Preconditioning Eigenvalues and Some Comparison of Solvers

Ronald B. Morgan

Abstract

Preconditioning techniques are discussed for symmetric eigenvalue problems. The methods Davidson, Jacobi-Davidson, Rayleigh quotient iteration, and preconditioned Lanczos are considered. Some relationships are given between these different approaches, and some experimental comparisons are done. Jacobi-Davidson appears to be efficient in both expense and storage. A hybrid method may be helpful for the case of a poor initial vector.

short running title: **Preconditioning Eigenvalues**

Key Words: eigenvalues, preconditioning, Davidson's method, Jacobi-Davidson

AMS(MOS) Subject Classifications: 65F15, 15A18

1 Introduction

Finding eigenvalues is an important task in scientific computation. There are many applications in physics, chemistry, and engineering. These include computing energy levels of atoms, finding vibrational states of molecules, and determining how buildings will vibrate during earthquakes. Frequently scientists wish to know some eigenvalues of very large matrices. For such problems, Krylov subspace methods are well known. The Lanczos algorithm [9, 22] is a Krylov subspace method for symmetric problems. For nonsymmetric matrices, the methods are Arnoldi [1, 26, 35] and nonsymmetric Lanczos [9, 26].

For large systems of linear equations, preconditioning is an important technique for improving the spectrum. While it is not as straightforward, preconditioning can also be used for eigenvalue problems. Methods that use preconditioning are no longer strictly Krylov methods, although they are generally still related. In cases where an effective, inexpensive preconditioner is available, preconditioning can significantly improve the convergence and provide a better method.

Preconditioning of eigenvalue problems is only partly developed. However, work has been done for some time in Russia; see [8] for a summary and references. Also, Ruhe [25, 23, 24] used SOR and the conjugate gradient method to find eigenvalues in the 1970's. Around the same time, quantum chemists, including Davidson [4], developed correction methods

for their large symmetric matrices. In 1986, in [19], Davidson's method was viewed as a diagonal preconditioning method, and it was generalized for arbitrary preconditioners. For more on Davidson's method, see [18, 16, 3, 37, 38]. The Jacobi-Davidson method [31] was developed in 1996. The inexact rational Krylov method [10] and truncated RQ [36] are recent approaches that are related to Jacobi-Davidson.

This paper discusses preconditioning methods and does some fairly simple comparisons. The focus is on the methods more than on the preconditioners (not much has been done on effectiveness of various preconditioners for eigenvalue problems, but see [33, 21, 32]). We consider only a few methods and only the symmetric case. The methods are the generalized Davidson (GD) method [19], preconditioned Lanczos (PL) [20], the Rayleigh quotient iteration (RQI) with preconditioned conjugate gradient solution of the linear equations [11, 39], and Jacobi-Davidson (JD) [31] with also the preconditioned conjugate gradient method. New implementations of PL and RQI are also given: vectors in the outer loops are saved and the Rayleigh-Ritz procedure [22] is applied.

One motivation for this paper is to see how the recent JD method compares with the others. Also, we wanted to see if the robustness of PL could be improved with the addition of the outer Rayleigh-Ritz; if it could compete better with GD, in terms of total number of iterations. And finally, the more robust implementation of RQI is given to make the comparisons with it more fair. It is desired that the discussion and the comparisons in this paper provide some insights into these methods. However, further comparisons of preconditioning methods are definitely needed.

Section 2 describes the methods that will be considered and gives the new modifications. Section 3 has discussion, and section 4 has experiments.

2 Description of methods

We consider the eigenvalue problem $Az = \lambda z$, where A is a symmetric matrix. However, the algorithms listed below all have nonsymmetric versions. Instead of PL, preconditioned Arnoldi [13] can be used. JD and RQI just need a nonsymmetric iterative linear equations solver [27, 28, 7, 40, 2] in the inner loop.

The GD method generates a subspace with the preconditioned operator $M^{-1}(A - \theta I)$, where θ is an approximate eigenvalue and M is an approximation to $A - \theta I$. It uses the Rayleigh-Ritz procedure to extract approximate eigenvectors from the subspace. We quickly describe the Rayleigh-Ritz procedure [22, 26]. Let V be an orthonormal matrix with columns spanning the desired subspace. The Rayleigh-Ritz procedure finds eigenpairs (θ, s) of the small matrix $V^T AV$. The θ's are approximate eigenvalues, called Ritz values. The approximate eigenvectors or Ritz vectors are $y = Vs$.

ALGORITHM .1 : **Generalized Davidson's Method**

- *Begin with k orthonormal starting vectors $v_1, v_2, \ldots, v_k$.*
- *For $j = k, k+1, \ldots$ do*
  1. *Apply the Rayleigh-Ritz procedure to the subspace $Span\{v_1, v_2, \ldots, v_j\}$. Let the best approximation to the eigenpair of interest be (θ, y), with y normalized. Choose a preconditioner M which may either be fixed or may depend on θ.*
  2. *Find the residual vector for y, $r = (A - \theta I)y$. If $||r|| \leq TOL$, accept that eigenpair, otherwise continue.*
  3. *Compute $w_j = M^{-1}r$. Orthonormalize w_j against $v_1, \ldots, v_j$ to form v_{j+1}.*

In Davidson's original method, $M = D - \theta I$, where D is the diagonal matrix with the same main diagonal as A.

When j becomes too large, the Rayleigh-Ritz expense can be prohibitive. Then Davidson's method needs to be restarted. For instance, in step 1 we can add: If $j = jmax$, pick the k best approximate eigenvectors, orthonormalize them to give $v_1, \ldots v_k$, and let $j = k$.

The PL method takes GD's operator $M^{-1}(A - \theta I)$, but requires M to be a positive definite preconditioner. Then $M^{-1}(A - \theta I)$ can be symmetrized and the Lanczos algorithm applied. There is an outer loop that updates the approximate eigenpair.

ALGORITHM .2 : **Preconditioned Lanczos**

- *Choose a starting vector* y_0. *Let* $\theta_0 = y_0^T A y_0 / y_0^T y_0$.
- *For* $j = 0, 1, 2, \ldots$ *do*
  1. *Choose a SPD preconditioner* M_j *for* $A - \theta_j I$, *and factor as* $M_j = L_j L_j^T$.
  2. *Apply the Lanczos method to* $W_j = L_j^{-1}(A - \theta_j I)L_j^{-T}$ *with initial vector* $L_j y_j$ *and with stopping criterion* $rn < -\nu_j$, *where* ν_j *is the smallest Ritz value of* W_j *and* rn *is the associated residual norm. When the Lanczos loop has ended, let* w_j *be the normalized Ritz vector that corresponds to* ν_j.
  3. *Compute* $y_{j+1} = L^{-T} w_j$, *which is an approximate eigenvector of* A, *and its Rayleigh quotient* $\theta_{j+1} = \theta_j + \nu_j / y_{j+1}^T y_{j+1}$.
  4. *Find the residual vector for* y_{j+1}, $r = (A - \theta_{j+1} I) y_{j+1} / ||y_{j+1}||$. *If* $||r|| \leq TOL$, *accept that eigenpair, otherwise continue.*

When several eigenvalues are being sought, an approximation to the next one is calculated by finding the Ritz vector corresponding to the second Ritz value of W_j, and then multiplying it by L^{-T} (this will be the new y_0 vector after the current eigenpair is accepted). This is done when $||r||$ is first less than $TOL^{2/3}$. While computing the second and subsequent eigenvalues, the ones already determined are shifted out of the way. For the lth eigenvalue, replace $A - \theta_j I$ in step 2 with $A - \theta_j I + \gamma z_1 z_1^T + \ldots + \gamma z_{l-1} z_{l-1}^T$, for γ a value such that $\lambda_1 + \gamma$ is moved beyond the eigenvalues of interest.

Some expense can be saved by not checking the convergence of the inner Lanczos loop at every step. Also, a test can be used to terminate the Lanczos loop early when convergence is near [20]. When the M_j inner product is used in the Lanczos loop, there is no need to factor M_j [12]. The Krylov basis is then M_j orthogonal. This is simlar to how the preconditioned conjugate gradient method is implemented.

As was suggested in the conclusion of [20], we now modify PL in order to make it more robust. The outer loop now applies a small Rayleigh-Ritz procedure to the last few approximate eigenvectors that have been developed.

ALGORITHM .3 : **Preconditioned Lanczos with an Outside Rayleigh-Ritz Projection**

- *Same as Preconditioned Lanczos except for:*

  3. Compute $x = L^{-T}w_j$ and put it in a set of x vectors. If a new y_0 vector is computed (see the discussion in the paragraph after the PL algorithm), immediately add it also to the set of x vectors. Apply the Rayleigh-Ritz procedure with matrix A to the subspace spanned by the x vectors. Let (θ_{j+1}, y_{j+1}) be the smallest Ritz pair from this outside Rayleigh-Ritz.

  4. Find the residual vector for y_{j+1}, $r = (A - \theta_{j+1}I)y_{j+1}/||y_{j+1}||$. If $||r|| \leq TOL$, accept that eigenpair and let the new y_0 be the second approximate eigenvector from the Rayleigh-Ritz procedure in 3, otherwise continue.

We abbreviate this method as PL-RR.

Similar to the discussion after the GD algorithm, once the subspace of $x's$ becomes too large, it can be restarted, retaining the best approximate eigenvectors. The same is true for the Rayleigh-Ritz procedures in the RQI and JD algorithms that are given next. We modify the simple RQI algorithm by adding the outside Rayleigh-Ritz.

ALGORITHM .4 : **Rayleigh Quotient Iteration**

- *Same as Generalized Davidson except for:*

  3. Solve $(A - \theta I)w_j = y$, using a stable conjugate gradient method such as SYMMLQ, preconditioned by a SPD matrix M.

For the stopping test in SYMMLQ, we use improvement in residual norm of $min\{.01, ||r||^2\}$.

For JD, a different system of linear equations is solved.

ALGORITHM .5 : **Jacobi-Davidson**

- *Same as Generalized Davidson except for:*

  3. Solve $(I - yy^T)(A - \theta I)(I - yy^T)w_j = r$, using a preconditioned conjugate gradient method. In the preconditioning step, CG uses the inverse of $(I - yy^T)M(I - yy^T)$.

See [31, 33, 29] for JD implementation details, including how to avoid most applications of $(I - yy^T)$. We use regular preconditioned CG instead of SYMMLQ, because it is stable in

our tests. The CG loop uses stopping test of relative residual norm less than 2^{-j} [29]. As in PL, a test is used to terminate the inner loop early when convergence is near. Specifically, the 2^{-j} test is replaced by $min(.5, .5 * 10^{log_{10}(TOL)-log_{10}(||r||)})$ if this quantity is larger than 2^{-j}.

For computing the second and subsequent eigenvalues, the previous ones can be deflated by using $(I-yy^T)(I-QQ^T)(A-\theta I)(I-QQ^T)(I-yy^T)$, where Q is the orthonormal matrix whos columns are the converged eigenvectors, in place of $(I-yy^T)(A-\theta I)(I-yy^T)$ [29]. This is used in some of our experiments.

3 Discussion of methods

3.1 The operator $M^{-1}(A-\theta I)$

All of the preconditioning methods discussed here use essentially the same operator in their inner loop. This operator is $M^{-1}(A-\theta I)$, with θ an approximate eigenvalue. GD has this operator with θ changing every iteration. With PL, the symmetric version of this operator is used during the Lanczos run, and θ is fixed for each run. RQI has the matrix $M^{-1}(A-\theta I)$ in its inner loop of preconditioned conjugate gradients. The same is true for JD, except that the current approximate eigenvector is deflated out of this operator.

Since these methods are all similar in their core operators, we might expect that they would give similar results. This is not necessarily the case. However, it is safe to say that they all share the same limitation. They can only be as effective as the preconditioned operator $M^{-1}(A-\theta I)$ allows them to be. For the preconditioning to be worthwhile, the spectrum of this operator needs to be a significant improvement over that of the original operator A. Let (λ, z) be the desired eigenpair. If θ is approximately equal to λ, the important operator is $M^{-1}(A-\lambda I)$. This operator has eigenvalue 0 with eigenvector z. So it has the correct eigenvector, and a major question is whether the eigenvalue 0 is better separated from the rest of the spectrum than λ was in the spectrum of A.

We give a couple of examples of how the spectrum of A can be changed by the preconditioning. The matrices are nonsymmetric, even though the focus of this paper is on symmetric problems. This makes the examples more general, and also the spectral plots are more interesting with complex eigenvalues. The first example shows that eigenvalue preconditioning can be effective. Then a case is given where preconditioning does not work well.

Example 3.1. The first matrix has diagonal elements 1, 2, 3, . . . 100, and all other entries distributed normally with mean 0 and standard deviation 1. The smallest eigenvalue of A is 0.2787. We look at the spectrum of A and the spectrum of $N = (D-\theta I)^{-1}(A-\theta I)$, where D is the diagonal matrix with the main diagonal of A. Also $\theta = 0.28$, which is accurate to two decimal places. See figures 3.1 and 3.2 for the plots. It is clear that the preconditioned spectrum of N is an improvement over that of A. The desired eigenvalue of N has much better separation relative to the entire spectrum. We give the ratio of the distance to the closest eigenvalue with the distance to the farthest eigenvalue. A smaller ratio can mean the eigenvalue is more difficult to find, although there other factors such as the positioning of the eigenvalues and the degree of nonnormality. The ratio is 0.04 for the smallest eigenvalue of A compared to 0.28 for the eigenvalue of N near zero. This is a significant improvement.

Fig. 3.1. Spectrum without preconditioning from example 3.1.

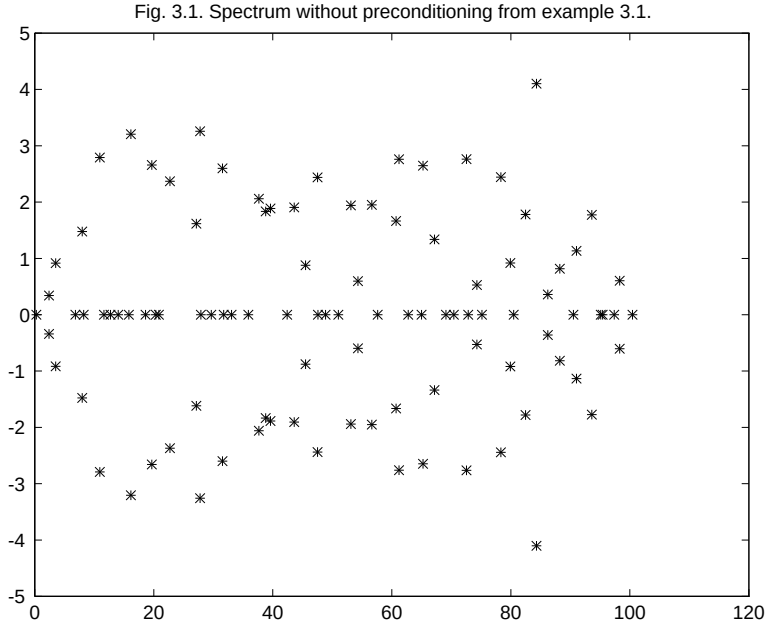
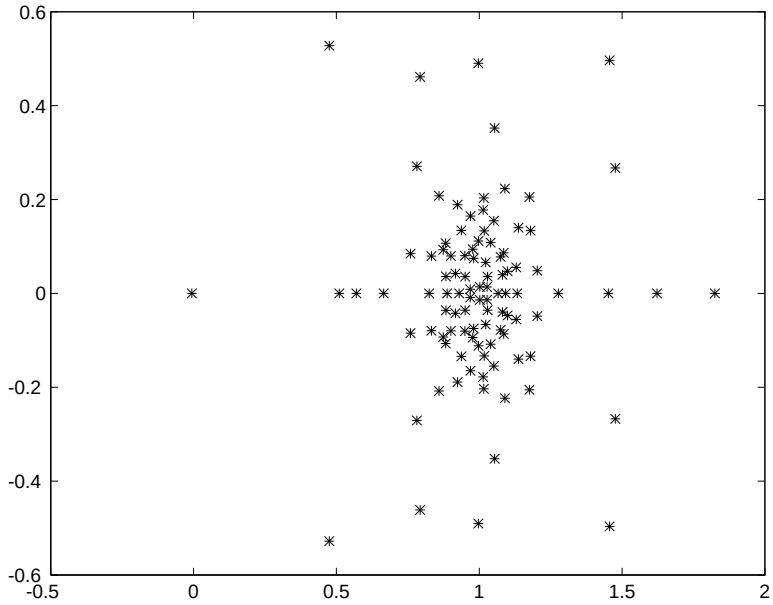


Fig. 3.2. Spectrum with preconditioning from example 3.1.



Example 3.2. The second matrix is the same as the first except the diagonal elements are also random. Unlike in example 3.1, the diagonal of the matrix is not a good approximation to the entire matrix. So we use a larger portion of the matrix as preconditioner. We let P be a band matrix with 49 diagonals of A, so $p_{ij} = a_{ij}$ if $|i - j| < 25$ and otherwise $p_{ij} = 0$. In figures 3.3 and 3.4, we give the spectrums of A and of $N = (P - \theta I)^{-1}(A - \theta I)$. Here $\theta = -11.16$ is an approximation to the leftmost eigenvalue of A, -11.1633. There is very little improvement in the relative separation. For both A and N, the ratio of distances to

closest and farthest eigenvalues is about 0.16. It is interesting that in this example, the inaccuracy of θ is magnified and N's smallest eigenvalue is not very near to zero. We can get some improvement by using a preconditioner with 99 diagonals of A. The ratio of distances to closest and farthest eigenvalues becomes 0.27, but this is with 75% of the entries of the matrix used in the preconditioner.

Fig. 3.3. Random matrix from example 3.2, no preconditioning.

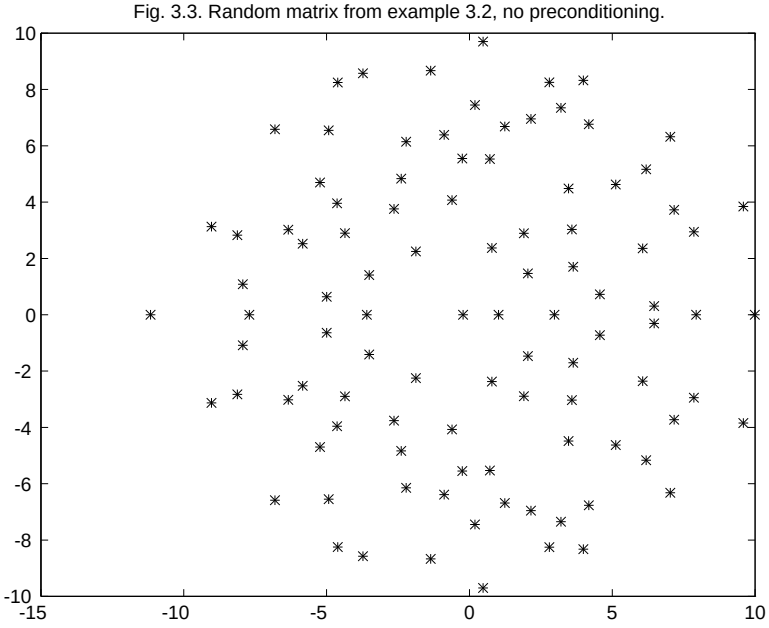
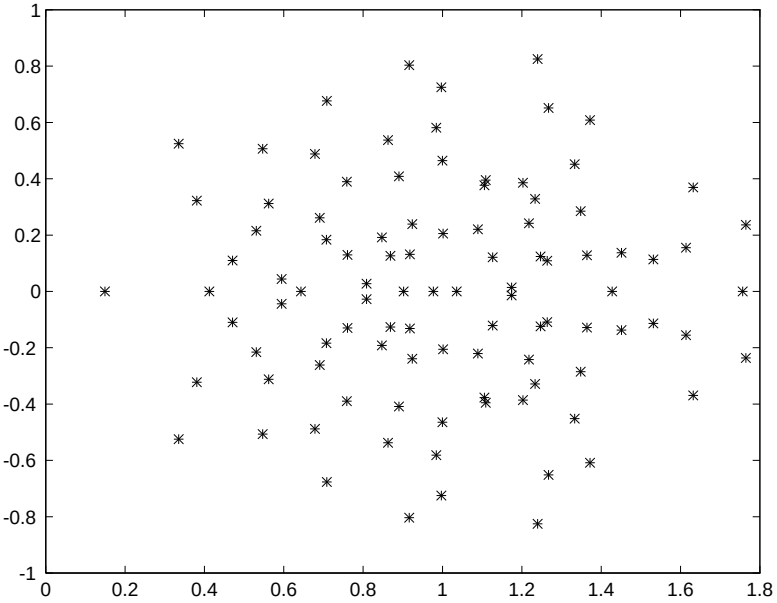


Fig. 3.4. Random matrix, 49 diagonals preconditioning.



The results in example 3.2 are not really surprising. It is already known for linear

equations that preconditioning a random matrix is difficult. Some structure is needed for effective preconditioning. It is interesting that some theoretical results can be established for preconditioning eigenvalues, just as they can for linear equations. See [17, 12] for results about convergence with modified incomplete factorization versus incomplete factorization for matrices from Poisson's equation, and for supporting experiments that show the advantage of modified incomplete factorization as the problem size increases.

Next, the differences between the methods are discussed. First, we compare GD with the other approaches, then discuss JD, compare RQI and JD, and finally give relationships and comparisons for JD and PL.

3.2 The GD method

GD changes its approximate eigenvalue at every iteration. This is an advantage in that it always uses the best information available. Based on this, it seems that GD will converge in less iterations. Another advantage for GD is that an SPD preconditioner is not required. However, it has the disadvantage of requiring more overhead expenses. Unlike the other methods, it cannot use an efficient Lanczos or CG algorithm. So it may require more CPU time. Restarting reduces the overhead expense but can also slow convergence. For difficult problems that need large subspaces, restarted GD can even end up needing more iterations than other methods. In cases where it is expensive to apply the matrix-vector product and the preconditioner, GD's overhead is not so important, and it may be the best method.

3.3 Jacobi-Davidson

The ideal JD operator in the outer loop is $((I - yy^T)(A - \theta I)(I - yy^T))^{-1}(I - yy^T)(A - \theta I)(I - yy^T)$. We discuss what the deflation of y accomplishes. Without deflation, this ideal operator become the identity matrix and is useless. Normally an approximation to the inverse is used, so the deflation may not be needed. However, it is reassuring to know that the method does not break down if an approximate inverse is too good. And there is a case where we can expect to have a good approximation. That is when the application of $(A - \theta I)^{-1}$ is accomplished by solving linear equations with an iterative method. Of course this is the approach of JD.

3.4 RQI and JD

The versions of RQI and JD that we used here are similar in that they both have a variant of the preconditioned conjugate gradient method in their inner iteration. However, JD has an advantage which we discuss quickly here. For more, see [10], where the shift-and-invert transform is compared to the Cayley transform. Let (λ, z) be the desired eigenpair, and suppose that θ is converging to λ. Assume that $(A-\theta I)^{-1}$ is being approximated. Then this approximation does not necessarily have an eigenvector converging to z, unless the accuracy increases as θ becomes more accurate. This greater accuracy corresponds to solving the linear equations in RQI to an increasing degree of accuracy (it may theoretically be possible to have an eigenvector converging to z without increasing the accuracy, with a different choice of starting vectors for the linear solver). Meanwhile, the approximation to $((I - yy^T)(A -$

$\theta I)(I-yy^T))^{-1}(I-yy^T)(A-\theta I)(I-yy^T)$ in JD does have an eigenvector that converges to z. This happens even if the accuracy of the approximation to $((I-yy^T)(A-\theta I)(I-yy^T))^{-1}$ does not improve. So in the JD method, the degree of accuracy needed in the solution of the linear equations is not so crucial.

JD also has the small eigenvalue removed from the conjugate gradient loop. This may improve convergence and stability compared to RQI.

3.5 PL and JD

We now consider PL and JD including how they relate to each other. The PL inner loop is inaccurate because of the effect of the preconditioning (for θ not yet equal to an eigenvalue, $M^{-1}(A-\theta I)$ does not have the same eigenvector as A). This necessitates restarting the loop and the restarting slows down the convergence. It does appear that the stopping test in the Lanczos loop is effective and minimizes the loss. Even with the early termination of the inner loop, the convergence is asymptotically quadratic with respect to the outer loop. The quadratic convergence indicates that the inner loop iteration can run longer as the method proceeds. So the efficiency compared to GD increases as it goes along.

Now with JD, the inner loop solves linear equations instead of an eigenvalue problem. So some effort potentially could be wasted if it goes into improving the linear equation solution in a way that does not eventually benefit the eigenvalue problem. And as with PL, some efficiency may be lost due to the restarts of the inner iteration. However, JD can give asymptotic cubic convergence with proper residual tolerances [6]. So when it is near convergence, its inner loop can be solved longer and still get a benefit.

There actually is a close equivalence between the problems solved in the inner loops of PL and JD. It is well known for an SPD matrix that these two tasks are approximately equivalent: 1) solving linear equations with the conjugate gradient method, 2) using the Lanczos algorithm to compute an eigenvalue *added* to the spectrum at zero (see for example [30]). They are equivalent in the sense that the same polynomial will be effective for both problems. This polynomial needs to be large at zero and small over the spectrum of A (the CG polynomial also needs to be normalized to one at zero). The comparison between PL and JD is somewhat similar, but instead of computing an eigenvalue added to the spectrum, an eigenvalue is removed from the linear equations spectrum. We look at the asymptotic case with θ equal to the eigenvalue λ, with eigenvector z. For PL, we need a polynomial large at the zero eigenvalue of $M^{-1}(A-\lambda I)$ and small at the other eigenvalues. For JD, the operator in the inner loop is $((I-zz^T)M(I-zz^T))^{-1}(I-zz^T)(A-\lambda I)(I-zz^T)$. This operator has the same spectrum as in the PL inner loop. The linear equations problem has zero removed from the spectrum, because of the deflation in this operator and the fact that the right-hand side for the linear equations is orthogonal to y (and asymptotically to z). So we need a polynomial that is one at zero and small over the rest of the spectrum of the JD inner loop operator. This is equivalent to the polynomial for PL. So asymptotically, the two methods solve problems of similar difficulty in their inner loops.

It is argued in [20] that PL is likely to converge to the smallest eigenvalue, although initial convergence can be very slow if there is neither a good starting vector nor an initial estimate for the smallest eigenvalue. Meanwhile, JD can get hung up and converge to the wrong eigenvalue. The inner loop of PL spreads out the spectrum of $M^{-1}(A-\lambda I)$ and

can compute a number of approximate eigenvectors (although only one or two are desired and generally only one is an accurate approximation to an eigenvector of A), while JD goes after just one vector in solving its linear equations. This one vector actually may improve approximations to several eigenpairs. Nevertheless, it seems that PL may have an advantage initially. See example 4.3. On the other hand, in this situation of a difficult start, some other method might be considered initially. A hybrid approach could begin with application of the Lanczos algorithm with either operator A or M^{-1}.

A major advantage for JD is in storage. PL must save all of the Lanczos vectors generated in its inner loop in order to form the Lanczos Ritz vector. But since JD has the conjugate gradient method in its inner loop, the storage demands are minimal. And the number of vectors saved for the outer Rayleigh-Ritz procedure can be kept small if necessary.

4 Experiments

In these tests, the number of matrix-vector products (mvp's) is listed. An application of the preconditioner accompanies each mvp. CPU time on a Vax is also given. However, the algorithms are not necessarily implemented optimally, so limited attention should be paid to the timings. For instance, the small eigenvalue problems in all methods are solved with Eispack routines [34]. So the solution could perhaps be more efficient, particularly for GD which solves similar problems at every iteration. The Lanczos algorithm does not use partial reothogonalization. For the Rayleigh-Ritz procedures in all methods, basis vectors are reorthogonalized when the vector's norm drops by 90% during the orthogonalization.

The GD method is restarted when subspaces reach dimension 20. JD has subspaces of maximum size 10 in its outer loop. GD generally needs larger subspaces than JD, because it builds its subspace with a weaker preconditioner than JD uses for its outer subspace. PL-RR restarts the outer Rayleigh-Ritz after an eigenvector is computed. To reduce the expense, just two eigenpairs are computed in the inner Lanczos loop of the PL methods. Also, the convergence in the inner loop is checked for the first seven Lanczos steps and then only at multiples of five.

Example 4.1. For testing, we choose a matrix A of dimension 5000. It is tridiagonal with entries 1, 2, 3, . . . 5000 on the main diagonal and with 0.5's in all the superdiagonal and subdiagonal positions. The starting vector has elements chosen randomly from the interval (-1,1). Two preconditioners of different accuracy are used. Both are fixed diagonal matrices. First $M = \text{Diag}(1.1, 1.2, 1.3, \ldots 500.9, 501)$ is a good preconditioner, then $M = \text{Diag}(1.002, 1.004, 1.006, \ldots, 10.998, 11)$ is a mediocre preconditioner. It is implicitly assumed in these preconditioners that we have 0.0 as an estimate for the desired eigenvalues. So we are using $M = D - 0.0I$ instead of $D - \theta I$. Both PL methods use $\gamma = 100$ to shift already converged eigenvectors [20]. We let JD use the initial estimate of 0.0 in place of θ for the first few outer iterations, because the poor starting vector gives a bad θ. This is switched back once θ improves (distance from 0.0 less than the eigenvalue residual norm). For this example, deflation of converged eigenvectors is not used in JD, because it does not significantly improve convergence.

The smallest eigenvalue and eigenvector are found first. Then the smallest five eigenvalues are computed. Tables 1 and 2 give the results with the two preconditioners. We see that

GD converges in the fewest iterations, but costs more.

There appears to be some inherent loss in going from GD to the double iteration of PL, at least for the good preconditioner case. PL-RR is not a big improvement upon PL. However, for the case of five eigenvalues with the mediocre preconditioner, PL-RR does come close to GD in mvp's.

RQI is the slowest method if several eigenvalues are computed. Performance of JD and PL is fairly similar. However, as mentioned earlier, JD uses less storage than PL. When using the poorer preconditioner, PL builds subspaces as large as 85 in the inner Lanczos loop. For tougher problems, even larger subspaces would be needed.

To show that GD can be the best in terms of time, we do an experiment with a similar, but less sparse matrix: 200 elements of value 0.1 are added to each row. We don't give a table, but do compare GD with PL for the good preconditioner. GD takes less time to compute 5 eigenvalues, 13.4 versus 21.8 seconds.

TABLE 1
Test Matrix, Good Preconditioner

	1 eigenvalue		5 eigenvalues	
	mvp's	cpu time	mvp's	cpu time
GD	26	1.2	87	4.9
PL	41	0.40	183	2.2
PL-RR	41	0.41	190	2.8
RQI	36	0.46	308	3.6
JD	38	0.47	150	2.6

TABLE 2
Test Matrix, Mediocre Preconditioner

	1 eigenvalue		5 eigenvalues	
	mvp's	cpu time	mvp's	cpu time
GD	164	8.6	641	36.2
PL	207	2.1	812	9.6
PL-RR	213	1.8	698	8.0
RQI	237	2.1	1428	11.4
JD	230	1.6	869	8.2

Example 4.2. In this example we demonstrate that the implementation does matter for a preconditioned method. Simply having the operator $M^{-1}(A - \theta I)$ does not guarantee effectiveness. RQI had some difficulty in the previous example when computing several eigenvalues, but we seek a more dramatic example. One of the simplest preconditioning methods for eigenproblems is to compute $M^{-1}(A-\theta I)y$ as in Davidson's method, but use it as a correction to y instead of applying Rayleigh-Ritz (see [8] for some other simple preconditioning methods). So this simple iteration starts with y and computes a new approximate eigenvector $y - M^{-1}(A - \theta I)y$, where θ is the Rayleigh quotient of y. This new approximation then becomes y. This method is tested on the matrix from example 1 with the

good preconditioner. The starting vector has first two components 100 and -50 and the rest random on (-1,1). The true eigenvector is $(.91, -.41, .095, -.015, .002, ...)^T$. The method did not converge. This is not surprising, since divergence can be expected if any eigenvalues of $I - M^{-1}(A - \lambda_1 I)$ are greater than one in magnitude.

Next we make the method a little more complicated. After $M^{-1}(A - \theta I)y$ is computed, a 2 by 2 Rayleigh-Ritz procedure is applied to combine y and $M^{-1}(A - \theta I)y$. So this is equivalent to GD with maximum size subspace of two. This approach did give convergence, even with the random starting vector from example 4.1. With the good preconditioner, the first eigenvalue was computed with 85 mvp's and 1.1 cpu seconds. This compares to 26 mvps and 1.2 seconds for GD with subspaces of dimension 20. With the mediocre preconditioner, there is a much bigger difference. The 2 by 2 method uses 3663 mvp's and 44 seconds compared to 164 mvp's and 8.6 seconds for GD (and only 1.6 seconds for JD). This is similar to large Krylov subspaces being better than either power method or dimension two Krylov subspaces. And large Krylov subspaces are especially important for tough problems.

Example 4.3. We compare PL-RR to JD with the assumption that there is neither an eigenvalue estimate available nor an extremely accurate eigenvector approximation. Three starting vectors are used. The first has all entries random on (-1,1). The second has first component 10 and the others random. The third has first entries 100 and -50, as mentioned in the previous example. Even though this last vector is better, its Rayleigh quotient is 293.3, not near the desired eigenvalue. The PL inner loop is limited to 20 steps, but a better stopping test is needed for the Lanczos loop in this situation of a poor starting vector. JD does not use 0.0 in place of θ initially, as in the earlier tests. The good preconditioner from example 4.1 is used, except it is shifted by θ, and absolute values are taken of negative elements. The results are given in Table 3. JD does not converge for the poorer vectors. It gets hung up searching for wrong eigenvalues. The same thing can happen to GD and RQI. PL separates out the eigenvectors in its inner loop and does not focus so much on the eigenvector nearest the current θ (see the discussion in subsection 3.5).

TABLE 3
Test Matrix, Good Prec., Different Starting Vectors

	(random)	(10, random)	(100,-50,random)
	mvp's	mvp's	mvp's
PL-RR	250	149	76
JD	-	-	39

Example 4.4. We next look at a situation where JD works better than PL. A bad preconditioner is chosen. It actually has a detrimental effect. PL approximately computes an eigenvector of $M^{-1}(A - \theta I)$. So if that eigenvector is distorted away from the eigenvector of A by the preconditioner, the method may have trouble until θ is very near λ. For JD, poor preconditioning may slow the solution of the linear equations, but does not change the vector that the method is trying to compute. To elaborate on this, we note that if both PL and JD solve their inner loops to full accuracy, JD is not affected by the distortion of the preconditioner. PL is affected.

We let the preconditioner be $M = LL^T$, where L is lower bidiagonal with 0.95's on the main diagonal and $1's$ on the subdiagonal. The matrix is the same as in the earlier examples, but because of the bad preconditioning, we use $n = 200$ instead of 5000. All inner loops are limited to 200 iterations. The starting vector is all random. To find the smallest eigenvalue, JD uses 1164 matrix-vector products. PL requires 3436. The bad preconditioning does have a stronger effect on PL.

Example 4.5. Next we run tests similar to those in Example 4.1, but with the matrix Sherman1 from the Harwell-Boeing collection [5]. The matrix is dimension 1000. The smallest eight eigenvalues are 0.00032, 0.00102, 0.00111, 0.00151, 0.00192, 0.00205, 0.00209 and 0.00210, and the largest is 5.04. So some of the small eigenvalues are close together. A good preconditioner is incomplete factorization [14] of A with no fill-in, and a mediocre preconditioner is diagonal preconditioning. Note that with incomplete factorization, only one factorization is done, and the matrix is not shifted before the factoring. So the estimated value 0.0 is assumed to be known for the smallest eigenvalues. JD again uses that estimate initially in place of θ. Here JD does deflate out converged eigenvectors. The PL methods use $\gamma = 0.01$ to remove converged eigenvectors. The results are given in Tables 4 and 5. Note that PL-RR and JD use similar numbers of matrix-vector products. Both require less iterations than GD for finding five eigenvalues with diagonal preconditioning. PL-RR performs significantly better than PL, so the outside Rayleigh-Ritz seems worthwhile. The RQI algorithm here works much better in this situation of close eigenvalues than the algorithm used in [20] (some eigenvalues were skipped). The outside Rayleigh-Ritz loop makes a big difference.

TABLE 4
Sherman1 matrix, Inc. Fact. Preconditioner

	1 eigenvalue		5 eigenvalues	
	mvp's	cpu time	mvp's	cpu time
GD	42	0.7	182	2.4
PL	66	0.4	467	1.5
PL-RR	66	0.4	365	1.5
RQI	99	0.5	616	2.1
JD	52	0.4	351	1.6

TABLE 5
Sherman1 matrix, Diagonal Preconditioner

	1 eigenvalue		5 eigenvalues	
	mvp's	cpu time	mvp's	cpu time
GD	528	5.3	2080	20.4
PL	473	1.2	2073	6.4
PL-RR	432	1.3	1537	4.7
RQI	614	1.7	3107	7.0
JD	522	1.4	1541	4.2

5 Conclusion

We have discussed that preconditioning methods are generally related by their use of the same operator, $M^{-1}(A-\theta I)$. So convergence rates are often similar, but the implementation can make a difference.

We give some conclusions about the methods compared in this paper. GD is expensive compared to the other methods when matrix-vector products are cheap. JD and RQI require the least memory. PL-RR is likely to use the most memory. JD and PL-RR usually have similar performance for the examples discussed here. However, PL-RR is more sensitive to a bad preconditioner than JD. And JD is less robust than PL-RR when the starting vector is poor.

We cannot recommend just one method as being best. However, JD is reliable and efficient if the starting vector is good. And an initial subspace can be computed for JD with another method. PL-RR is a possible initial method, but perhaps better would be a Lanczos method using operator A or M^{-1}, since PL is not immune to initial problems [15, pp. 99-101]. We also note that GD is likely best if the matrix-vector product is quite expensive.

While this paper deals with the symmetric case, much of the discussion carries over for nonsymmetric problems. It would be interesting to continue the comparisons for that case, and for interior eigenvalue problems and generalized eigenvalue problems.

Acknowledgements: The author wishes to thank Henk van der Vorst for inviting this paper. Also thanks to Henk and Zhaojun Bai for their work on the Eigenvalue Templates project which helped lead to this paper. The anonymous referees understood the methods in this paper very well. They made many suggestions for improving the paper.

References

[1] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951.

[2] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, PA, USA, 1994.

[3] M. Crouzeix, B. Philippe, and M. Sadkane. The Davidson method. *SIAM J. Sci. Comput.*, 15:62–76, 1994.

[4] E. R. Davidson. The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *J. Comput. Phys.*, 17:87–94, 1975.

[5] I. S. Duff, R. G. Grimes, and J. G. Lewis. Sparse matrix test problems. *ACM Trans. Math. Soft.*, 15:1–14, 1989.

[6] S. C. Eisenstat and H. F. Walker. Choosing the forcing terms in an inexact Newton method. *SIAM J. Sci. Comput.*, 17:16–32, 1996.

[7] R. W. Freund and N. M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, 60:315–339, 1991.

[8] A. Knyazev. Preconditioning eigensolvers. In *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, 2000. To appear.

[9] C. Lanczos. An iterative method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Standards*, 45:255–282, 1950.

[10] R. B. Lehoucq and K. Meerbergen. Using generalized Cayley transformations within an inexact rational Krylov sequence method. *SIAM J. Matrix Anal. Appl.*, 20:131–148, 1998.

[11] J. G. Lewis. *Algorithms for Sparse Matrix Eigenvalue Problems*. PhD thesis, Stanford University, Stanford, CA, 1977.

[12] K. Meerbergen and R. B. Morgan. Inexact methods. In *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, 2000. To appear.

[13] K. Meerbergen and D. Roose. The restarted Arnoldi method applied to iterative linear system solvers for the computation of rightmost eigenvalues. *SIAM J. Matrix Anal. Appl.*, 18:1–20, 1997.

[14] J. A. Meijerink and H. A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.*, 31:148–162, 1977.

[15] R. B. Morgan. *Preconditioning Eigenvalue Problems*. PhD thesis, University of Texas, Austin, TX, 1986.

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

[17] R. B. Morgan. Theory for preconditioning eigenvalue problems. *Proceedings of the Copper Mountain Conference on Iterative Methods*, 1990.

[18] R. B. Morgan. Generalizations of Davidson's method for computing eigenvalues of large nonsymmetric matrices. *J. Comput. Phys.*, 101:287–291, 1992.

[19] R. B. Morgan and D. S. Scott. Generalizations of Davidson's method for computing eigenvalues of sparse symmetric matrices. *SIAM J. Sci. Statist. Comput.*, 7:817–825, 1986.

[20] R. B. Morgan and D. S. Scott. Preconditioning the Lanczos algorithm for sparse symmetric eigenvalue problems. *SIAM J. Sci. Comput.*, 14:585–593, 1993.

[21] R. B. Morgan and M. Zeng. Harmonic projection methods for large non-symmetric eigenvalue problems. *Num. Lin. Alg. with Appl.*, 5:33–55, 1998.

[22] B. N. Parlett. *The Symmetric Eigenvalue Problem*. Prentice-Hall, Englewood Cliffs, N.J., 1980.

[23] A. Ruhe. SOR-methods for the eigenvalue problem with large sparse matrices. *Math. Comp.*, 28:695–710, 1974.

[24] A. Ruhe. Iterative eigenvalue algorithms based on convergent splittings. *J. Comput. Phys.*, 19:110–120, 1975.

[25] A. Ruhe and T. Wiberg. The method of conjugate gradients used in inverse iteration. *BIT*, 12:543–554, 1972.

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

[27] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing, Boston, MA, 1996.

[28] Y. Saad and M. H. Schultz. GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7:856–869, 1986.

[29] G. Sleijpen and H. van der Vorst. Jacobi-Davidson methods. In *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, 2000. To appear.

[30] G. L. G. Sleijpen and A. van der Sluis. Further results on the convergence behavior of conjugate-gradients and Ritz values. *Linear Algebra Appl.*, 246:233–278, 1996.

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

[32] G. L. G. Sleijpen and H. A. van der Vorst. Preconditioning for eigenvalue correction equations. Technical Report in preparation, Utrecht University, Utrecht, The Netherlands, 1999.

[33] G. L. G. Sleijpen, H. A. van der Vorst, and E. Meijerink. Efficient expansion of subspaces in the Jacobi-Davidson method for standard and generalized eigenvalue problems. *ETNA*, 7:75–89, 1998.

[34] B. T. Smith, J. M. Boyle, Y. Ikebe, B. C. Klema, and C. B. Moler. *Matrix Eigensystems Routines: EISPACK Guide*. Springer-Verlag, New York, NY, 1970.

[35] D. C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13:357–385, 1992.

[36] D. C. Sorensen and C. Yang. A truncated RQ iteration for large scale eigenvalue calculations. *SIAM J. Matrix Anal. Appl.*, 19:1045–1073, 1998.

[37] A. Stathopoulos, Y. Saad, and C. Fischer. Robust preconditioning of large, sparse, symmetric eigenvalue problems. *J. Comput. Appl. Math.*, 64:197–215, 1995.

[38] A. Stathopoulos, Y. Saad, and K. Wu. Dynamic thick restarting of the Davidson, and the implicitly restarted Arnoldi methods. *SIAM J. Sci. Comput.*, 19:227–245, 1998.

[39] D. B. Szyld. Criteria for combining inverse and Rayleigh quotient iteration. *SIAM J. Numer. Anal.*, 6:1369–1375, 1988.

[40] H. A. van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 12:631–644, 1992.