SIAM J. SCI. COMPUT.
Vol. 23, No. 2, pp. 517–541

TOWARD THE OPTIMAL PRECONDITIONED EIGENSOLVER: LOCALLY OPTIMAL BLOCK PRECONDITIONED CONJUGATE GRADIENT METHOD[*]

ANDREW V. KNYAZEV[†]

Abstract. We describe new algorithms of the locally optimal block preconditioned conjugate gradient (LOBPCG) method for symmetric eigenvalue problems, based on a local optimization of a three-term recurrence, and suggest several other new methods. To be able to compare numerically different methods in the class, with different preconditioners, we propose a common system of model tests, using random preconditioners and initial guesses. As the "ideal" control algorithm, we advocate the standard preconditioned conjugate gradient method for finding an eigenvector as an element of the null-space of the corresponding homogeneous system of linear equations under the assumption that the eigenvalue is known. We recommend that every new preconditioned eigensolver be compared with this "ideal" algorithm on our model test problems in terms of the speed of convergence, costs of every iteration, and memory requirements. We provide such comparison for our LOBPCG method. Numerical results establish that our algorithm is practically as efficient as the "ideal" algorithm when the same preconditioner is used in both methods. We also show numerically that the LOBPCG method provides approximations to first eigenpairs of about the same quality as those by the much more expensive global optimization method on the same generalized block Krylov subspace. We propose a new version of block Davidson's method as a generalization of the LOBPCG method. Finally, direct numerical comparisons with the Jacobi–Davidson method show that our method is more robust and converges almost two times faster.

Key words. symmetric eigenvalue problems, preconditioning, conjugate gradient methods, the Lanczos method

AMS subject classifications. 65F15, 65N25

PII. S1064827500366124

1. Introduction.

We consider a generalized *symmetric definite* eigenvalue problem of the form $(A-\lambda B)x = 0$ with real symmetric n-by-n matrices A and B, assuming that A is positive definite. That describes a regular matrix pencil $A - \lambda B$ with a discrete spectrum (set of eigenvalues λ). It is well known that such a generalized eigenvalue problem has all real eigenvalues λ_i, and corresponding (right) eigenvectors x_i, satisfying $(A - \lambda_i B)x_i = 0$, can be chosen orthogonal in the following sense: $(x_i, Ax_j) = (x_i, Bx_j) = 0,\ i \neq j$. In some applications, the matrix B is simply the identity $B = I$, and then we have the standard symmetric eigenvalue problem with matrix A, which has n real positive eigenvalues

$$0 < \lambda_{\min} = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n = \lambda_{\max}.$$

In general, when $B \neq I$, the pencil $A - \lambda B$ has n real, some possibly infinite, eigenvalues. If B is nonsingular, all eigenvalues are finite. If B is positive semidefinite, some eigenvalues are infinite, all other eigenvalues are positive, and we consider the

[*]Received by the editors May 18, 2000; accepted for publication (in revised form) October 6, 2000; published electronically July 10, 2001. The main results of the paper were presented by the author at the Mini Symposium on Very Large Eigenvalue Problems at the Fifth U.S. National Congress on Computational Mechanics, University of Colorado at Boulder, 1999, and at the Sixth Copper Mountain Conference on Iterative Methods, Copper Mountain, CO, 2000. A preliminary version of the paper was published as a technical report [16].
http://www.siam.org/journals/sisc/23-2/36612.html

[†]Department of Mathematics, University of Colorado at Denver, P.O. Box 173364, Campus Box 170, Denver, CO 80217-3364 (andrew.knyazev@cudenver.edu, http://www-math.cudenver.edu/~aknyazev).

problem of computing the smallest m eigenvalues of the pencil $A - \lambda B$. When B is indefinite, it is convenient to consider the pencil $\mu A - B$ with eigenvalues

$$\mu = \frac{1}{\lambda},\ \mu_{\min} = \mu_n \leq \cdots \leq \mu_1 = \mu_{\max},$$

where we want to compute the largest m eigenvalues, $\mu_1, \ldots \mu_m$, and corresponding eigenvectors.

An important class of eigenproblems is the class of *mesh eigenproblems*, arising from discretizations of boundary value problems with self-adjoint differential operators of mathematical physics. Such problems appear, e.g., in structural mechanics, where it is usual to call A a *stiffness* matrix and B a *mass* matrix. A mass matrix is usually positive definite, but in some applications, e.g., in buckling, the matrix B is only nonnegative, or even indefinite, while A is positive definite.

Typical properties of mesh eigenproblems are well known; see, e.g., [22]. We just want to highlight that the desired eigenpairs of the matrix pencil $B - \mu A$ are rarely needed with high accuracy as the pencil itself is just an approximation of the original continuous problem and the approximation error may not be small in practice. It means that the typical ratio of the number of iterations and the number of unknowns should be small when a preconditioner is of a reasonable quality. For that reason, in the present paper we are not much interested in such properties of eigensolvers, e.g., in superlinear convergence, which could be observed only after large number of steps.

In the rest of the paper, for brevity, we deal with the pencil $B - \mu A$ mostly. With $B = I$ and $\lambda = 1/\mu$, our results and arguments are readily applied for the pencil $A - \lambda I$.

The paper is organized as follows.

In section 2, we introduce, following [21, 22], preconditioning for eigenvalue solvers and give general definitions of preconditioned single-vector and block, or simultaneous, iterative eigensolvers. We describe the global optimization method on the corresponding generalized Krylov subspace. No efficient algorithm is presently known to perform a global optimization of the Rayleigh quotient on the generalized Krylov subspace. We shall show numerically in section 8, however, that the method we suggest in section 4 provides approximations often quite close to those of the global optimization, at a small fraction of the cost.

In section 3, we outline the "ideal" control algorithm, namely, the standard preconditioned conjugate gradient method, for finding an eigenvector as an element of the null-space of the corresponding homogeneous system of linear equations under assumption that the eigenvalue is known. The algorithm cannot, of course, be used in practice as it requires knowledge of the extreme eigenvalue, but it seems to be a perfect choice as a benchmark for preconditioned eigensolvers for computing the extreme eigenpair.

In section 4, we describe, in some details, a new algorithm of the locally optimal preconditioned conjugate gradient method, based on the local optimization of the three-term recurrence suggested by the author in [19, 21, 22]. In the original algorithm of [19], the three-term recurrence contains the current eigenvector approximation, the current preconditioned residual, and the previous eigenvector approximation. As the current eigenvector approximation and the previous eigenvector approximation get closer in the process of iterations, special measures need to be used in the algorithm to overcome the potential instability. In our new algorithm, the three-term recurrence contains the current eigenvector approximation, the current preconditioned residual, and the implicitly computed difference of the current and previous eigenvector approximations. Such choice resolves instability issues and allows us to write a much simpler and more efficient code.

We present block versions of the method, the locally optimal block preconditioned conjugate gradient (LOBPCG) methods for symmetric eigenvalue problems, in section 5.

To be able to compare numerically different methods in the class, with different preconditioners, we suggest, in section 6, a common system of model tests, using random preconditioners and initial guesses. We recommend that every new preconditioned eigensolver for computing the extreme eigenpair be compared with our "ideal" algorithm on our model test problems in terms of the speed of convergence, costs of every iteration, and memory requirements. We also recommend a comparison with the global optimization method in terms of accuracy.

We provide such comparison for our LOBPCG method in sections 7 and 8. Numerical results of section 7 establish that our algorithm is practically as efficient as the "ideal" algorithm when the same preconditioner is used in both methods. In section 8 we show numerically that the block version of our method comes close to finding the global optimum of the Rayleigh quotient on the corresponding generalized block Krylov subspace.

Section 9 contains an informal discussion of the block Davidson method. We describe a nonstandard restart strategy that makes the block Davidson method a generalization of our LOBPCG method. We argue, however, that such generalization may not be beneficial for symmetric eigenvalue problems.

In section 10, we compare directly our method with a version of the Jacobi–Davidson method [14] for $B = I$. No MATLAB code of the Jacobi–Davidson method for a generalized eigenvalue problem is currently publicly available. We find that our method is much more robust and typically converges almost two times faster. This is not very surprising, as the MATLAB version of the Jacobi–Davidson method available to us for testing is not apparently optimized for symmetric eigenvalue problems, while our method takes full advantage of the symmetry by using a three-term recurrence.

Finally, section 11 contains references to some relevant software written by the author.

We note that the simplicity, robustness, and fast convergence of preconditioned eigensolvers we propose make them a more and more popular choice in applications, e.g., in band structure calculations in two- and three-dimensional photonic crystals [6, 7] and eigenvalue problems for thin elastic structures [32]. Some eigenvalue problems in mechanics, e.g., vibration of a beam supported by springs, lead to equations with nonlinear dependence on the spectral parameter. Preconditioned eigensolvers for such equations are analyzed in [39, 40], where, in particular, a generalization of the theory of a preconditioned subspace iteration method of [9, 10] is presented.

2. Preconditioning for eigenvalue problems.

First, we briefly review a traditional approach for large symmetric generalized eigenproblems, based on using classical methods, e.g., the Lanczos method, for a shifted-and-inverted operator $(B - \nu A)^{-1}A$. It typically lets us quickly compute the eigenvalues closest to the shift ν, assuming that this operation may be implemented with an efficient factorization of $B - \nu A$. However, for very large problems such factorizations are usually too expensive. An inner iterative solver is often used to somewhat circumvent this difficulty; see a review and references in [21, 22] and a recent paper [38].

If B is efficiently factorizable, e.g., $B = I$, so that we can multiply vectors by

AB^{-1}, or $B^{-1}A$, we take $\nu = 0$. In this case, a single iteration may not be expensive, but eigenvalues μ close to zero are usually not of practical interest, and the convergence for eigenvalues of interest is often very slow.

Thus, the traditional approach is inefficient for very large mesh eigenproblems. Preconditioning is the key for significant improvement of the performance as it allows one to find a path between the Scylla of expensive factorizations and the Charybdis of slow convergence.

Preconditioned methods are designed to handle the case when the only operation we can perform with matrices A and B of the pencil is multiplication of a vector by A and B. To accelerate the convergence, we introduce a *preconditioner* T.

In many engineering applications, preconditioned iterative solvers for linear systems $Ax = b$ are already available, and efficient preconditioners $T \approx A^{-1}$ are constructed. It is important to realize that the same preconditioner T can be used to solve an eigenvalue problem $Ax = \lambda x$, or $Bx = \mu Ax$.

We assume that the preconditioner T is *symmetric positive definite*. As A is also symmetric positive definite, there exist positive constants $\delta_1 \geq \delta_0 > 0$ such that

$$\delta_0(T^{-1}x, x) \leq (Ax, x) \leq \delta_1(T^{-1}x, x). \tag{2.1}$$

The ratio δ_1/δ_0 can be viewed as the spectral condition number $\kappa(TA)$ of the preconditioned matrix TA and measures how well the preconditioner T approximates, up to a scaling, the matrix A^{-1}. A smaller ratio δ_1/δ_0 usually ensures faster convergence.

We want to highlight that the assumption we just made on T is essential for the theory (see [21]) but may not be an actual limitation in numerical computations for some methods. In particular, our own method of section 5 is quite robust in practice with respect to the choice of the preconditioner, even when the assumptions above are not satisfied; see Figure 5.2 below as an example of using an indefinite preconditioner.

As we want to discuss an optimality of preconditioned eigensolvers, we need to have a formal definition of the whole class of such methods. We first define, following [21], a preconditioned single-vector iterative solver for the pencil $B - \mu A$, as a generalized polynomial method of the following kind:

$$x^{(k)} = P_k(TA, TB)x^{(0)}, \tag{2.2}$$

where P_k is a polynomial of the kth degree of two independent variables, $x^{(0)}$ is an initial guess, and T is a fixed preconditioner.

We need only choose a polynomial, either a priori or during the process of iterations, and use a recursive formula which leads to an iterative scheme. For an approximation $\mu^{(i)}$ $(\lambda^{(i)})$ to an eigenvalue of the pencil $B - \mu A$ $(A - \lambda B)$ for a given eigenvector approximation $x^{(i)}$ the Rayleigh quotient $\mu(x)$ $(\lambda(x))$ defined as

$$\mu(x^{(i)}) = \frac{(x^{(i)}, Bx^{(i)})}{(x^{(i)}, Ax^{(i)})} \quad \left(\lambda(x^{(i)}) = \frac{(x^{(i)}, Ax^{(i)})}{(x^{(i)}, Bx^{(i)})}\right) \tag{2.3}$$

is typically used.

Let us now define the generalized Krylov subspace:

$$K_k\left(TA, TB, x^{(0)}\right) = \left\{P_k(TA, TB)x^{(0)}\right\}, \tag{2.4}$$

where P_k runs through the set of all polynomials of the kth degree of two independent variables and $x^{(0)}$ is a fixed initial vector. In particular,

$$K_2\left(TA, TB, x^{(0)}\right) \\ = \operatorname{span}\left\{x^{(0)}, TAx^{(0)}, TBx^{(0)}, (TA)^2x^{(0)}, TATBx^{(0)}, TBTAx^{(0)}, (TB)^2x^{(0)}\right\}.$$

We notice that in our definition (2.2) of the preconditioned eigensolver

$$x^{(k)} \in K_k\left(TA, TB, x^{(0)}\right).$$

Having definition (2.2) of the whole class of preconditioned eigensolvers, we can introduce, as in [21], the *global optimization* method for computing the first eigenpair simply by maximizing the Rayleigh quotient $\mu(x)$ on the Krylov subspace (2.4):

$$x_o^{(k)} = \operatorname{argmax}_{x \in K_k\left(TA, TB, x^{(0)}\right)} \mu(x). \tag{2.5}$$

We want to highlight that an efficient algorithm for finding $x_o^{(k)}$, e.g., based on short-term recurrences, is not presently known, and that the number of vectors in the basis of the Krylov subspace (2.4) grows exponentially, which makes the method very expensive in practice, similarly to the situation with Davidson's method (see discussion in [21, 22]), unless restarts are used. Therefore, the global optimization method (2.5) is optimal only in the sense that it provides the global maximum of the Rayleigh quotient on the Krylov subspace, but it may not be optimal if we also count computational costs.

For block methods, we introduce the generalized block Krylov subspace:

$$K_k\left(TA, TB, X^{(0)}\right) = \operatorname{span}\left\{P_k(TA, TB)x_j^{(0)},\ j = 1, \ldots, m\right\}, \tag{2.6}$$

where P_k runs through the set of all polynomials of the kth degree of two independent variables and $X^{(0)} = \operatorname{span}\{x_j^{(0)},\ j = 1, \ldots, m\}$.

A general preconditioned block eigensolver is a generalization of method (2.2) with a single vector being replaced with several ones. Using the Rayleigh–Ritz method is typical for block methods; see [21, 22].

Here, we only want to define the *block global optimization* method, Algorithm 2.1, as we use it later in our numerical experiments.

```
ALGORITHM 2.1. THE BLOCK GLOBALLY OPTIMAL PRECONDITIONED EIGENSOLVER.
Input: $m$ starting vectors $x_1^{(0)}, \ldots x_m^{(0)}$, devices to compute: $Ax$, $Bx$, and $Tx$ for
    a given vector $x$, and the vector inner product $(x, y)$.
  1. Start: select $x_j^{(0)},\ j = 1, \ldots, m$.
  2. Iterate to compute the basis of the generalized block Krylov subspace (2.6).
  3. Use the Rayleigh–Ritz method for the pencil $B - \mu A$ in the subspace
       to compute the Ritz values $\mu_j^{(k)}$ and the corresponding Ritz vectors $x_j^{(k)}$.
Output: the approximations $\mu_j^{(k)}$ and $x_j^{(k)}$ to the largest eigenvalues
    $\mu_j$ and corresponding eigenvectors, $j = 1, \ldots, m$.
```

In our code of the block global optimization method, we do not even try to minimize computation costs and simply compute recursively

$$K_{k+1} = K_k + TAK_k + TBK_k, \tag{2.7}$$

followed by complete orthogonalization. The only purpose of the code is to provide a comparison, in terms of accuracy, for the actual block method we suggest in section 5.

3. The "ideal" preconditioned conjugate gradient method.

In this section, we outline the "ideal" control algorithm, namely, the standard preconditioned conjugate gradient (PCG) method for finding an eigenvector, corresponding to the minimal eigenvalue, as an element of the null-space of the corresponding homogeneous system of linear equations under the assumption that the eigenvalue is known.

We assume $B > 0$ in this section. Let us *suppose* that the minimal eigenvalue λ_1 is *already known*, and we just need to compute the corresponding eigenvector x_1, an element of the null-space of the homogeneous system of linear equations

$$(A - \lambda_1 B)x_1 = 0,$$

where the matrix of the system is symmetric and nonnegative definite. What would be an *ideal* preconditioned method of computing x_1? As such, we choose the *standard PCG method*. It is well known that a PCG method can be used to compute a nonzero element of the null-space of a homogeneous system of linear equations with a symmetric and nonnegative definite matrix if a nonzero initial guess is used. While fitting perfectly the definition of a single-vector preconditioned eigensolver of the previous section, this ideal method *cannot be used in practice* as it requires knowledge of the eigenvalue.

We suggest using the method as a *control* in numerical comparison with practical eigenvalue solvers, in particular, with PCG eigensolvers, e.g., with our locally optimal PCG method. If an eigensolver finds the eigenvector u_1 with the same accuracy and costs as the ideal method, we have reasons to call such an eigensolver "optimal" for computing the eigenvector u_1.

For our numerical tests, we write a code PCGNULL.m, which is a slightly modified version of the built-in MATLAB code PCG.m, revision 1.11, of the standard PCG method to cover solution of homogeneous systems of linear equations with symmetric and nonnegative definite system matrices.

The standard theory of the PCG method for computing an element of the null-space of a symmetric nonnegative definite matrix implies convergence to the eigenvector x_1, which is the T^{-1}-orthogonal projection of the initial guess $y^{(0)}$ to the null-space of the matrix $A - \lambda_1 B$. On the $(i + 1)$st step, the "energy" norm of the error, in our case it's actually the seminorm based on $A - \lambda_1 B$, i.e.,

$$\sqrt{(y^{(i+1)}, (A - \lambda_1 B)y^{(i+1)})}, \tag{3.1}$$

is minimized over the hyperplane

$$H_{i+1} = y^{(0)} + K_i(T(A - \lambda_1 B),\ T(A - \lambda_1 B)y^{(0)}), \tag{3.2}$$

where K_i is the standard Krylov subspace. As x_1 is in the null-space of the matrix $A - \lambda_1 B$, we have

$$(y, T^{-1}x_1) = (y^{(0)}, T^{-1}x_1) \quad \forall y \in H_i, \tag{3.3}$$

in particular, $(y^{(i)}, T^{-1}x_1)$ does not change in the process of iterations, while (3.1) converges to zero at least linearly with the asymptotic average convergence factor $(1 - \sqrt{\xi})/(1 + \sqrt{\xi})$, where ξ is an upper bound for the ratio of the largest and smallest nonzero eigenvalues of the preconditioned matrix $T(A-\lambda_1 B)$. Using estimates of [18] of eigenvalues of $T(A-\lambda_1 B)$ in terms of eigenvalues λ of the pencil $A-\lambda B$ and constants of (2.1), we can obtain such an upper bound and get the following upper bound of the asymptotic average convergence factor:

$$q=\left(\frac{1-\sqrt{\xi}}{1+\sqrt{\xi}}\right),\ \xi=\frac{1}{\kappa(TA)}\left(1-\frac{\lambda_1}{\lambda_2}\right). \tag{3.4}$$

Finally, we would like to remind the reader of a long forgotten version of the PCG method based on optimization of a three-term recurrence, e.g., [35]:

$$y^{(i+1)}=y^{(i)}+\alpha^{(i)}v^{(i)}+\beta^{(i)}(y^{(i)}-y^{(i-1)}),\ v^{(i)}=T(A-\lambda_1 B)y^{(i)},\ \beta^{(0)}=0, \tag{3.5}$$

with both scalar parameters $\alpha^{(i)}$ and $\beta^{(i)}$ computed by minimizing seminorm (3.1) of $y^{(i+1)}$. This version is mathematically equivalent, in exact arithmetic, to the standard version implemented in PCGNULL, which uses two linked two-term recurrences.

This provides an insight into the locally optimal PCG eigensolver [19] we discuss in the next section, where we simply replace in (3.5) exact λ_1 with its approximation, and instead of minimizing seminorm (3.1) of $y^{(i+1)}$ we compute $y^{(i+1)}$ by using the Rayleigh–Ritz method on the subspace

$$y^{(i+1)}\in \text{Span}\ \left\{v^{(i)},\ y^{(i)},\ y^{(i-1)}\right\}. \tag{3.6}$$

An investigation of a possible connection between the two methods is in progress. As we see in section 7, they behave quite similarly in our numerical tests.

4. The PCG methods.

In this section, we propose a new version of the *locally optimal PCG method* [19].

In [19], the author suggested the following PCG method for the pencil $B-\mu A$:

$$x^{(i+1)}=w^{(i)}+\tau^{(i)}x^{(i)}+\gamma^{(i)}x^{(i-1)},\ w^{(i)}=T(Bx^{(i)}-\mu^{(i)}Ax^{(i)}),\ \mu^{(i)}=\mu(x^{(i)}),\ \gamma^{(0)}=0, \tag{4.1}$$

with scalar iteration parameters $\tau^{(i)}$ and $\gamma^{(i)}$ chosen using an idea of *local optimality* [19], namely, select $\tau^{(i)}$ and $\gamma^{(i)}$ that maximize the Rayleigh quotient $\mu(x^{(i+1)})$ by using the Rayleigh–Ritz method. As the current eigenvector approximation $x^{(i)}$ and the previous eigenvector approximation $x^{(i-1)}$ are getting closer to each other in the process of iterations, special measures need to be used in the algorithm to overcome the potential instability.

Formula (4.1) has been used in an earlier revision of our MATLAB code LOBPCG. In our numerical tests, it often led to so ill-conditioned Gram matrices that the Rayleigh–Ritz method would produce spurious eigenpairs. As a cure, we had to use an A-based orthogonalization of the three-dimensional trial subspace, which increased computational costs as had to multiply by A more often.

In our new algorithm, the three-term recurrence contains the current eigenvector approximation, the current preconditioned residual, and the implicitly computed difference between the current and the previous eigenvector approximations:

$$\begin{aligned} x^{(i+1)}&=w^{(i)}+\tau^{(i)}x^{(i)}+\gamma^{(i)}p^{(i)}, \quad w^{(i)}=T(Bx^{(i)}-\mu^{(i)}Ax^{(i)}),\\ p^{(i+1)}&=w^{(i)}+\gamma^{(i)}p^{(i)}, \quad p^{(0)}=0, \quad \mu^{(i)}=\mu(x^{(i)}), \end{aligned} \tag{4.2}$$

with scalar iteration parameters $\tau^{(i)}$ and $\gamma^{(i)}$ chosen using the idea of *local optimality* as above, namely, select $\tau^{(i)}$ and $\gamma^{(i)}$ that maximize the Rayleigh quotient $\mu(x^{(i+1)})$

by using the Rayleigh–Ritz method. We see that

$$p^{(i+1)} = x^{(i+1)} - \tau^{(i)} x^{(i)};$$

thus,

$$x^{(i+1)} \in \text{Span}\,\{w^{(i)},\ x^{(i)},\ p^{(i)}\} = \text{Span}\,\{w^{(i)},\ x^{(i)},\ x^{(i-1)}\},$$

and therefore the new formula (4.2) is mathematically equivalent to the previous one, (4.1), in exact arithmetic.

We describe the actual algorithm of the method given by (4.2) as Algorithm 4.1.

Our experiments confirm that Algorithm 4.1 is much more numerically stable compared to the previous version (4.1) and that it can be used without extra A-orthogonalization in most situations. However, for ill-conditioned problems and when a high accuracy is required, even our new choice of the basis $w^{(i)}, x^{(i)}, p^{(i)}$ of the trial subspace of the Rayleigh–Ritz method may lead to ill-conditioned 3-by-3 Gram matrices, which makes necessary orthogonalization prior to the use of the Rayleigh–Ritz method. In the actual code of LOBPCG, we check for ill-conditioned Gram matrices on every iteration and implement A-orthogonalization if necessary. Since by our assumptions matrix B may not be positive definite, there is no other option except to use the A-based scalar product for the orthogonalization. This typically increases the cost of iterations, but it makes the algorithm more robust.

```
ALGORITHM 4.1. THE LOCALLY OPTIMAL PCG METHOD.
Input: devices to compute: $Ax$, $Bx$, and $Tx$ for a given vector $x$, the vector inner
    product $(x, y)$, and a starting vector $x^{(0)}$.
 1. Start: select $x^{(0)}$ and set $p^{(0)} = 0$.
 2. Iterate: For $i = 0, \ldots,$ Until Convergence Do:
 3.     $\mu^{(i)} := (x^{(i)}, B\,x^{(i)})/(x^{(i)}, A\,x^{(i)})$
 4.     $r := B\,x^{(i)} - \mu^{(i)} A x^{(i)}$
 5.     $w^{(i)} := Tr$
 6.     Use the Rayleigh–Ritz method for the pencil $B - \mu A$
        on the trial subspace Span $\{w^{(i)}, x^{(i)}, p^{(i)}\}$
 7.     $x^{(i+1)} := w^{(i)} + \tau^{(i)} x^{(i)} + \gamma^{(i)} p^{(i)}$,
        (the Ritz vector corresponding to the maximal Ritz value)
 8.     $p^{(i+1)} := w^{(i)} + \gamma^{(i)} p^{(i)}$
 9. EndDo
Output: the approximations $\mu^{(k)}$ and $x^{(k)}$ to the largest eigenvalue $\mu_1$ and its
    corresponding eigenvector.
```

We want to highlight that the algorithm can be implemented with only one application of the preconditioner T, one matrix-vector product Bx, and one matrix-vector product Ax, per iteration.

Storage requirements for Algorithm 4.1 are small—only several n-vectors and no n-by-n matrices at all. Such methods are sometimes called *matrix-free*.

For the stopping criterion, we compute some norms of the preconditioned residual $w^{(i)}$ on every iteration. Such norms may provide accurate two-sided bounds for eigenvalues and a posteriori error bounds for eigenvectors; see [18]. For brevity, we do not consider it in the present paper.

Let us also mention the possibility of avoiding the residual in method (4.1) by splitting it into two vectors:

$$x^{(i+1)} \in \text{Span } \left\{ TAx^{(i)}, TBx^{(i)}, x^{(i)}, x^{(i-1)} \right\}. \tag{4.3}$$

In this new method, the trial subspace is enlarged, which may lead to somewhat faster convergence. However, we need to apply the preconditioner two times now on every iteration; therefore, in our opinion, method (4.3) will not be more efficient for computing the extreme eigenpair than our favorite method, Algorithm 4.1, since the latter already converges practically with the optimal speed; see sections 7 and 8.

Other known CG methods for eigenproblems, e.g., [41, 12, 15, 11, 42, 13, 1, 26], starting from Bradbury and Fletcher [3], are usually constructed as general CG minimization methods, applied to the Rayleigh quotient or to a quadratic form (x, Bx) under the constrain $(x, Ax) = 1$. They are often based on (now standard for linear systems) two linked two-term recurrences,

$$p^{(i)} = -w^{(i)} + \beta^{(i)} p^{(i-1)}, \ x^{(i+1)} = x^{(i)} + \alpha^{(i)} p^{(i)},$$

where $\alpha^{(i)}$ is chosen using a line search to minimize the Rayleigh quotient of $x^{(i+1)}$, which leads to a quadratic equation for $\alpha^{(i)}$, but $\beta^{(i)}$ is computed to make directions $p^{(i)}$ to be conjugate in some sense. These methods *do not utilize* the specific property of the Rayleigh quotient, i.e., that the local minimization of the Rayleigh quotient can be cheaply carried out using the Rayleigh–Ritz method not just in two dimensions, for line search for finding $\alpha^{(i)}$, but in three-dimensional or larger dimensional subspaces as well.

Let us now discuss theoretical convergence rate results for Algorithm 4.1.

The basic fact is that the locally optimal version of the PCG method trivially converges not slower on every step than the preconditioned steepest ascent in terms of the maximizing the Rayleigh quotient [19]; thus, we can use known and well-developed convergence theory of the latter method (e.g., [19]; see also very recent results by Klaus Neymeyr [29, 30]). Our numerical comparison in [21, 22] shows, however, that the PCG method converges much faster in practice than the preconditioned steepest ascent. A ten-fold increase of δ_1/δ_0 leads to the increase in number of iterations, ten-fold for the preconditioned steepest ascent, but only about three-fold for the PCG method, exactly as we would expect for a genuine PCG solver.

No comprehensive convergence theory that would explain such a fast convergence is available yet. Moreover, no even similar results are apparently known at present, e.g., there is no adequate convergence theory of CG methods in nonquadratic optimization. Even if one considers the simplest version of (4.1) for the standard eigenproblem and with no preconditioning, $T = A = I$ when it is reduced to the following trivial method, suggested in [17, 18, 25]: find

$$x^{(i+1)} \in \text{Span } \left\{ Bx^{(i)}, x^{(i)}, x^{(i-1)} \right\} \tag{4.4}$$

by maximizing the Rayleigh quotient $(x, Bx)/(x, x)$ on every step. There is still no convergence theory currently known that would be able to compare (4.4) with the optimal method, in this case, with the Lanczos method for the global maximization of Rayleigh quotient $(x, Bx)/(x, x)$ on the corresponding standard Krylov subspace.

One possible approach to develop an adequate convergence theory may be based on comparison of method (4.4) with the following stationary three-term recurrence:

$$x^{(i+1)} = \alpha Bx^{(i)} + \beta x^{(i)} + \gamma x^{(i-1)},$$

where α, β, γ are fixed scalar parameters, sometimes called the *heavy ball method* in optimization [35]. However, for this simpler method, no accurate convergence theory in terms of the Rayleigh quotient apparently exists yet; cf. [25].

In this paper, we do not prove any new theoretical convergence rate results for Algorithm 4.1, but we suggest a different kind of remedy: numerical comparisons using the benchmark routines we propose; see sections 7 and 8.

We present block versions of Algorithm 4.1 in the next section.

5. Preconditioned simultaneous iterations.

The well-known idea of using *simultaneous*, or *block*, iterations provides an important improvement over single-vector methods, and permits us to compute an $(m > 1)$-dimensional invariant subspace, rather than one eigenvector at a time. It can also serve as an acceleration technique over single-vector methods on parallel computers, as convergence for extreme eigenvalues usually increases with the size of the block, and every step can be naturally implemented on wide varieties of multiprocessor computers.

As in other block methods, the block size should be chosen, if possible, to provide a large gap between first m eigenvalues and the rest of the spectrum as this typically leads to a better convergence; see (5.5) below. Let us also mention that block methods generally handle clusters in the spectrum and multiple eigenvalues quite well; and the block methods we propose below are no exceptions. An attempt should be made to include the whole cluster of eigenvalues into the block, while for multiple eigenvalues this is not essential at all; e.g., if $\mu_{m-1} > \mu_m = \mu_{m+1} > \mu_{m+2}$, then the convergence rate will be determined by the gap $\mu_{m+1} - \mu_{m+2}$ even though the block size is only m; however, only one vector of the two-dimensional eigenspace corresponding to $\mu_m = \mu_{m+1}$ will be computed, as we observe in numerical experiments.

A block version of the locally optimal PCG method [19] was suggested in [21, 22]:

$$x_j^{(i+1)} \in \text{Span}\ \left\{x_1^{(i-1)},\ x_1^{(i)},\ T(B-\mu_1^{(i)}A)x_1^{(i)}, \ldots,\ x_m^{(i-1)},\ x_m^{(i)},\ T(B-\mu_m^{(i)}A)x_m^{(i)}\right\}, \tag{5.1}$$

where $x_j^{(i+1)}$ is computed as the jth Ritz vector. It shares the same problem, as the single-vector version of [19] discussed in the previous section, of having close vectors in the trial subspace.

Our new block Algorithm 5.1 is a straightforward generalization of the single-vector Algorithm 4.1 and is combined with the Rayleigh–Ritz procedure. Here we present two different variants of the algorithm. They differ in the way that the Rayleigh–Ritz method is used. The first version is mathematically equivalent (without round-off errors) to that of [21, 22], but uses directions $p_j^{(i)}$ instead of $x_j^{(i-1)}$, similar to that of Algorithm 4.1.

Here, the column-vector

$$\left(\alpha_1^{(i)}, \ldots, \alpha_m^{(i)},\ \tau_1^{(i)}, \ldots, \tau_m^{(i)}\ \gamma_1^{(i)}, \ldots, \gamma_m^{(i)}\right)^T$$

is the jth eigenvector corresponding to the jth largest eigenvalue of the $3m$-by-$3m$ eigenvalue problem of the Rayleigh–Ritz method, so it should have had an index j also, but we run out of space for indexes.

We observe that

$$p_j^{(i+1)} = x_j^{(i+1)} - \sum_{k=1,\ldots,m} \tau_k^{(i)} x_k^{(i)},$$

```
ALGORITHM 5.1. THE LOBPCG METHOD I.
Input: m starting vectors $x_1^{(0)}, \ldots x_m^{(0)}$, devices to compute: $Ax$, $Bx$, and $Tx$
    for a given vector $x$, and the vector inner product $(x, y)$.
 1. Start: select $x_j^{(0)}$, and set $p_j^{(0)} = 0,\ j = 1, \ldots, m$.
 2. Iterate: For $i = 0, \ldots,$ Until Convergence Do:
 3.     $\mu_j^{(i)} := (x_j^{(i)}, B\, x_j^{(i)})/(x_j^{(i)}, A\, x_j^{(i)}),\ j = 1, \ldots, m;$
 4.     $r_j := B\, x_j^{(i)} - \mu_j^{(i)} A x_j^{(i)},\ j = 1, \ldots, m;$
 5.     $w_j^{(i)} := T r_j,\ j = 1, \ldots, m;$
 6.     Use the Rayleigh–Ritz method for the pencil $B - \mu A$ on the trial subspace
        Span $\{w_1^{(i)}, \ldots, w_m^{(i)}, x_1^{(i)}, \ldots, x_m^{(i)}, p_1^{(i)}, \ldots, p_m^{(i)}\}$;
 7.     $x_j^{(i+1)} := \sum_{k=1,\ldots,m} \alpha_k^{(i)} w_k^{(i)} + \tau_k^{(i)} x_k^{(i)} + \gamma_k^{(i)} p_k^{(i)}$,
        (the $j$th Ritz vector corresponding to the $j$th largest Ritz value),
        $j = 1, \ldots, m$;
 8.     $p_j^{(i+1)} := \sum_{k=1,\ldots,m} \alpha_k^{(i)} w_k^{(i)} + \gamma_k^{(i)} p_k^{(i)}$;
 9. EndDo
Output: the approximations $\mu_j^{(k)}$ and $x_j^{(k)}$ to the largest eigenvalues $\mu_j$ and
    corresponding eigenvectors, $j = 1, \ldots, m$.
```

and thus,

$$\begin{aligned} x_j^{(i+1)} &\in \text{Span}\,\{w_1^{(i)}, \ldots, w_m^{(i)}, x_1^{(i)}, \ldots, x_m^{(i)}, p_1^{(i)}, \ldots, p_m^{(i)}\} \\ &= \text{Span}\,\{w_1^{(i)}, \ldots, w_m^{(i)}, x_1^{(i)}, \ldots, x_m^{(i)}, x_1^{(i-1)}, \ldots, x_m^{(i-1)}\}, \end{aligned}$$

and, indeed, the new Algorithm 5.1 is mathematically equivalent to method (5.1).

We note that Algorithm 5.1 without preconditioning, i.e., with $T = I$, appears in [25] as a "W-accelerated simultaneous gradient method."

Example 5.1. Let us consider the problem of computing first eigenpairs of the standard five-point finite-difference approximation of the Laplacian Δ_h in a $[-1, 1] \times [-1, 1]$ square on a uniform mesh with the step $h = 1/10$, such that the total number of the unknowns is 361. We set $B = I$ and $A = \Delta_h$. The initial approximations are random. No preconditioning is used in the first test, i.e., $T = I$. We plot in Figure 5.1 on a semilog scale relative errors in this example defined as $\|Ax_j^{(i)} - \lambda_j^{(i)} x_j^{(i)}\|_A / \|x_j^{(i)}\|_A$, where we use the standard notation of an A-based norm, $\|\cdot\|_A^2 = (\cdot, A\cdot)$.

Figure 5.1 shows a clear superlinear convergence. This is quite important as some authors, e.g., [34], consider a superlinear convergence as a sign of a genuine CG method. We see that a larger block size in the method improves convergence in this example. We also observe different convergence speed for different eigenpairs on Figure 5.1—the convergence is faster for smaller eigenvalues λ, in particular, λ_1 converges first.

In the second test, we use a preconditioner $T = A - \nu I$ with the shift $\nu = 20$, which is approximately in the middle of the group of the first ten eigenvalues that we compute, starting with random initial approximations. We plot $\|Ax_j^{(i)} - \lambda_j^{(i)} x_j^{(i)}\| / \|x_j^{(i)}\|$, using the Euclidean norm, in Figure 5.2 on a semilog scale. Figure 5.2 shows a superior convergence of the preconditioning. Now, we have a better convergence for eigenvalues close to the shift on the right part of Figure 5.2.

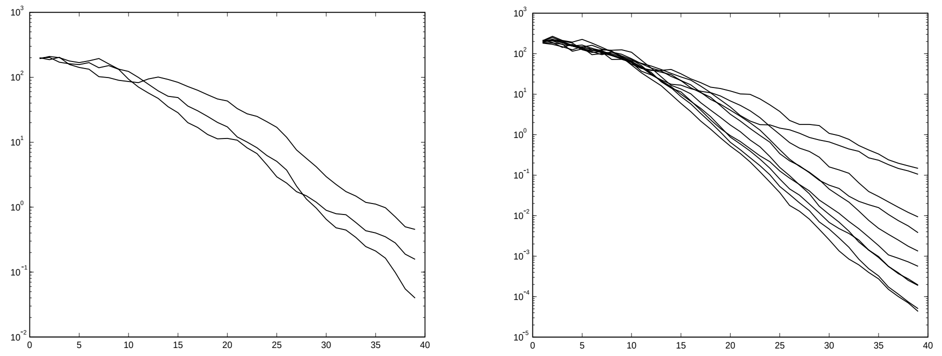


FIG. 5.1. *Errors for the Laplace operator in a square without preconditioning for three (left) and ten (right) first eigenpairs.*

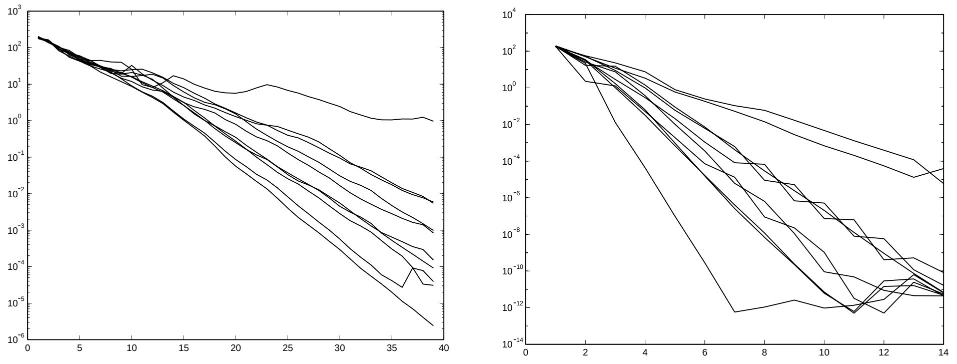


FIG. 5.2. *Errors for the Laplacian operator in a square without preconditioning (left) and with an indefinite preconditioner(right).*

The example thus illustrates that our method works without preconditioning and may even work with an indefinite preconditioner.

Example 5.2. As our next example, we solve a similar eigenvalue problem, but in the L-shaped domain—a union of three unit squares with a mesh uniform in both directions with the step 1/8, such that the total number of unknowns is 161. A specialized domain-decomposition without overlap method for such problem is suggested in [23]. Our numerically computed eigenvalues are consistent with those found in [23]. We compare the performance of the method without preconditioning and with preconditioning based on an incomplete Choleski decomposition of A with a drop tolerance of 10^{-3}. We plot $\|Ax_j^{(i)} - \lambda_j^{(i)} x_j^{(i)}\|/\|x_j^{(i)}\|$ in Figure 5.3. We see that preconditioning leads to approximately quadruple acceleration.

The actual MATLAB code LOBPCG of Algorithm 5.1 that we wrote uses the basis of the trial subspace exactly the way it appears in Algorithm 5.1 until this choice leads to ill-conditioned Gram matrices in the Rayleigh–Ritz method. When such an ill-conditioning shows up, we perform a selected orthogonalization. If this does not fix the problem, as a last resort we apply a complete orthogonalization prior to the Rayleigh–Ritz method. By our assumptions, A may be the only positive definite

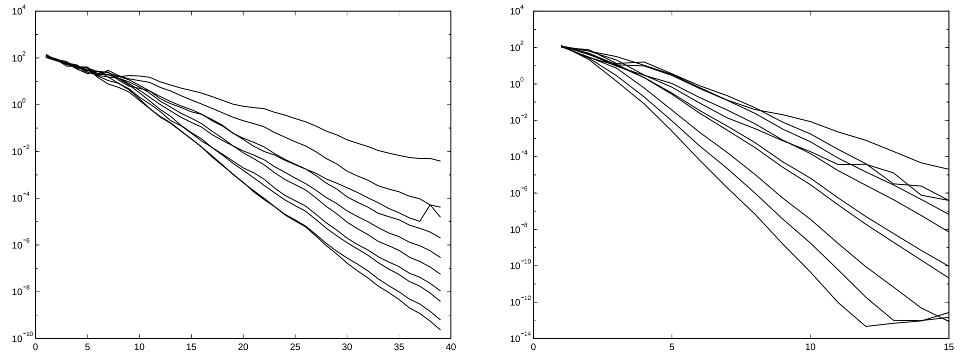


FIG. 5.3. *Residuals for the Laplacian operator in the L-shaped domain with no preconditioning (left) and an ILU preconditioning (right).*

matrix of the pencil; thus, the A-based scalar product is used for the orthogonalization.

There is no theory available yet to predict accurately the speed of convergence of Algorithm 5.1. Similarly to the single-vector case of the previous section, we can easily see that Algorithm 5.1 does not converge slower than the block steepest ascent on every step. Unfortunately, earlier known convergence results [9, 10, 4] for block preconditioned eigensolvers cannot be used to take advantage of this fact. Only the very recent result by Neymeyr [28] for the block preconditioned simple iterations allows us to conclude that Algorithm 5.1 converges linearly with the average asymptotic convergence factors

$$q_j = 1 - \frac{2}{\kappa(TA)+1}\frac{\mu_j - \mu_{j+1}}{\mu_j - \mu_{\min}},\ j = 1, \ldots, m. \tag{5.2}$$

Let us formulate and prove this as the following.

THEOREM 5.1. *For simplicity, let*

$$\mu_j > \mu_j^{(i)} > \mu_{j+1},\ j = 1, \ldots, m.$$

Then

$$\frac{\mu_j - \mu_j^{(i+1)}}{\mu_j^{(i+1)} - \mu_{j+1}} \leq q_j^2 \frac{\mu_j - \mu_j^{(i)}}{\mu_j^{(i)} - \mu_{j+1}}, \tag{5.3}$$

with q_j given by (5.2).

Proof. The proof is straightforward and is based on a possibility of using the convergence rate estimate of [28] recursively and on the fact that our Algorithm 5.1 does not converge slower than the method of [28] in terms of Ritz values as our trial subspace is enlarged compared to that of [28].

The first step is to notice that all results of [28], written for the pencil $A - \lambda B$ with $B = I$, can be trivially applied to a more general case of $B \neq I,\ B > 0$ by using substitutions involving $B^{1/2}$. Secondly, we take the main convergence result of [28], presented as a nasty looking inequality for $\lambda_j^{(i+1)}$, and after tedious but elementary algebraic manipulations we show that it can be reduced to (5.3) with

$$q_j = 1 - \frac{2}{\kappa(TA)+1}\left(1 - \frac{\lambda_j}{\lambda_{j+1}}\right) = 1 - \frac{2}{\kappa(TA)+1}\frac{\mu_j - \mu_{j+1}}{\mu_j}. \tag{5.4}$$

Here, T is optimally scaled such that $\|I - TA\|_A \leq (\kappa(TA) - 1)/(\kappa(TA) + 1)$ to be consistent with assumptions of [28]. Neymeyr has recently found (private communication) an easier proof that takes only two pages.

Now, we use a trick, suggested in [17] and reproduced in [8]; namely, we substitute our actual matrix B, which is not necessarily positive definite with positive definite $B_\alpha = B - \alpha A > 0$ with a scalar $\alpha < \mu_{\text{min}}$ and apply the previous estimate to the pencil $B_\alpha - \mu_\alpha A$ with eigenvalues $\mu_\alpha = \mu - \alpha$, which gives (5.3) with

$$q_j = 1 - \frac{2}{\kappa(TA) + 1} \frac{\mu_j - \mu_{j+1}}{\mu_j - \alpha}.$$

Finally, we realize that the block method of [28], which is the same as that of [4], itself is invariant with respect to α, and everything depends continuously on $\alpha < \mu_{\text{min}}$, so we can take the limit $\alpha = \mu_{\text{min}}$ as well. This proves estimate (5.3) with q_j given by (5.2) for the block method used in [4, 28] for the general pencil $B - \mu A$ satisfying assumptions of the present paper. However, in Algorithm 5.1 the trial subspace is enlarged compared to that of the method of [4, 28]; thus, the convergence rate estimate (5.3) with q_j given by (5.2) holds for our method, too. $\square$

This result, however, does not appear to be sharp for our method, while it is sharp for the method of [28] in a certain sense.

First, if the computed eigenvalues form, or include, a cluster, the factor given by (5.2) is quite pessimistic as it depends on $\mu_j - \mu_{j+1}$, which may be small. For a block method, we expect to have a term $\mu_j - \mu_{m+1}$ instead, where m is the block size.

Second, for a genuine CG method we should count on having $\sqrt{\kappa(TA)}$ instead of $\kappa(TA)$ in the expression for q.

In [21, 22], we demonstrate numerically that our method is much faster than that of [4, 28]. Having in mind estimate (3.4) we have for the PCGNULL and results of [20] on the Rayleigh–Ritz method used in [4], which informally speaking allows us to analyze the error in jth Ritz vector $x_j^{(i)}$ ignoring components $(x_j^{(i)}, x_k)_A,\ k = 1, \ldots, m,\ k \neq j$, we should expect convergence of norms of residuals to be asymptotically linear with the average asymptotic convergence factors

$$q_j = \left(\frac{1 - \sqrt{\xi_j}}{1 + \sqrt{\xi_j}} \right),\ \xi_j = \frac{1}{\kappa(TA)} \frac{\mu_j - \mu_{m+1}}{\mu_j - \mu_{\text{min}}},\ j = 1, \ldots, m. \tag{5.5}$$

Thus, convergence of $\mu_j^{(i)}$ to μ_j should be linear with the ratio q_j^2.

All our numerical tests (see some selected results in sections 7 and 8 below) support our expectation of having asymptotically linear convergence with the ratio (5.5).

A potential disadvantage of Algorithm 5.1 can manifest itself when the number m of eigenpairs of interest is large, as we need to form $3m$-by-$3m$ matrices and solve the corresponding eigenvalue problem of the size $3m$ in the Rayleigh–Ritz method on every step. It seems that vectors $w_j^{(i)}$ and $p_j^{(i)}$ may not always be helpful as basis vectors of the trial subspace of the Rayleigh–Ritz method to improve the approximation of $x_k^{(i+1)}$ for $j \neq k$. And adding unnecessary vectors in the trial subspace increases computational costs. Even more importantly, it may make the algorithm less stable, as the $3m$-by-$3m$ eigenvalue problem of the Rayleigh–Ritz method is likely to inherit ill-conditioning of the original eigenvalue problem when m is large.

In our second variant, Algorithm 5.2, we apply the Rayleigh–Ritz method in two stages: first, as in Algorithm 4.1, for individual indices j, and, second, we include only the minimal set of vectors, namely, just current approximations to different eigenvectors into the trial subspace.

Thus, on the first stage of an iteration of Algorithm 5.2 we solve m three-dimensional eigenproblems, and on the second stage we solve one m-dimensional eigenproblem. But this m-dimensional eigenproblem is constructed using approximate eigenvectors, corresponding to extreme eigenvalues, as a basis of the trial subspace. Therefore, this eigenvalue problem should not be ill-conditioned and no orthogonalization would be required.

We note that arguments of the previous theorem cannot be applied to Algorithm 5.2. Theoretical investigation of accurate convergence estimates of Algorithm 5.2 does not seem to be a trivial exercise.

Our numerical comparison using model problems described in section 6 below shows that Algorithm 5.2 converges with practically the same speed as Algorithm 5.1, except for the case of finding a cluster of eigenvalues with high accuracy. For eigenvalues in a cluster, Algorithm 5.2 at first converges similarly to Algorithm 5.1, but it then slows down and soon after may hit a plateau, i.e., may stop improving the quality of approximations. See Figure 5.4, which shows convergence history of eigenvalue errors of a one run of Algorithms 5.1 (solid) and 5.2 (dotted) with block size $m = 3$. The left picture is for well-separated eigenvalues, and errors for different algorithms are not possible to distinguish. The right picture corresponds to the cluster of three eigenvalues with $(\mu_1 - \mu_3)/(\mu_1 - \mu_{\min}) \approx 10^{-11}$.

Algorithm 5.2. The LOBPCG method II.

Input: m starting vectors $x_1^{(0)}, \ldots x_m^{(0)}$,devices to compute: Ax, Bx, and Tx for a given vector x,and the vector inner product (x, y),

```
1. Start: select $x_j^{(0)}$, and set $p_j^{(0)} = 0,\ j = 1, \ldots, m$.
2. Iterate: For $i = 0, \ldots,$ Until Convergence Do:
3.     $\mu_j^{(i)} := (x_j^{(i)}, B x_j^{(i)})/(x_j^{(i)}, A x_j^{(i)}),\ j = 1, \ldots, m$;
4.     $r_j := B x_j^{(i)} - \mu_j^{(i)} A x_j^{(i)},\ j = 1, \ldots, m$;
5.     $w_j^{(i)} := T r_j,\ j = 1, \ldots, m$;
6.     Use the Rayleigh–Ritz method $m$ times for the pencil $B - \mu A$ on the trial
       subspaces Span $\{w_j^{(i)},\ x_j^{(i)},\ p_j^{(i)}\},\ j = 1, \ldots, m$;
7.     $\widehat{x}_j^{(i+1)} := w_j^{(i)} + \tau_j^{(i)} x_j^{(i)} + \gamma_j^{(i)} p_j^{(i)}$,
       (the Ritz vector corresponding to the maximal Ritz value), $j = 1, \ldots, m$;
8.     $p_j^{(i+1)} := w_j^{(i)} + \gamma_j^{(i)} p_j^{(i)}$;
9.     Use the Rayleigh–Ritz method for the pencil $B - \mu A$ on the trial
       subspaces $\{\widehat{x}_1^{(i+1)}, \ldots, \widehat{x}_m^{(i+1)}\}$;
10.    $x_j^{(i+1)}$ := the $j$th Ritz vector corresponding to the $j$th largest Ritz value,
       $j = 1, \ldots, m$;
11. EndDo
```

Output: the approximations $\mu_j^{(k)}$ and $x_j^{(k)}$ to the largest eigenvalues μ_j and corresponding eigenvectors, $j = 1, \ldots, m$.

For mesh eigenproblems, when high accuracy of computed eigenvalues and eigenvectors is not often required, Algorithm 5.2 can be recommended. For a general purpose method, it seems best to use a mix of Algorithms 5.1 and 5.2 such that eigenvalues, which have already formed a cluster, are treated together using Algorithm 5.1,

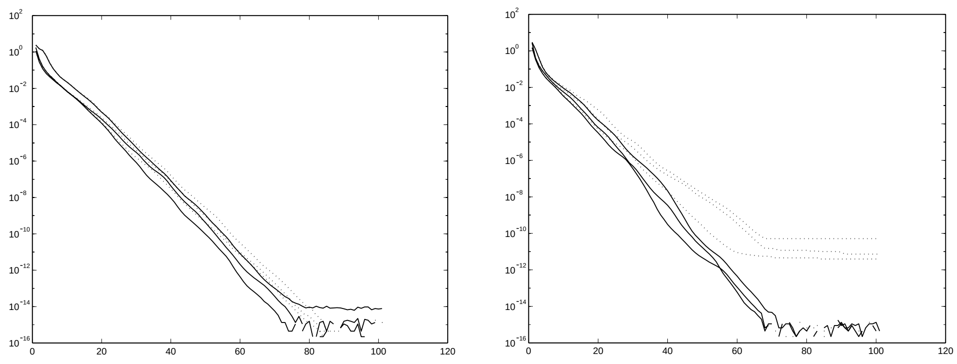


FIG. 5.4. *LOBPCG* I *(solid) vs. LOBPCG* II *(dotted): well-separated eigenvalues (left) and eigenvalue cluster (right).*

while all other eigenvalues in the block are treated separately, as in Algorithm 5.2. This approach will be implemented in the code LOBPCG.m in future revisions.

In the rest of the paper, all our numerical results are for Algorithm 5.1 only.

As theoretical investigation and comparison of preconditioned eigensolvers are quite tedious ([4] provides a perfect example of this), a possibility of a fair numerical comparison becomes even more important than usual. In the next section, we suggest a numerical benchmark for preconditioned eigensolvers.

6. Model test problems with random preconditioners.

To be able to compare numerically different methods in the class with different preconditioners, we suggest the following system of model tests, with random preconditioners and initial guesses, be used for benchmarking.

For simplicity, we take the mass matrix $B = I$.

The stiffness matrix A is in our model tests a diagonal matrix with the minimal entry 1 and the maximal entry 10^{10}; therefore, all eigenvalues μ of the pencil $B - \mu A$ lie on the semiclosed interval $(0, 1]$. We are interested in finding a group of the largest eigenvalues and corresponding eigenvectors. In most of the tests of the present paper, we compute only one, the largest eigenvalue $\mu = 1$.

For preconditioned eigensolvers, we expect that the convergence does not slow down when the condition number of A gets larger [18, 19, 4, 21, 22], with a properly chosen preconditioner. Because of this, we simply use a fixed large value, 10^{10}, for the maximal entry of A. Our code seems to work robustly for condition number of A as large as 10^{16} in double precision arithmetic.

The gap between computed and excluded parts of the spectrum is known to play an important role in the convergence speed. It seems necessary to fix the gap within a series of tests. We do it with the gap ranging from 1 to 0.01 in different series. A small value of the gap may or may not lead to slow convergence, depending on several factors, the first of which is the distribution of eigenvalues in the excluded part of the spectrum close to the desired part. This makes comparison of different methods somewhat unreliable when the gap is small.

It is also necessary to choose a distribution of eigenvalues. The desired eigenvalues, if there is more then one, are distributed randomly on a given interval, as their distribution should not affect performance significantly. In the rest of the spectrum, the distribution of eigenvalues does not noticeably affect the speed of convergence in our tests for a general preconditioner. If, however, the preconditioner commutes with A, e.g., $T = I$, or $T = A^{-1}$, we do observe a strong influence of the distribution on convergence. For such cases, we choose a distribution that mimics that of an ordinary differential equation of the fourth order, but with the given maximal entry 10^{10} (see above). The initial guess is fixed for every run of the actual and the control codes but is changed for every new run as a vector with random entries, chosen from a normal distribution with mean zero and variance one.

The preconditioner T is also fixed for every run of the actual and the control programs but is modified for every new run as a random symmetric positive definite matrix with the fixed value of $\kappa(TA) = \delta_1/\delta_0$ of (2.1). We construct T as follows.

First, we chose a diagonal matrix D with random diagonal entries, chosen from a uniform distribution on the interval $(0, 1)$. Then we find minimal min$D$ and maximal max$D$ values of D and do a linear scaling $D = 1+(D-\text{min}D)/(\text{max}D-\text{min}D)*(\kappa-1)$, where $\kappa = \kappa(TA)$ is the desired fixed value of the spectral condition number of TA. That makes diagonal entries of D uniformly distributed on the interval $(1, \kappa)$ with minimal and maximal values exactly at the end points of the interval; thus, the condition number of D equals exactly κ.

Second, we take a square matrix with random entries, chosen from a normal distribution with mean zero and variance one, and perform the standard orthogonalization procedure on it. That produces a random orthogonal matrix Q. We now scale it, $S = QA^{-1/2}$, keeping in mind that Q is orthogonal and A is diagonal, and therefore, $S^T = A^{-1/2}Q^{-1}$.

Finally, we form $T = S^TDS$. The matrix T is clearly symmetric. Moreover, the diagonal entries of D and columns of Q are eigenpairs of the matrix $A^{1/2}TA^{1/2} = Q^{-1}DQ$, which completes our argument.

There are two reasons for using random preconditioners for our model test problems. First, it is a natural choice when solving eigenvalue problems for diagonal matrices. Second, it allows us to make a fair numerical comparison of different eigensolvers and gives a simple opportunity to check that the best method in the class consistently outperforms other methods independently of the choice of the preconditioner for a fixed value of κ.

The size of the problem varies from 1000–4000.

The upper bound, 4000, is simply determined by our computer resources. The above algorithm of constructing a random preconditioner is quite expensive and leads to a full matrix T. The total cost grows cubically with the size of the problem. We find in our tests that small problems may lead to unreliable conclusions when the number of iterations is large enough. Namely, in some tests, depending on distribution of excluded eigenvalues of the original pencil, we observe a superlinear convergence of our methods when the total number of steps was more than 30% of the size of the problem. However, in practical applications of interest, eigenvalue problems are so large that the number of steps should not usually exceed 20% of the size of the problem, taking also into account that the high accuracy of desired eigenpairs of the algebraic pencil is rarely needed, as the pencil itself is just an approximation of the original continuous problem and the approximation error may not be small. Thus, one should not count in practice on having a superlinear convergence; and we try to rule such a possibility out by choosing the size of the problem large enough.

We recommend every new preconditioned eigensolver be compared with our "ideal" algorithm in our model test problems in terms of the speed of convergence, costs of every iteration, and memory requirements. We provide such comparison for our

LOBPCG method in the next section.

7. Numerical results: LOBPCG vs. PCGNULL. Here, we solve a model eigenvalue problem with $\lambda_1 = 1,\ \lambda_2 = 2$ and take the condition number of A to be 10^{10}.

For simplicity, the mass matrix equals the identity $B = I$.

We compute only the first eigenpair, i.e., the smallest eigenvalue $\lambda_1 = 1$ and the corresponding eigenvector. Thus, we set the block size $m = 1$ in our LOBPCG Algorithm 5.1.

The initial guess is fixed for every run of the actual and the control codes but is modified randomly for every new run. The preconditioner T is also fixed for every run of the actual and the control programs but is modified for every new run as a random symmetric positive definite matrix with the fixed value of the condition number $\kappa(TA)$; see above. We vary $\kappa(TA)$ in different series of tests.

The straight (and green on a color print) line corresponds to linear convergence with the residual reduction rate (3.4), which is the same as (5.5) with $m = 1$ and $\mu_{\min} = 0$. To be consistent with MATLAB's built-in code PCG.m, we measure the error as the Euclidean norm of the residual, i.e., $\|(A-\lambda_1 B)x^{(i)}\|/\|x^{(i)}\|$ in PCGNULL and $\|(A-\lambda^{(i)} B)x^{(i)}\|/\|x^{(i)}\|$ in our code. With these definitions, norms of the residuals are quite large on the first few iterations in our tests as our matrix A is very ill-conditioned.

The average slope is the most important. We observe in Figure 7.1 that the average residual reduction rate is about the same for the "ideal" method, PCGNULL, and for our LOBPCG, and is quite close to the theoretical prediction. Convergence history lines for every method are tightly bundled together, with the bundle for our LOBPCG (colored red in the electronic version of the paper) consistently a bit lower than the bundle for the PCGNULL (dotted and blue). We present here, because of space limitations, only cases $\kappa(TA) = 4$ and $\kappa(TA) = 1000$. Pictures with other values of κ are similar to those shown in Figure 7.1. Thus, our code converges essentially as fast as the "ideal" method.

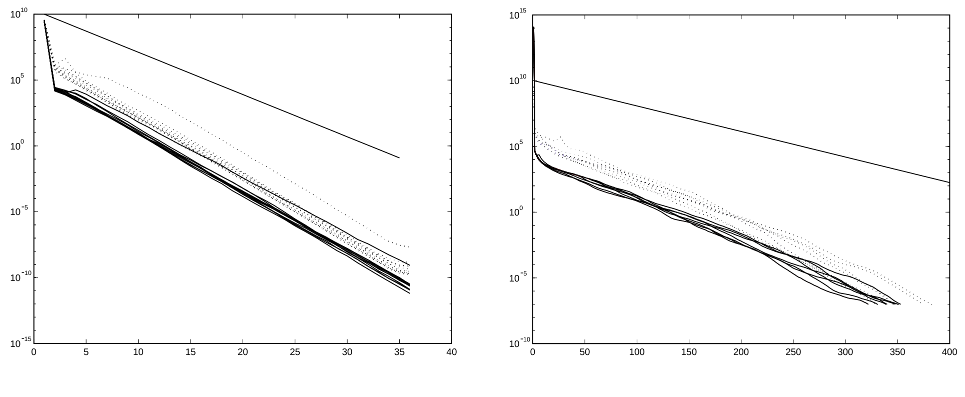


FIG. 7.1. *LOPCG (solid) vs. ideal (dotted):* $\kappa(TA) = 4$ *(left) and* $\kappa(TA) = 1000$ *(right).*

Let us now compare computational costs of a single iteration. PCGNULL involves one application of the preconditioner T, and one multiplication of A and a vector. LOBPCG (revision 3.2.9) has exactly the same major costs; however, for very ill-conditioned problems and when a very high accuracy is required, to increase stability we perform A-based orthogonalization of basis vectors of the trial subspace, which may require matrix A be multiplied two times instead of one on every iteration.

In terms of memory use, both methods are similar, as they require only several vectors, but no large matrices, to be stored.

To conclude, numerical results establish that our algorithm is practically as efficient as the "ideal" algorithm when preconditioners and initial approximations are the same in both methods.

8. Numerical results: LOBPCG vs. GLOBALMIN. In the previous section, we compare our LOBPCG with the benchmark based on PCGNULL and show that LOBPCG is practically the optimal preconditioned method for finding the extreme eigenvalue and the corresponding eigenvector. LOBPCG can also be used, of course, for computing a group of extreme eigenvalues when the block size $m > 1$.

What benchmark do we advocate for preconditioned eigensolvers for finding several eigenpairs? We do not have an answer to this question as satisfactory as that for a single extreme eigenpair, because we are not yet able to suggest a convincing "ideal" (in terms of speed and costs) solver. We do have, however, the block globally optimal solver, Algorithm 2.1, which computes optimal approximations on the block generalized Krylov subspace.

Let us highlight again that the number of vectors in the basis of the block Krylov subspace (2.6) grows exponentially, which makes Algorithm 2.1 very expensive. On the other hand, it provides the global optimization of the Rayleigh quotient on the block Krylov subspace and, thus, can be used for numerical comparison with actual block preconditioned eigensolvers to check if they provide approximations close to those of the global optimization.

We write a MATLAB code of Algorithm 2.1, called GLOBALMIN.m, using recursion (2.7), followed by complete orthogonalization.

We tested LOBPCG vs. GLOBALMIN on model problems described in section 6 with $n = 3000$. The gap between computed and excluded parts of the spectrum is one. The condition number of A is chosen to be 10^8 as GLOBALMIN fails in some tests for larger condition numbers.

We find in the present section that putting only one run on a figure is more illustrative, as LOBPCG and GLOBALMIN produce very similar results, but they change greatly with different random initial guesses. We remove the initial value of the error from all pictures as it is typically too large to fit the chosen scale. LOBPCG is presented by a solid (and red in a color version of the paper) line, while GLOBALMIN is dashed (and blue). The straight (and green) line corresponds, as in section 7, to the average error reduction predicted by (5.5).

The residual-based error is measured as $\|(A-\lambda_j^{(i)}B)x_j^{(i)}\|/\|x_j^{(i)}\|$ on a corresponding Ritz vector $x_j^{(i)}$. The eigenvalue error is simply measured as the difference of the Ritz value $\lambda_j^{(i)}$ and the corresponding eigenvalue λ_j. Both methods, LOBPCG and GLOBALMIN, should monotonically decrease the eigenvalue error. GLOBALMIN provides the global minimum of the Rayleigh quotient; therefore, the eigenvalue error of GLOBALMIN should be always not larger than that of LOBPCG.

We first compare, on Figure 8.1, errors just for the smallest eigenvalue λ_1 for the LOBPCG and GLOBALMIN both with block size one as in section 7. Figure 8.1 displays errors for the same problems as Figure 7.1, but we also add the eigenvalue error. We highlight again that dimension of the generalized Krylov subspace global (2.4) grows exponentially with the number of iterations. For numerical tests

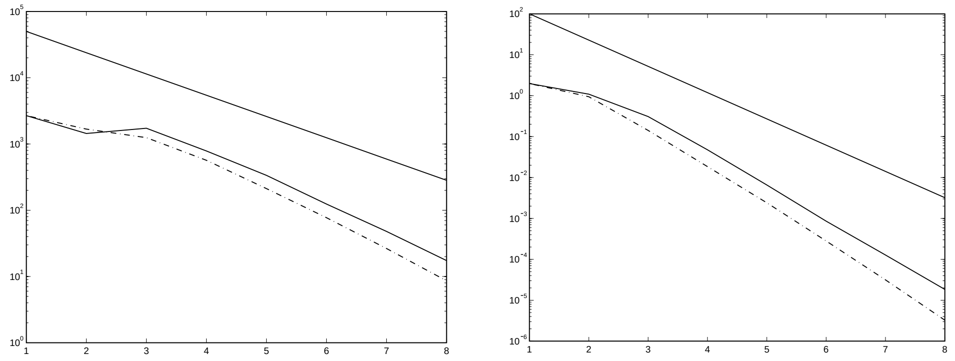


FIG. 8.1. *LOBPCG (solid) vs. GLOBALMIN (dashed), $\kappa(TA) = 4$, $m = 1$: residuals (left) and eigenvalue errors (right).*

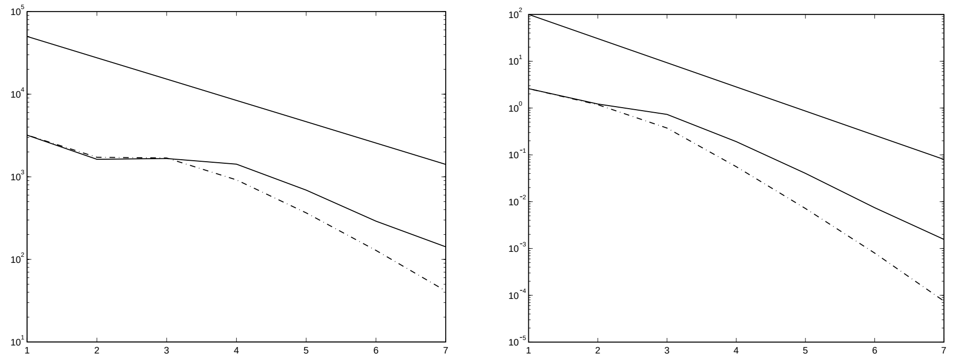


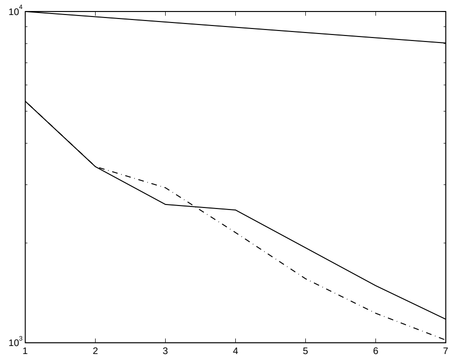
FIG. 8.2. *LOBPCG (solid) vs. GLOBALMIN (dashed), $\kappa(TA) = 4$, $m = 3$: residuals (left) and eigenvalue errors (right).*

presented in Figure 8.1, typical dimensions are $3, 7, 15, 31, 63, 127, 255, 511$, e.g., on the last (eighth) iteration, GLOBALMIN minimizes the Rayleigh quotient on a trial subspace of dimension 511.

In Figures 8.2 and 8.3, we compare the error for the third smallest eigenvalue λ_3 for the LOBPCG and GLOBALMIN, both with block size three. In these experiments, dimensions of the block generalized Krylov subspace global (2.6) typically are $9, 21, 45, 93, 189, 381, 765$. As our code GLOBALMIN is based on complete orthogonalization that filters out possible linearly dependent vectors in the trial subspace, in different tests we observe slightly different dimensions.

The trial subspace in GLOBALMIN is getting large enough—about 10%–20% of the size of the problem—to lead to a superlinear convergence of GLOBALMIN when $\kappa(TA)$ is not too large; see Figure 8.2. We believe that this effect is artificially created by the fact that our problem is of a small size, only 3000, and should be disregarded. Our computer resources do not allow us to solve larger problems.

We first observe that LOBPCG and GLOBALMIN produce almost the same approximations on the first two steps. Most importantly, by comparing the slopes on the figures, we come to the conclusion that our LOBPCG provides approximations close to those of the global optimization method on the generalized block Krylov



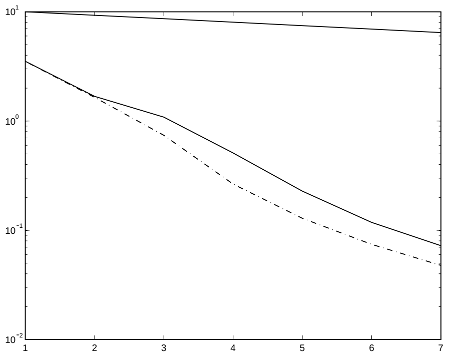


FIG. 8.3. *LOBPCG (solid) vs. GLOBALMIN (dashed), $\kappa(TA) = 1000$, $m = 3$: residuals (left) and eigenvalue errors (right).*

subspace and has a similar convergence speed.

9. LOBPCG vs. Davidson's method.

The discussion above allows us to make some conclusions on LOBPCG vs. Davidson's method, though we do not have a numerical comparison. The block Davidson method without restarts can be presented (cf. [5, 27, 37]) as the Rayleigh–Ritz method on the trial subspace spanned on vectors:

$$\begin{Bmatrix} x_1^{(0)},\ T(B-\mu_1^{(0)}A)x_1^{(0)},\ T(B-\mu_1^{(1)}A)x_1^{(1)},\dots,\ T(B-\mu_1^{(i)}A)x_1^{(i)},\dots \\ x_m^{(0)},\ T(B-\mu_m^{(0)}A)x_m^{(0)},\ T(B-\mu_m^{(1)}A)x_m^{(1)},\dots,\ T(B-\mu_m^{(i)}A)x_m^{(i)} \end{Bmatrix}, \tag{9.1}$$

and $x_j^{(i+1)}$ is computed as the jth Ritz vector. All vectors (9.1) are in the block generalized Krylov subspace (2.6) (assuming a fixed preconditioner), so such defined block Davidson method cannot converge faster than the GLOBALMIN. But our LOBPCG converges with about the same rate as the GLOBALMIN. Therefore, we can expect that LOBPCG is more efficient than Davidson's method, since the former should not converge much slower than, but is significantly less expensive than, the latter.

To make Davidson's method more competitive with our LOBPCG, one needs to restart after every k steps in the following special way: the Rayleigh–Ritz method is now used on the trial subspace spanned by vectors

$$\begin{Bmatrix} x_1^{(i-1)},\ x_1^{(i)},\ T(B-\mu_1^{(i)}A)x_1^{(i)},\ T(B-\mu_1^{(i+1)}A)x_1^{(i+1)},\dots,\ T(B-\mu_1^{(i+k)}A)x_1^{(i+k)},\dots, \\ x_m^{(i-1)},\ x_m^{(i)},\ T(B-\mu_m^{(i)}A)x_m^{(i)},\ T(B-\mu_m^{(i+1)}A)x_m^{(i+1)},\dots,\ T(B-\mu_m^{(i+k)}A)x_m^{(i+k)} \end{Bmatrix}, \tag{9.2}$$

and $x_j^{(i+k+1)}$ is computed as the jth Ritz vector. The new trial subspace is still a subset of the block generalized Krylov subspace (2.6), but its dimension does not depend on the number of iterations.

Compared to a naive method of restarts, we have extra vectors, $x_j^{(i-1)}$, $j = 1,\dots,m$, in the basis of the trial subspace, which we expect will make method (9.2) much faster. We now notice that Davidson's method based on (9.2) with $k = 0$ coincides with our earlier method (5.1). Thus, our Algorithm 5.1 can be viewed as a specially-restarted-at-every-step block Davidson method.

Is there any benefit to using block Davidson method, based on (9.2) with $k > 0$? In our opinion, for symmetric eigenproblems, the answer seems to be no. We expect methods with $k = 0$ and $k > 0$ to be quite similar to each other in terms of speed of convergence, as the method with $k = 0$ already provides approximations practically close to those of the global optimization method on the block Krylov subspace. At the same time, method (9.2) with $k > 0$ will be somewhat more computationally expensive and less stable for ill-conditioned problems simply because the dimension of its trial subspace for the Rayleigh–Ritz method is larger. A direct numerical comparison is yet to be done.

10. Numerical results: LOBPCG vs. JDQR.

Here, we present numerical results for the MATLAB code JDQR.m of the inexact Jacobi–Davidson method [14], written by Gerard Sleijpen, which is publicly available on the Internet (http://www.math.uu.nl/people/sleijpen/JD_software/JDQR.html). We are not able to compare it with PCGNULL, as recommended, because norms of the actual residuals are not available in JDQR on every inner iteration. Instead, we provide results of direct numerical comparison of JDQR with our method LOBPCG.

As in the previous two sections, the comparison is made using model eigenvalue problems with $B = I$ (a revision of JDQR code for generalized eigenproblems is not yet available) and random preconditioners, suggested in section 6. JDQR is used with the default tolerance. The number of iterations of our LOBPCG is chosen to match the accuracy of eigenvector approximations provided by the JDQR. We measure the accuracy as the angle between computed and exact invariant subspaces in the two-norm.

First, we find that JDQR is not as robust as our method with respect to ill-conditioning of the matrix A and the number of required eigenpairs. JDQR consistently fails to find even one eigenpair for condition number of A above 10^8. JDQR becomes even more sensitive to ill-conditioning when we increase the number of required eigenpairs. With $cond(A) = 10^6$, JDQR typically fails to compute all ten required eigenpairs in another series of tests, and in some of the tests outputs only one eigenpair out of ten. Attempts to compute forty eigenpairs using JDQR even with $cond(A) = 10$ produce no more than 16 eigenpairs.

JDQR does not handle random initial guess very well. In some tests it converges to the second eigenpair instead of the desired first one, with the smallest eigenvalue λ.

Our LOBPCG is much more robust and successfully computes all required eigenpairs in all tests mentioned above without any difficulties. Moreover, LOBPCG always converges about one-and-a-half to two times faster than JDQR if we count the number of iterations as the number of times the preconditioner is invoked. This may not be very surprising, as the only MATLAB version of the inexact Jacobi–Davidson method available to us for testing can be used for nonsymmetric eigenproblems as well and is not apparently optimized for symmetric eigenvalue problems, while our method takes full advantage of the symmetry by using a three-term recurrence.[1]

Both methods are scalable, as expected, with respect to the size of the problem and the quality of the preconditioner used.

A comparison of LOBPCG with JDQR has also been made for some practical problems, e.g., for an eigenvalue problem describing vibrations of a slender beam,

[1]A new MATLAB package, JDCG by Yvan Notay, was released in 2001; see http://mnsgi.ulb.ac.be/pub/docs/jdcg. It implements the Jacobi–Davidson method with a PCG inner solver, specifically tuned for the symmetric case [31]. According to our numerical tests, while JDCG does accelerate JDQR, it still converges slower than—and for clusters of eigenvalues is not as reliable as—our LOBPCG.

with a domain-decomposition-based preconditioner. This is outside of the scope of the present paper and will be reported elsewhere. Let us just highlight that the conclusions based on these practical comparisons are the same as above for our model diagonal eigenproblem with random preconditioners.

11. Availability of software for the preconditioned eigensolvers.

The Internet page http:// www-math.cudenver.edu/~aknyazev/software/CG/ is maintained by the author. It contains all the software used for numerical experiments of the present paper, including benchmarking codes written in MATLAB by the author.

12. Conclusion.

Let us formulate here the main points of the paper:

- Numerical results establish that our algorithm LOBPCG is practically as efficient as the "ideal" algorithm for computing an extreme eigenpair and provides approximations close to those of the global optimization method on the generalized block Krylov subspace.
- Our method is much more robust and typically converges about one and a half to two times faster than the inexact Jacobi–Davidson method.
- We provide a system of test problems with random preconditioners that we suggest be used for benchmarking. Every new preconditioned solver for finding an extreme eigenpair should be compared with the "ideal" algorithm in terms of the speed of convergence, costs of every iteration, and memory requirements. As the number of publications on different preconditioned eigenvalue solvers and their applications, e.g., recent papers [36, 43, 44, 24, 2, 34, 33], keeps growing rapidly, a need for such benchmarking becomes evident.

REFERENCES

[1] L. Bergamaschi, G. Gambolati, and G. Pini, *Asymptotic convergence of conjugate gradient methods for the partial symmetric eigenproblem*, Numer. Linear Algebra Appl., 4 (1997), pp. 69–84.

[2] L. Borges and S. Oliveira, *A parallel Davidson-type algorithm for several eigenvalues*, J. Comput. Phys., 144 (1998), pp. 727–748.

[3] W. W. Bradbury and R. Fletcher, *New iterative methods for solution of the eigenproblem*, Numer. Math., 9 (1966), pp. 259–267.

[4] J. H. Bramble, J. E. Pasciak, and A. V. Knyazev, *A subspace preconditioning algorithm for eigenvector/eigenvalue computation*, Adv. Comput. Math., 6 (1996), pp. 159–189.

[5] E. R. Davidson, *Matrix eigenvector methods*, in Methods in Computational Molecular Physics, G. H. F. Direcksen and S. Wilson, eds., D. Reidel, Boston, MA, 1983, pp. 95–113.

[6] D. C. Dobson, *An efficient method for band structure calculations in* 2D *photonic crystals*, J. Comput. Phys., 149 (1999), pp. 363–376.

[7] D. C. Dobson, J. Gopalakrishnan, and J. E. Pasciak, *An efficient method for band structure calculations in* 3D *photonic crystals*, J. Comput. Phys., 161 (2000), pp. 668–679.

[8] E. G. D'yakonov, *Optimization in solving elliptic problems*, CRC Press, Boca Raton, FL, 1996.

[9] E. G. D'yakonov and A. V. Knyazev, *Group iterative method for finding lower-order eigenvalues*, Moscow Univ. Comput. Math. Cybernet., No. 2, 1982, pp. 32–40.

[10] E. G. D'yakonov and A. V. Knyazev, *On an iterative method for finding lower eigenvalues*, Russian J. Numer. Anal. Math. Modelling, 7 (1992), pp. 473–486.

[11] A. Edelman, T. A. Arias, and S. T. Smith, *The geometry of algorithms with orthogonality constraints*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 303–353.

[12] A. Edelman, T. A. Arias, and S. T. Smith, *Curvature in conjugate gradient eigenvalue computation with applications to Materials and Chemistry Calculations*, in Proceedings of the Fifth SIAM Conference on Applied Linear Algebra, J. G. Lewis, ed., SIAM, Philadelphia, 1994, pp. 233–238.

[13] Y. T. Feng and D. R. J. Owen, *Conjugate gradient methods for solving the smallest eigenpair of large symmetric eigenvalue problems*, Internat. J. Numer. Methods Engrg., 39 (1996), pp. 2209–2229.

[14] 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 (1999), pp. 94–125.

[15] G. Gambolati, F. Sartoretto, and P. Florian, *An orthogonal accelerated deflation technique for large symmetric eigenproblems*, Comput. Methods Appl. Mech. Engrg., 94 (1992), pp. 13–23.

[16] A. V. Knyazev, *Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method*, Technical report UCD-CCM 149, Center for Computational Mathematics, University of Colorado at Denver, 2000.

[17] A. V. Knyazev, *Computation of eigenvalues and eigenvectors for mesh problems: Algorithms and error estimates*, Dept. Numerical Math., USSR Academy of Sciences, Moscow, 1986 (in Russian).

[18] A. V. Knyazev, *Convergence rate estimates for iterative methods for mesh symmetric eigenvalue problem*, Soviet J. Numer. Anal. Math. Modelling, 2 (1987), pp. 371–396.

[19] A. V. Knyazev, *A preconditioned conjugate gradient method for eigenvalue problems and its implementation in a subspace*, in Eigenwertaufgaben in Natur- und Ingenieurwissenschaften und ihre numerische Behandlung, Oberwolfach, 1990, Internat. Ser. Numer. Math. 96, Birkhäuser, Basel, 1991, pp. 143–154.

[20] A. V. Knyazev, *New estimates for Ritz vectors*, Math. Comp., 66 (1997), pp. 985–995.

[21] A. V. Knyazev, *Preconditioned eigensolvers—an oxymoron?*, Electron. Trans. Numer. Anal., 7 (1998), pp. 104–123.

[22] A. V. Knyazev, *Preconditioned eigensolvers: Practical algorithms*, in Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, SIAM, Philadelphia, 2000, pp. 352–368. An extended version published as Technical report UCD-CCM 143, Center for Computational Mathematics, University of Colorado, Denver, 1999.

[23] A. V. Knyazev and A. L. Skorokhodov, *Preconditioned gradient-type iterative methods in a subspace for partial generalized symmetric eigenvalue problems*, SIAM J. Numer. Anal., 31 (1994), pp. 1226–1239.

[24] S. H. Lui, H. B. Keller, and T. W. C. Kwok, *Homotopy method for the large, sparse, real nonsymmetric eigenvalue problem*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 312–333.

[25] A. Meyer, *Modern Algorithms for Large Sparse Eigenvalue Problems*, Mathematical Research 34, Akademie-Verlag, Berlin, 1987.

[26] M. Mongeau and M. Torki, *Computing Eigenelements of Real Symmetric Matrices via Optimization*, Technical report 99.54, MIP University Paul Sabatier, Toulouse, France, 1999.

[27] R. B. Morgan, *Davidson's method and preconditioning for generalized eigenvalue problems*, J. Comput. Phys., 89 (1990), pp. 241–245.

[28] K. Neymeyr, *A geometric theory for preconditioned inverse iteration applied to a subspace*, Math. Comp., submitted.

[29] K. Neymeyr, *A geometric theory for preconditioned inverse iteration.* I*: Extrema of the Rayleigh quotient*, Linear Algebra Appl., 322 (2001), pp. 61–85.

[30] K. Neymeyr, *A geometric theory for preconditioned inverse iteration.* II*: Sharp convergence estimates*, Linear Algebra Appl., 322 (2001), pp. 87–104.

[31] Y. Notay, *Combination of Jacobi–Davidson and conjugate gradients for the partial symmetric eigenproblem*, Numer. Linear Algebra Appl., to appear. Also available from http://homepages.ulb.ac.be/˜ynotay/.

[32] E. E. Ovtchinnikov and L. S. Xanthis, *Effective dimensional reduction algorithm for eigenvalue problems for thin elastic structures: A paradigm in three dimensions*, Proc. Natl. Acad. Sci. USA, 97 (2000), pp. 967–971.

[33] E. E. Ovtchinnikov and L. S. Xanthis, *Successive eigenvalue relaxation: A new method for generalized eigenvalue problems and convergence estimates*, Proc. Roy. Soc. London Sect. A, 457 (2001), pp. 441–451.

[34] B. G. Pfrommer, J. Demmel, and H. Simon, *Unconstrained energy functionals for electronic structure calculations*, J. Comput. Phys., 150 (1999), pp. 287–298.

[35] B. T. Polyak, *Introduction to Optimization*, Optimization Software Inc. Publications Division, New York, 1987.

[36] A. Ruhe, *Rational Krylov: A practical algorithm for large sparse nonsymmetric matrix pencils*, SIAM J. Sci. Comput., 19 (1998), pp. 1535–1551.

[37] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Halsted, New York, 1992.

[38] P. Smit and M. H. C. Paardekooper, *The effects of inexact solvers in algorithms for symmetric eigenvalue problems*, Linear Algebra Appl., 287 (1999), pp. 337–357.

[39] S. I. Solov'ev, *Convergence of the Modified Subspace Iteration Method for Nonlinear Eigenvalue Problems*, Preprint SFB393/99-35, Sonderforschungsbereich 393 an der Technischen

Universität Chemnitz, Technische Universität, Chemnitz, Germany, 1999. Also available from http://www.tu-chemnitz.de/sfb393/Files/PS/sfb99-35.ps.gz.

[40] S. I. Solov'ev, *Preconditioned Gradient Iterative Methods for Nonlinear Eigenvalue Problems*, Preprint SFB393/00-28, Sonderforschungsbereich 393 an der Technischen Universität Chemnitz, Technische Universität, Chemnitz, Germany, 2000. Also available from http://www.tu-chemnitz.de/sfb393/Files/PS/sfb00-28.ps.gz.

[41] E. Suetomi and H. Sekimoto, *Conjugate gradient like methods and their application to eigenvalue problems for neutron diffusion equation*, Ann. Nuclear Energy, 18 (1991), pp. 205.

[42] H. Yang, *Conjugate gradient methods for the Rayleigh quotient minimization of generalized eigenvalue problems*, Computing, 51 (1993), pp. 79–94.

[43] T. Zhang, G. H. Golub, and K. H. Law, *Subspace iterative methods for eigenvalue problems*, Linear Algebra Appl., 294 (1999), pp. 239–258.

[44] T. Zhang, K. H. Law, and G. H. Golub, *On the homotopy method for perturbed symmetric generalized eigenvalue problems*, SIAM J. Sci. Comput., 19 (1998), pp. 1625–1645.