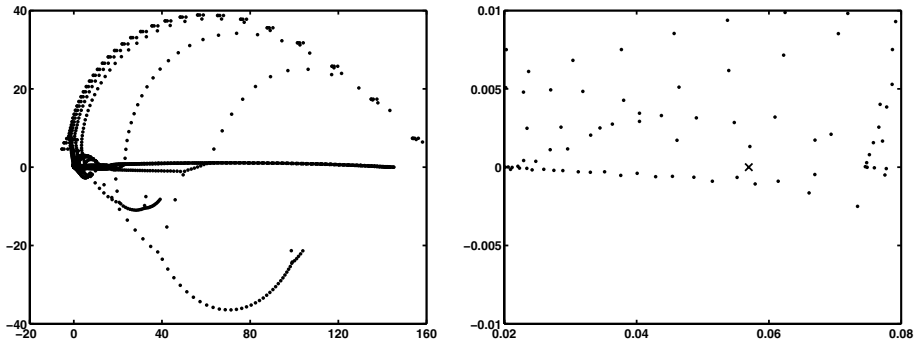
In Table 5.1 the number nit_{jd} of steps of the (outer) JD iteration and the *total* number n_{prec} of calls of the preconditioner are given. The execution of the preconditioner is the most expensive portion of the code. The total number of inner iteration steps is approximately $n_{\mathrm{prec}} - nit_{\mathrm{jd}}$. The correction equations are thus solved on average in about $n_{\mathrm{prec}}/nit_{\mathrm{jd}} - 1$ steps.

As a comparison we also give corresponding numbers for the general purpose JD algorithm, JDQZ, for computing some interior eigenvalues close to the target τ [4, p. 242]. This algorithm computes a partial QZ decomposition for the matrix pencil $(A; B)$. It does not take into account any structure of A or B. The correction equations are solved by GMRES. GMRES does not exploit the structure of our problem either. While COCG and QMR implicitly build (nonorthogonal) bases for the underlying Krylov subspace by means of three-term recurrences, GMRES constructs an orthonormal basis for the Krylov subspace. This gain in stability comes at a high price in that as many auxiliary vectors must be stored as iteration steps are needed to solve the system. The computational complexity even grows quadratically in this number.

JDQZ/GMRES consistently needs more outer iteration steps than the JDCS algorithms. So, the search spaces it builds are not better than those the latter build. To some extent this is due to the fact that in this implementation of JDQZ/GMRES only one iteration step is executed for approximately solving the correction equation in the initial $j_{\min}$ iteration steps. Nevertheless, the number of times the preconditioner is called grows more slowly in JDQZ/GMRES.

The poorer the preconditioner is, the higher is the number of iteration steps needed to solve the correction equation. From Table 5.1 we see that an incomplete LDL^T preconditioner with very low drop tolerance ζ is as powerful as the LDL^T preconditioner; this is not surprising since the number of nonzeros is still quite high. As the drop tolerance increases, the number of nonzeros of the triangular factors decreases substantially—as does the quality of the resulting preconditioner. Thus, n_{prec} grows rapidly. The iteration count to solve a single correction equation is below 30 for $\zeta \leq 10^{-5}$, below 60 for $\zeta \leq 10^{-4}$, and below 90 for $\zeta \leq 10^{-3}$. For COCG and QMR this simply means that investing less memory space in the preconditioners for solving a certain system is done at the expense of higher iteration counts. However, with GMRES increasing iteration counts entail more memory space for the search space. In this particular example, the search space that is built inside JDCS/GMRES consumes much more memory space than the LDL^T preconditioner. The common remedy to save memory by restarting GMRES does not work here. The initial phase of the Krylov subspace methods when the residual norm decreases only slowly is very long such that the restart dimension in GMRES would have to be very big.

MATLAB's `eigs` needs 43 iteration steps to compute the desired six eigenvalues. This function is an implementation of the implicitly restarted Arnoldi algorithm [19]. We applied `eigs` with the shift-and-invert spectral transformation. The shift was chosen to be τ. The Arnoldi iteration was restarted every 20 iteration steps. The spectral transformation implies that a system of equations with the matrix $A - \tau B$ has to be solved in each iteration step of the implicitly restarted Arnoldi algorithm. These systems could be solved iteratively. But, as the Arnoldi relation has to be established accurately, the number of (inner) iteration steps would be much higher



(a) (b)

FIG. 5.2. (a) *complete spectrum of* `toy2` *and* (b) *portion of the spectrum close to target* $\tau = 0.057$. *The target is indicated by* $\times$. *The plot shows imaginary versus real parts of the eigenvalues.*

than those reported in Table 5.1; see, e.g., [2].

We do not provide execution times, as the MATLAB implementations of our codes do not have comparable quality. n_{prec} gives some indication of the execution time, as the invocation of the preconditioner is the most expensive portion of the algorithm. Also $A - \tau B$ is applied to a vector as many times. If GMRES is used as the system solver, then the orthogonalizations can contribute considerably to the execution time.

5.2. A radiating dielectric cavity problem. Our second test example, `toy2`, is a one-dimensional layered dielectric cavity with six distributed Bragg reflector (DBR) pairs at the top and the bottom, and an active quantum well region sandwiched between the two DBR pairs. A PML lining terminates the one-dimensional structure at the two ends. This structure has no practical significance for vertical cavity surface emitting lasers (VCSEL) design. Nevertheless, the treatment of an open cavity using PML can be illustrated. Here, A and B are sparse nonsingular matrices of order 6243. We are looking for six interior eigenvalues close to the real axis in the neighborhood of a real target value that is determined by the laser designer analytically. On the left of Figure 5.2 the whole spectrum is depicted. On the right the vicinity of the spectrum near the target value 0.0569545 ($\times$) is shown. The eigenvalues that we search are

$$\begin{aligned} \lambda_1 &\approx 0.05713 + 0.00132i, & \lambda_4 &\approx 0.05476 + 0.00285i, \\ \lambda_2 &\approx 0.05792 - 0.00107i, & \lambda_5 &\approx 0.06140 - 0.00090i, \\ \lambda_3 &\approx 0.05519 - 0.00065i, & \lambda_6 &\approx 0.06091 + 0.00320i. \end{aligned}$$

The plots indicate that the condition number of the correction equation is at least 10^6. In this example the ratio of imaginary and real parts of the computed eigenvalues is largest.

We proceed just as in the first problem. The results that we obtained are summarized in Table 5.2. Here, we can observe convergence with a drop tolerance as large as $\zeta = 10^{-3}$. The inner iteration does not converge for $\zeta = 10^{-2}$ within the maximally allowed steps. Notice that for $\zeta = 10^{-3}$ the number of nonzeros of L is 75,472, which is still 70% of that for the LDL^T preconditioner. The number of nonzeros of a triangle of $A - \tau B$ is 43,038.

In this example the various solvers behave quite differently. Only JDCS/COCG solves all the problems smoothly and quickly. The numbers in parentheses indi-

TABLE 5.2
Statistics for the test problem `toy2` *of order* $n = 6243$.

Preconditioner	nnz(L)	JDCS/ QMR		JDCS/ COCG		JDCS/ GMRES		JDQZ/ GMRES	
		nit_{jd}	n_{prec}	nit_{jd}	n_{prec}	nit_{jd}	n_{prec}	nit_{jd}	n_{prec}
LDL^T	94399	(20)	(86)	19	86	(22)	(86)	14	85
$ILDL^T(10^{-5})$	90861	(21)	(102)	21	114	(29)	(114)	21	130
$ILDL^T(10^{-4})$	85497	(30)	(207)	29	183	(33)	(197)	30	239
$ILDL^T(10^{-3})$	75472	(29)	(184)	30	184	34	199	41	273
$ILDL^T(10^{-2})$	52740	—	—	—	—	—	—	—	—

cate that some of the ten test runs did not succeed. With JDCS/GMRES seven (LDL^T), eight ($ILDL^T(10^{-5})$), and nine ($ILDL^T(10^{-4})$) runs computed all six desired eigenpairs. In the runs that failed, GMRES could not solve the correction equation sufficiently accurately. The low-quality approximate solutions in turn prevented JDCS from converging. With JDCS/QMR only four, three, four, and two runs properly succeeded with the preconditioners LDL^T, $ILDL^T(10^{-5})$, $ILDL^T(10^{-4})$, and $ILDL^T(10^{-3})$, respectively. In the failed runs, QMR diverged at least once. We did not take failed runs into account. Thus, numbers in parentheses are averages of the successful runs.

Taking only successful runs into account, the JDCS solvers behave quite similarly. They often need less inner iteration steps than JDQZ to compute the desired quantities. It must be stressed, however, that JDCS/QMR has troubles converging without a look-ahead strategy; cf. [11]. For all solvers the number of outer iterations increases slowly as the quality of the preconditioner decreases. The average number of inner iteration steps is small for all solvers.

MATLAB's `eigs` needs 100 iterations to compute the six eigenpairs with the memory requirements of JDCS with the LDL^T preconditioner.

5.3. A waveguide problem with PML. In the third example we deal with a two-dimensional optical waveguide problem. The cross section of the waveguide is considered. The waveguide is designed such that the fundamental optical mode experiences considerably lower losses by leakage into the substrate compared with the higher order optical modes. In this way more reliable single/fundamental mode operations can be achieved in practice. A PML lining terminates the two-dimensional structure on the boundary. The PML is used to render the effect of leakage into the substrate [13]. The order of A and B is $n = 32{,}098$. As in the first example, A has a 2×2 block structure, where only the block A_{11} is nonzero; cf. (5.1). The dimension of the null space is now $m = 10{,}680$.

We are looking for the eigenvalues with small imaginary parts closest to the real target value $\tau = 5.4412$. These are

$$\begin{aligned} \lambda_1 &\approx 5.441307 - 0.000001i, & \lambda_4 &\approx 5.440369 - 0.007041i, \\ \lambda_2 &\approx 5.441331 - 0.005197i, & \lambda_5 &\approx 5.442366 - 0.007212i, \\ \lambda_3 &\approx 5.438328 - 0.004531i, & \lambda_6 &\approx 5.448426 - 0.003837i. \end{aligned}$$

All parameters were set as in the previous two problems. We obtained the results summarized in Table 5.3.

The incomplete LDL^T preconditioners work with drop tolerances $\zeta \le 10^{-3}$. The correction equations were not solved until the required accuracy within 100 steps with $\zeta = 10^{-2}$. All successful $ILDL^T$ preconditioners consume at least 60% of the memory

TABLE 5.3
Statistics for the test problem `wg` *of order* $n = 32{,}098$.

Preconditioner	nnz(L)	JDCS/ QMR		JDCS/ COCG		JDCS/ GMRES		JDQZ/ GMRES	
		$nit_{\rm jd}$	$n_{\rm prec}$	$nit_{\rm jd}$	$n_{\rm prec}$	$nit_{\rm jd}$	$n_{\rm prec}$	$nit_{\rm jd}$	$n_{\rm prec}$
LDL^T	2595054	(24)	(115)	26	181	(24)	(119)	22	120
$ILDL^T(10^{-5})$	1759300	(25)	(112)	25	157	(24)	(100)	27	157
$ILDL^T(10^{-4})$	1652157	(27)	(306)	32	431	(24)	(284)	32	184
$ILDL^T(10^{-3})$	1459440	(36)	(1305)	38	1365	(26)	(715)	64	481
$ILDL^T(10^{-2})$	1099434	—	—	—	—	—	—	—	—

space of the LDL preconditioner. The lower triangle of $A-\tau B$ has 264,194 nonzeros.

JDCS/QMR completed three, nine, seven, and six out of the ten runs with preconditioners LDL^T, $ILDL^T(10^{-5})$, $ILDL^T(10^{-4})$, and $ILDL^T(10^{-3})$, respectively. The corresponding numbers for JDCS/GMRES are four, eight, seven, and seven. As with the previous problem only the successful runs of JDCS/QMR and JDCS/GMRES contributed to the statistics in Table 5.3. JDCS/COCG solved all the problems, although some of the inner iterations did not converge within 100 iteration steps.

In this example the outer, and additionally the inner, iteration counts of JDCS/QMR and JDCS/COCG grow more quickly than those of JDCS/GMRES. Here, we notice the stabilizing effect of the orthonormal basis in GMRES.

Similarly as in the other test problems, $nit_{\rm jd}$ is smallest for JDQZ/GMRES with the LDL^T preconditioner but grows more rapidly than with the other solvers. However, $n_{\rm prec}$ is clearly smallest for the weak preconditioners.

For solving this example, MATLAB's `eigs` needs 32 steps until convergence.

We summarize our experiments as follows. The JDCS algorithm combined with complex symmetric system solvers performs best in example 2 (`toy2`) in almost all cases. In the other two examples both perform similarly to the JD variants that do not exploit structure as long as the quality of the preconditioner is high. Exploiting the matrix structure saves memory and computing time such that in this situation the JDCS/COCG or JDCS/QMR is to be preferred. If the quality of the preconditioner decreases, JDCS combined with GMRES as the solver for the correction equation, or even JDQZ/GMRES, often performs better in terms of inner iteration count. QMR and COCG need more iterations than GMRES to solve the correction equation to the required accuracy. The latter consumes more memory space, however.

6. Discussion and conclusions. We have suggested an algorithm, JDCS, for finding a few eigenvalues and corresponding eigenvectors of special and generalized eigenvalue problems with complex symmetric matrices. JDCS is a natural generalization of standard and two-sided JD for complex symmetric matrices. Most of the techniques known in JD (such as preconditioning the correction equation, using a target, restarting, and computing interior eigenvalues) easily carry over to JDCS.

Exact JDCS has asymptotically cubic convergence for simple eigenvalues of complex symmetric matrices. To get this high convergence rate it is crucial to replace the Euclidean inner product $\mathbf{x}^*\mathbf{y}$ with the bilinear form $\mathbf{x}^T\mathbf{y}$. We have shown that the Rayleigh quotient θ based on this indefinite inner product is asymptotically closer to the exact eigenvalue λ than the Rayleigh quotient ρ derived from the Euclidean inner product.

Compared with the Lanczos algorithm for complex symmetric matrices [6], JDCS is more flexible in that we can restart with any vectors we like and add some extra

vectors to the search space. JDCS also is more stable than Lanczos in the following sense. It is well known that Lanczos may break down; look-ahead versions try to deal with this problem. In JDCS we may have two difficulties, but they can be easily solved. First, the linear solver for the correction equation may break down; in this case we can just use the residual for the subspace expansion. Second, it may be impossible to expand the search space in a complex orthogonal manner (such that $U^T U = I$). In this case, we may restart or expand the search space by the residual or a random vector. In various cases in our experiments, JDCS is superior to JDQZ. While its convergence behavior is similar, it takes the particular structure of the problem into account, which reduces flop count and memory requirements. In the discussed problems, JDCS also converged in fewer iteration steps than MATLAB's `eigs`.

Of course, JDCS can have disadvantages. We may expect problems when we try to approximate an eigenvector $\mathbf{x}$ that is (approximately) quasi-null: the oblique projections and the Rayleigh quotient (1.3) may affect the accuracy and stability. A standard JD algorithm that computes a partial Schur form could be better suited to such a situation. JDCS does not have the short recurrences that the Lanczos algorithm has. Notice, however, that the implicitly restarted Lanczos algorithm in `eigs` performs complete reorthogonalization among the Lanczos vectors, which entails the same recurrence length as JDCS.

The success of JDCS depends very much on the quality with which the correction equations are solved. Furthermore, much of the advantage of JDCS would be lost if the system solver did not exploit the complex symmetry of the problem.

With our most powerful preconditioners, a complex symmetric complete or incomplete LDL^T factorization of $A - \tau B$ with a very restrictive drop tolerance, we observe little differences in the iteration counts obtained with COCG, complex symmetric QMR, or GMRES. As the quality of the preconditioners decreases, the number of iterations to solve the correction equation to the desired accuracy increases. In this situation GMRES builds better, namely orthonormal, bases for the underlying Krylov subspace and extracts a different solution from the space. Therefore, fewer iteration steps are needed to solve the correction equations.

In our numerical problems where we are looking for interior eigenvalues we get convergence only when we set the drop tolerance ζ below or equal to 10^{-3}. With such a choice of ζ, the preconditioners require about two thirds of the memory space of a direct solver. In many practical situations such preconditioners will be too memory consuming. It is therefore of paramount importance to find effective complex symmetric preconditioners that, on the one hand, approximate well $A - \tau B$ and, on the other hand, do not consume so much memory space. Because the eigenvalues given in the examples have a much bigger real than imaginary part, one may think that a good approximation of the real part of $A - \tau B$ would be a good choice. However, the real and imaginary parts of the eigenvectors are in general not much different in norm, and the real part of $A - \tau B$ does not have additional properties other than being symmetric. We therefore think that a multilevel approach taking all of $A - \tau B$ into account will be more promising [3, 18]. This will be the subject of future research.

Acknowledgments. We thank Henk van der Vorst and Gerard Sleijpen (University of Utrecht) for valuable discussions. Matthias Streiff (Integrated Systems Laboratory, ETH Zürich) provided the test examples `toy2` and `wg`. We thank Roland Freund (Bell Laboratories, Murray Hill) for sending us a MATLAB code for the complexsymmetric QMR algorithm. We also would like to thank Angelika Bunse-Gerstner

(University of Bremen) for a code of the CSYM algorithm. The referees helped us by giving useful and constructive comments.

REFERENCES

[1] P. Arbenz and Z. Drmač, *On positive semidefinite matrices with known null space*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 132–149.
[2] P. Arbenz and R. Geus, *A comparison of solvers for large eigenvalue problems originating from Maxwell's equations*, Numer. Linear Algebra Appl., 6 (1999), pp. 3–16.
[3] P. Arbenz and R. Geus, *Multilevel Preconditioners for Solving Eigenvalue Problems Occurring in the Design of Resonant Cavities*, Tech. report 396, Institute of Scientific Computing, ETH Zürich, Zürich, Switzerland, 2003. Available online at http://www.inf.ethz.ch/research/publications/.
[4] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*, Software, Environments, Tools 11, SIAM, Philadelphia, 2000.
[5] A. Bunse-Gerstner and R. Stöver, *On a conjugate gradient-type method for solving complex symmetric linear systems*, Linear Algebra Appl., 287 (1999), pp. 105–123.
[6] J. K. Cullum and R. A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol.* 1: *Theory*, Birkhäuser Boston, Boston, MA, 1985. Reprinted as Classics in Appl. Math. 41, SIAM, Philadelphia, 2002.
[7] J. K. Cullum and R. A. Willoughby, *A QL procedure for computing the eigenvalues of complex symmetric tridiagonal matrices*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 83–109.
[8] D. R. Fokkema, G. L. G. Sleijpen, and H. A. van der Vorst, *Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils*, SIAM J. Sci. Comput., 20 (1998), pp. 94–125.
[9] R. W. Freund, *Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 425–448.
[10] F. R. Gantmacher, *The Theory of Matrices*, Vol. 2, Chelsea, New York, 1959.
[11] H. D. Gersem, D. Lahaye, S. Vandewalle, and K. Hameyer, *Comparison of quasi minimal residual and bi-conjugate gradient iterative methods to solve complex symmetric systems arising from time-harmonic magnetic simulations*, COMPEL, 10 (1999), pp. 298–310.
[12] I. C. Gohberg, P. Lancaster, and L. Rodman, *Matrices and Indefinite Scalar Products*, Oper. Theory Adv. Appl. 8, Birkhäuser Verlag, Basel, 1983.
[13] J. Heaton, M. Bourke, S. Jones, B. Smith, K. Hilton, G. Smith, J. Birbeck, G. Berry, S. Dewar, and D. Wight, *Optimization of deep-etched, single-mode GaAs-AlGaAs optical waveguides using controlled leakage into the substrate*, IEEE J. Lightwave Technology, 17 (1999), pp. 267–281.
[14] N. J. Higham, *Factorizing complex symmetric matrices with positive definite real and imaginary parts*, Math. Comp., 67 (1998), pp. 1591–1599.
[15] M. E. Hochstenbach and G. L. G. Sleijpen, *Two-sided and alternating Jacobi–Davidson*, Linear Algebra Appl., 358 (2003), pp. 145–172.
[16] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.
[17] Z. Jia, *Refined iterative algorithms based on Arnoldi's process for large unsymmetric eigenproblems*, Linear Algebra Appl., 259 (1997), pp. 1–23.
[18] D. Lahaye, H. De Gersem, S. Vandewalle, and K. Hameyer, *Algebraic multigrid for complex symmetric systems*, IEEE Trans. Magnetics, 36 (2000), pp. 1535–1538.
[19] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, Software, Environments, Tools 6, SIAM, Philadelphia, 1998. Available online at http://www.caam.rice.edu/software/ARPACK/.
[20] F. T. Luk and S. Qiao, *A fast eigenvalue algorithm for Hankel matrices*, Linear Algebra Appl., 316 (2000), pp. 171–182.
[21] *Matrix Market*, http://math.nist.gov/MatrixMarket/ (2003).
[22] C. C. Paige, B. N. Parlett, and H. A. van der Vorst, *Approximate solutions and eigenvalue bounds from Krylov subspaces*, Numer. Linear Algebra Appl., 2 (1995), pp. 115–134.
[23] B. N. Parlett, *The Rayleigh quotient iteration and some generalizations for nonnormal matrices*, Math. Comp., 28 (1974), pp. 679–693.
[24] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

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

[26] G. L. G. Sleijpen and J. van den Eshof, *On the use of harmonic Ritz pairs in approximating internal eigenpairs*, Linear Algebra Appl., 358 (2003), pp. 115–137.

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

[28] G. L. G. Sleijpen and H. A. van der Vorst, *The Jacobi–Davidson method for eigenvalue problems and its relation with accelerated inexact Newton scheme*, in Proceedings of the Second IMACS International Symposium on Iterative Methods in Linear Algebra, S. D. Margenov and P. S. Vassilevski, eds., IMACS, New Brunswick, NJ, 1996, pp. 377–389.

[29] G. L. G. Sleijpen and F. W. Wubs, *Exploiting multilevel preconditioning techniques in eigenvalue computations,* SIAM J. Sci. Comput., 25 (2003), pp. 1249–1272.

[30] A. van der Sluis and H. A. van der Vorst, *The convergence behaviour of Ritz values in the presence of close eigenvalues*, Linear Algebra Appl., 88/89 (1987), pp. 651–694.

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

[32] M. Streiff, A. Witzig, and W. Fichtner, *Computing optical modes for VCSEL device simulation*, IEE Proc. Optoelectron., 149 (2002), pp. 166–173.

[33] H. A. van der Vorst and J. B. M. Melissen, *A Petrov-Galerkin type method for solving $Ax = b$, where A is symmetric complex*, IEEE Trans. Magnetics, 26 (1990), pp. 706–708.