# A CASE FOR A BIORTHOGONAL JACOBI-DAVIDSON METHOD: RESTARTING AND CORRECTION EQUATION *

ANDREAS STATHOPOULOS †

**Abstract.** We propose a biorthogonal Jacobi-Davidson method (biJD), which can be viewed as an explicitly biorthogonalized, restarted Lanczos method, that uses the approximate solution of a correction equation to expand its basis. Through an elegant formulation, the algorithm allows for all the functionalities and features of the Jacobi-Davidson (JD), but it also includes some of the advantages of the nonsymmetric Lanczos.

The main motivation for this work stems from a correction equation and a restarting scheme that are possible with biJD but not with JD. Specifically, a correction equation using the left approximate eigenvectors available in biJD yields cubic asymptotic convergence, as opposed to quadratic with the JD correction equation. In addition, a successful restarting scheme for the symmetric JD depends on the Lanczos three term recurrence, and thus can only apply to the biJD. Finally, methods that require a multiplication with the adjoint of the matrix need to be reconsidered on today's computers with memory hierarchies, as this multiplication can be performed with minimal additional cost.

We describe the algorithm, its features, and the possible functionalities. In addition, we develop an appropriate correction equation framework, and analyze the effects of the new restarting scheme. Our numerical experiments confirm that biJD is a highly competitive method for a difficult problem.

**Key words.** Jacobi-Davidson, BCG, eigenvalues, Lanczos, three-term recurrence, preconditioning

**AMS Subject Classification.** 65F15

**1. Introduction.** The solution of the large eigenvalue problem $A\tilde{x} = \tilde{\lambda}\tilde{x}$ for a few eigenvalues closest to a given value $\sigma$ and their corresponding eigenvectors (together eigenpairs) is recognized as a harder problem than the solution of a linear system of equations with $A$. Because the eigenvalues are not known a priori, the system to be solved is non linear [38, 37]. Even if the eigenvalue were known, the resulting linear system would be indefinite for any eigenvalue that lies inside the spectrum. This usually implies slow convergence of the linear solver, and moreover it is hard to obtain good preconditioners [27, 1]. Non normality and ill conditioning exacerbate these problems further.

Preconditioning is also not straightforward to apply on eigenvalue iterative solvers. Early attempts included variants of the Davidson's method [9] and shift and invert methods [23], but Jacobi-Davidson (JD) type methods have provided an appropriate preconditioning framework for eigensolvers [30].

Another important problem with eigensolvers is their high storage requirements. For linear systems storage is less of an issue, because three term recurrence methods, such as CG and BCG, are as effective as full orthogonalization Arnoldi-type methods. In contrast, the three term recurrence Lanczos method for eigenproblems needs to store the basis vectors to recover eigenvector approximations. Moreover, most eigenvalue methods that use preconditioning do not build a Krylov space, and thus Arnoldi-type methods, like JD, are necessary [9, 30, 26].

For all the above reasons, the more complicated and expensive per step methods of Arnoldi and JD are usually preferred to the computational simplicity of the Lanczos method. The JD method can be viewed as an inner-outer method. At each outer step,

*Work supported by the College of William and Mary

†Department of Computer Science, College of William and Mary, Williamsburg, Virginia 23187-8795, (andreas@cs.wm.edu).

JD incorporates into the basis an approximate solution of the correction equation:

$$(I - xx^*)(A - \sigma I)(I - xx^*)\ \delta = (I - xx^*)(\mu I - A)x, \tag{1.1}$$

where $(\mu, x)$, with $\|x\| = 1$, is an approximation of the required eigenpair. Typically, a few steps of a Krylov iterative solver are applied to eq. (1.1), and available preconditioners for $A$ can be used through a technique described in [30, 12, 29].

Despite its improved convergence, for many hard problems the JD may still require a large number of steps, with overwhelming storage requirements. This problem is controlled by restarting the method when the basis size reaches a user specified upper limit. Because the JD basis is not required to be Krylov, a variety of restarting techniques can be used, including implicit restarting with various shift strategies [32], and thick restarting which, at restart, retains either Ritz vectors [35], or Schur vectors [12]. A host of other improvements on targeting eigenvalues, harmonic Ritz approximations, preconditioning techniques, and extensions to the generalized eigenvalue problem have helped make the JD a robust and widely used method [12, 29].

Yet, the non symmetric Lanczos method has two very appealing characteristics that could offer significant advantages, if exploited in a JD framework. The Lanczos maintains both a left and a right biorthogonal bases, generated by $A^*$ and $A$ respectively. As a result, approximations for the left eigenvectors are also obtained, and biorthogonality is implicitly maintained through a three term recurrence.

The availability of approximations to left eigenvectors suggests a natural alternative correction equation, based on an approximate spectral projector:

$$(I - xy^*)(A - \sigma I)(I - xy^*)\ \delta = (I - xy^*)(\lambda I - A)x, \tag{1.2}$$

where $(\lambda, x)$ is an approximation to the right eigenpair of $A$, and $y$ is an approximation to the corresponding left eigenvector, such that $\|x\| = 1$ and $y^*x = 1$. An interesting observation is that we can solve both eq. (1.2), and its conjugate transpose by a single BCG iteration, improving simultaneously both left and right eigenpairs. Performing inverse iteration with these two conjugate matrices is known to converge cubicly [37], and recently Sleijpen et al. proved the same convergence rate for the JD with the above correction equation [29]. However, for the latter to hold, $y$ must converge to the left eigenvector, which does not hold in general if only a right space is considered.

The left and right biorthogonal bases also suggest an effective restarting scheme. Restarting has significant performance shortcomings, since important components of the invariant subspace may be discarded. Restarting techniques attempt to identify and retain these important components to help future convergence. One class of techniques achieves this by retaining certain Ritz vectors that tend to improve convergence towards the desired eigenpair in a deflation-like way [35, 7, 12]. Another class traces the problem to the orthogonality lost when restarting, and tries to reinstate it by keeping those vector directions against which the basis tends to lose orthogonality [10, 11]. Because the Lanczos method maintains biorthogonality implicitly through the three term recurrence, it is natural to ask whether these recurrence vectors can be used in restarting. For the symmetric case, this idea has been explored in restarting the JD with impressive results [20, 34, 33]. For the nonsymmetric case, however, a short term recurrence is not possible for the JD and Arnoldi. A Lanczos based method seems to be the only alternative.

In this paper, we propose a biorthogonal Jacobi-Davidson method (biJD), which combines the Lanczos two sided iteration with the solution of the correction equation 1.2 for both left and right Ritz pairs. The goal is twofold: provide a faster converging algorithm, and exploit an effective restarting scheme. First, we describe the biJD algorithm and review its various advantageous features. We also show how the multiplication of the adjoint with a vector can be performed with minimal computational cost, which is of more general interest. In the second part, we examine how the correction equation (1.2) can be set up, how fast it converges, and how to use preconditioning with it. In the third part, we adapt the new restarting idea to biJD, and show why it is expected to be beneficial. We conclude with numerical examples and a summarizing discussion.

## 2. The biorthogonal Jacobi-Davidson (biJD).

**2.1. The algorithm.** Throughout this paper, we assume that the matrix $A$ is nonsymmetric, diagonalizable of order $N$, with eigenpairs $(\tilde{\lambda}_i, \tilde{x}_i)$, of which the one closest to a complex value $\sigma$ is sought. The Davidson method first appeared as a diagonally preconditioned version of the Lanczos method for the symmetric eigenproblem. Extensions, to both general preconditioners and to the nonsymmetric case have been given since [18, 8]. Morgan and Scott [19] proposed to solve approximately with some preconditioner the generalized Davidson correction equation: $(A - \sigma I)\ \delta = r = (A - \mu I)x$. In [30], Sleijpen et al. show that for stability, robustness, as well as efficiency, the operator in the correction equation should have a range orthogonal to $x$, yielding eq. (1.1). Several extensions have been proposed for the JD method, including general projections for eq. (1.1), restarting schemes, and the use of harmonic Ritz vectors for interior eigenpairs [12, 29, 34]. The convergence analysis of using eq. (1.2) in the JD is described in [29], but the formulation of a two sided biJD method appears to be new. Algorithm (2.1) outlines the basic steps of biJD, and introduces most of the notation in this paper.

We focus on finding only one eigenpair closest to $\sigma$, but the extension to finding more eigenpairs is straightforward. When an eigenpair converges, it can stay in the basis and never be targeted again, or it can be locked out of the basis [25]. If a preconditioner $M \approx A$ is known, we can apply it on the BCG iteration, but the details are discussed later. Also in the BCG, when $\lambda$ is far from $\sigma$, we solve eq. (1.2) instead, and we only need to apply one of the projections at every BCG iteration. When the basis reaches a maximum size of $m$, thick restarting is performed by keeping the $k$ Ritz vectors closest to the shift $\sigma$. The above algorithm uses complex arithmetic, but as with JD, a real-only version is also possible [12]. Finally, a block biJD based on the Davidson-Liu block variant can be developed if more than one Ritz pairs are targeted at each step.

Computationally, the biJD is more expensive per step than the JD. First, it requires a matrix vector multiplication with $A^*$, which may not be available in some applications. Second, in step 3, each iteration of the BCG performs two matrix-vector multiplications, one with $A$ and one with $A^*$. Similarly at step 6, the update of the auxiliary matrices $K$ and $L$ requires two such multiplications, twice the number required by JD. However, even the JD would require two multiplications per inner iteration, if a short recurrence method, such as BCGSTAB or QMR were used. Finally, the biJD requires twice as much storage as the JD, because of the left space $W$ and its image $L = A^*W$. If needed, we can save the space of the arrays $K$ and $L$ by computing the residuals at step 2 by explicit matrix-vector multiplications.

**2.2. Computational efficiency.** From the above discussion the biJD seems to incur about twice as many floating point operations per outer step than the JD, when the latter is using GMRES rather than BCGSTAB for the correction equation.

ALGORITHM 2.1. *BiJD*

*Input:* $\sigma$: *a complex value,* $m$: *maximum size allowed for the bases.*
$k$: *the number of vectors to be retained at restart.*
*Initial right and left spaces:* $V = [v_1, \ldots, v_{k_0}]$ *and* $W = [w_1, \ldots, w_{k_0}]$,
*with* $W^*V = I$, *and* $\|v_i\| = 1$, $i = 1, \ldots, k_0$.
*Output: Finds an approximate eigenpair* $(\lambda, x)$ *of* $A$, *with* $\lambda$ *closest to* $\sigma$.

*Compute initial* $K = AV$, $L = A^*W$, *and* $H = W^*AV$. *Set* $j = k_0$.
**Repeat**
1. *Compute right* $(g_i)$ *and left* $(f_i)$ *eigenvectors of* $H$,
   *with* $\|g_i\| = 1, f_i^* g_i = 1$, $i = 1, ..., j$
2. *Target the Ritz triplet* $(f, g, \lambda)$ *with Ritz value* $\lambda$ *closest to* $\sigma$:
   $x = Vg$ *with residual* $r_r = Kg - \lambda x$ *(right Ritz pair)*
   $y = Wf$ *with residual* $r_l = Lf - \bar{\lambda} y$ *(left Ritz pair)*
3. *Run* $p$ *steps of BCG simultaneously on the two correction equations:*
   $(I - xy^*)(A - \lambda I)(I - xy^*)\ \delta_r = r_r$
   $(I - yx^*)(A^* - \bar{\lambda} I)(I - yx^*)\ \delta_l = r_l$
4. *Set* $V = [V, \delta_r]$, *and* $W = [W, \delta_l]$
5. *Biorthogonalize the new basis vectors such that* $W^*V = I$, *and* $\|v_{j+1}\| = 1$
6. *Set* $j = j + 1$, *and compute* $K_j = Av_j$ *and* $L_j = A^*W_j$
7. *Compute the last column and row of the matrix* $H = W^*AV$

**Until** $(\|r_r\| < tolerance)$ **or** $(j == m)$
**If** $(\|r_r\| \geq tolerance)$ **then**
8. *Compute* $k < m$ *current Ritz vectors and restart:*
9. *set* $V = [x_1, \ldots, x_k]$, $W = [y_1, \ldots, y_k]$, *and* $H = \mathrm{diag}(\lambda_1, \ldots, \lambda_k)$
10. *set* $j = k$, *and goto step 1.*

**endif**

However, in todays multiple memory hierarchy computers the role of memory accesses is more relevant than the flops. In this context, the biJD can be performed with only a slight computational overhead over the JD.

The fact that the matrix vector multiplications with $A$ and with $A^*$ can be performed with only one access to the matrix $A$ has not received any attention in the literature. Let us assume for simplicity a compressed sparse row (CSR) storage of the matrix $A$ [27, 24]. For BCG-like methods two approaches are traditionally discussed [5]. The first performs the multiplication with $A$ using the code in fig. 2.1(a), and then with $A^*$ using the code in fig. 2.1(b). However, the CSR data structure increases memory traffic for the latter operation. The second approach explicitly transposes $A$ into a CSR stored matrix $A^*$. Then, it applies matrix-vector multiplications for both matrices using the code in fig. 2.1(a). Besides the extra storage, still two matrices are read from memory at each operation.

A first improvement would be to perform both $Ax$ and $A^*u$ while the same row of $A$ has been brought in from memory. The code in fig. 2.2(a) shows how this is performed by simply merging the codes of fig. 2.1. Each sparse row of $A$ is brought in once (both `a` and `ja`), and it is used to accumulate an inner product and to update various elements of $w = A^*u$. Depending on the cache size and the read/write channels available on the computer, this code can significantly reduce execution time.

Despite the locality of the array `a`, the vector `x` and the result vector `w` are

(a)

```
%% A x, A in CSR
for i = 1,n
  t = 0
  for k=ia(i), ia(i+1)-1
    t = t + a(k)*x(ja(k))
  end
  y(i) = t
end
```

(b)

```
%% A^T x, A in CSR
for i=1,n
  w(i) = 0
end
for i = 1,n
  for k=ia(i), ia(i+1)-1
    w(ja(k)) = w(ja(k)) + u(i)*a(k)
  end
end
```

FIG. 2.1. *Traditional methods for matrix-vector multiplication when $A$ is stored in CSR format. (a): $y = Ax$, (b): $w = A^*u$.*

(a)

```
%% Both Ax and A^Tx, A in CSR
for i=1,n
  w(i) = 0.0
end
for i = 1,n
  t = 0
  for k=ia(i), ia(i+1)-1
    ix = ja(k)
    w(ix) = w(ix) + u(i)*a(k)
    t = t + a(k)*x(ix)
  end
  y(i) = t
end
```

(b)

```
%% Ax and A^Tx, using temp vector
for i = 1,n
  Tmp(2*i-1) = x(i)
  Tmp(2*i)   = 0
  y(i)       = 0
end
for i = 1,n
  t = 0
  for k=ia(i), ia(i+1)-1
    ix = 2*ja(k)
    Tmp(ix) = Tmp(ix) + u(i)*a(k)
    t = t + a(k)*Tmp(ix-1)
  end
  y(i) = t
end
for i = 1,n
   w(i) = Tmp(2*i)
end
```

FIG. 2.2. *Proposed methods that perform the two matrix-vector multiplications simultaneously. (a): $y = Ax$ and $w = A^*u$ (b): $y = Ax$ and $w = A^*u$ but using a temporary vector.*

accessed in a non local, but identical pattern. The idea of the code in fig. 2.2(b), is to create a temporary vector `Tmp` with the elements of `x` and `w` interleaved in it. With this scheme, the two non local accesses to $x$ and $w$ become one non local access to `Tmp` with the second access being in the adjacent memory location. If the number of nonzero elements in the matrix is large, and if the machine can perform both a read and a write on `Tmp` efficiently, this modification can provide further reduction in execution time. Note that the overhead from the initialization of the arrays can be hidden, if these initializations are embedded in the calling BCG function. Finally, in contrast to BCGSTAB-like methods, the two matrix vector multiplications can be performed in parallel. Tables 2.1 and 2.2 present some timing results with g77 compiler on a SUN Ultra 2300 with 1 MB cache, and on a 400 MHz Pentium III with 512 KB cache respectively. On both machines we see that the proposed algorithms consistently improve execution time by about 25%. We expect bigger improvements with advanced optimizing compilers.

Another computational requirement that seems to limit the biJD applicability is the storage of $K$ and $L$. In practice this turns out to be a minor problem for several

| Method | matrix size $N$ / nonzero elements per row $nz$ 50000/30 | 50000/10 | 200000/15 | 200000/5 | 400000/5 |
|---|---|---|---|---|---|
| 2.1(a-b) | 0.28 | 0.10 | 0.63 | 0.25 | 0.52 |
| 2.1(a) with $A^*$ | 0.27 | 0.10 | 0.63 | 0.25 | 0.52 |
| 2.2(a) | 0.39 | 0.06 | 0.40 | 0.16 | 0.34 |
| 2.2(b) | 0.19 | 0.07 | 0.46 | 0.22 | 0.44 |

TABLE 2.1

*Time in seconds to execute the two matrix vector multiplications $y = Ax$ and $w = A^*u$, where $A$ is a matrix of size $N$, and with $nz$ nonzero elements per row randomly placed. The method numbers refer to the algorithms in figures 2.1-2.2. The machine is a SUN Ultra 2300.*

| Method | matrix size $N$ / nonzero elements per row $nz$ 50000/150 | 50000/10 | 200000/30 | 400000/40 | 800000/20 |
|---|---|---|---|---|---|
| 2.1(a-b) | 1.27 | 0.09 | 1.07 | 2.78 | 2.84 |
| 2.2(a) | 0.88 | 0.06 | 0.71 | 1.91 | 1.91 |
| 2.2(b) | 0.87 | 0.07 | 0.79 | 2.07 | 2.22 |

TABLE 2.2

*Time in seconds to execute the two matrix vector multiplications $y = Ax$ and $w = A^*u$, where $A$ is a matrix of size $N$, and with $nz$ nonzero elements per row randomly placed. The method numbers refer to the algorithms in figures 2.1-2.2. The machine is a 400 MHz Pentium III.*

reasons. First, the limiting factor is the expensive orthogonalization procedure, and for this reason the basis size is not allowed to grow very large. Second, with ever decreasing memory prices, storage for this limited basis size is not an issue, unlike the Lanczos process where hundreds or even thousands of vectors might be needed. Third, the availability of good preconditioners, and in particular the advanced restarting techniques that we propose allow the basis size to shrink even further without significant convergence deterioration.

Finally, for computational efficiency on cache-based and parallel computers, we use an iterative Gram-Schmidt biorthogonalization. When there is no preconditioner and the number of BCG steps equals 1, the method reduces to a stable implementation of the restarted nonsymmetric Lanczos [28].

**2.3. Features of the biJD.** Besides the attractive properties of the biJD for restarting and the correction equation, there is a host of features that enhance the overall performance and robustness of the algorithm.

**2.3.1. Benefits from the left/right bases.** The intrinsic advantage of the biJD is its ability to obtain the left eigenvectors almost for free, and to similar accuracy as the right ones. Left eigenvectors can be extremely useful, even if they are not specifically needed by the application. First, they can be used in the spectral projector to deflate converged eigenpairs. Second, left and right Ritz pairs provide an estimate to the condition number of the required eigenvalue, which is a measure of how reliably this eigenpair has been computed. Third, even before convergence is achieved, detecting an ill conditioned eigenvalue might help speed up the correction equation by approximately removing this ill conditioning through a similarity transformation.

Another significant advantage of the biJD is that the Ritz values are the generalized Rayleigh quotients (GRQ): $\lambda = y^*Ax/(y^*x)$. If the eigenvalue $\tilde{\lambda}$ is not too ill conditioned, the GRQ is known to be more accurate than the Rayleigh quotient (RQ): $\mu = x^*Ax/(x^*x)$. In fact, this is true even when the left eigenvectors are known to a lesser accuracy than the right ones. As explained in [37] (see also [4]), let $x, y$ be approximations to the right and left eigenvectors, with $x = \tilde{x} + \epsilon_x$ and $y = \tilde{y} + \epsilon_y$. If we assume for simplicity that $\epsilon_x \perp \tilde{y}$ and $\epsilon_y \perp \tilde{x}$, then,

$$\begin{aligned}
&\frac{y^*Ax}{y^*x} = \tilde{\lambda} + \frac{\epsilon_y^*(A-\tilde{\lambda})\epsilon_x}{\tilde{y}^*\tilde{x} + \epsilon_y^*\epsilon_x} \qquad \Rightarrow \\
&|\lambda - \tilde{\lambda}| \leq \left(\|A\| + |\tilde{\lambda}|\right) \frac{\|\epsilon_y\|\|\epsilon_x\|}{|\tilde{y}^*\tilde{x}| - \|\epsilon_x\|\|\epsilon_y\|},
\end{aligned} \tag{2.1}$$

which implies that the error in the Ritz value is $\mathcal{O}(\|\epsilon_y\|\|\epsilon_x\|)$, provided that the eigenvalue is not too ill conditioned. Interestingly, if we substitute $x = \tilde{x} + \epsilon_x$ with $\epsilon_x \perp x$ in the RQ, the term $\tilde{x}^*A\epsilon_x$ is not zero, unless the matrix is normal. Thus, assuming $\|\tilde{x}\| = 1$, the RQ is given by:

$$\begin{aligned}
&\frac{x^*Ax}{x^*x} = \tilde{\lambda} + \frac{\epsilon_x^*(A-\tilde{\lambda})\epsilon_x + \tilde{x}^*A\epsilon_x}{\|\tilde{x}\|^2 + \|\epsilon_x\|^2} \qquad \Rightarrow \\
&|\mu - \tilde{\lambda}| \leq \left(\|A\| + |\tilde{\lambda}|\right) \frac{\|\epsilon_x\|^2}{1 + \|\epsilon_x\|^2} + \frac{\|A\|\|\epsilon_x\|}{1 + \|\epsilon_x\|^2}.
\end{aligned} \tag{2.2}$$

Thus, the error in the RQ is $\mathcal{O}(\|\epsilon_x\|)$ in general, and $\mathcal{O}(\|\epsilon_x\|^2)$ in the normal case.

Moreover, the RQ of a vector can be close to the required eigenvalue even though the vector is a linear combination of unrelated eigenvectors. For normal matrices, this is common when $\sigma$ is in the interior of the spectrum, and for nonnormal matrices it is also possible for the exterior ones. In such cases, however, the GRQ and RQ differ substantially. Because the RQ can be computed inexpensively in the biJD, it provides an excellent means of assessing eigenvalue convergence.

For nonsymmetric matrices, neither the Galerkin nor the Petrov-Galerkin projection methods provide any useful optimality for the Ritz pairs [25]. It has been observed that sometimes approximations are extracted faster and more accurately from the Lanczos process than from an orthogonal projection method (like Arnoldi), but also the contrary is often true. The biJD inherits these characteristics, which for some problems may prove advantageous over the JD. However, differences solely because of the Petrov-Galerkin are expected to be minor because of the use of preconditioning and restarting.

**2.3.2. Flexibility of the biJD algorithm.** The biJD algorithm uses explicit biorthogonalization, thus avoiding problems from the loss of orthogonality of nonsymmetric Lanczos. More interestingly, it also retains the flexibility of JD. For example, it can accommodate a variety of restarting techniques because it does not have to maintain a tridiagonal projection matrix. In our description of the algorithm, we have used thick restarting where the $k$ Ritz pairs closest to $\sigma$ are retained. The left and right spaces facilitate an elegant extension of JD thick restarting to biJD, since left and right Ritz vectors are biorthogonal by construction, and $H = (y_i^*Ax_j)_{i,j}$ is the diagonal of the corresponding Ritz values. Implicit restarting with user defined shifts can also be applied in a way similar to the implicitly restarted nonsymmetric Lanczos [28], but the benefits in the absence of a Krylov space are not clear.

The biJD can also restart with any arbitrary vectors $Vc \in V$ and $Wz \in W$. Biorthogonality can be maintained inexpensively in the coefficient space, by biorthogonalizing the vectors $c$ and $z$ instead, and $H$ can be updated by inner products of the coefficient restarting vectors (see [34]). This flexibility is used in the restarting scheme proposed in a later section, and it is also useful with harmonic Ritz vectors.

When looking for interior eigenpairs, harmonic Ritz vectors often provide better approximations and may result in a more effective correction equation [6, 21, 12, 33]. The main idea is to perform a Petrov Galerkin on the matrix $(A - \sigma I)^{-1}$, for which the required eigenpairs lie on the extreme of its spectrum. The inversion of the matrix is avoided if the space $(A - \sigma)V$ is used instead in the projection. To compute the harmonic pairs for the biJD, we proceed similarly to the JD, with the exception that we modify both left and right projection spaces:

$$\begin{aligned} W_h &= (A - \sigma I)^* W = A^* W - \bar{\sigma} W = L - \bar{\sigma} W, \\ V_h &= (A - \sigma I) V = AV - \sigma V = K - \sigma V. \end{aligned}$$

The $W_h$ and $V_h$ can be computed without matrix vector multiplications. Moreover,

$$W_h^* V_h = W^* (A - \sigma I)^2 V,$$

and we can formulate the Petrov Galerkin projection with $W_h$ and $V_h$ solving for $g_h$:

$$\begin{aligned} W_h^* (A - \sigma I)^{-1} V_h g_h &= \frac{1}{\nu} W_h^* V_h g_h \qquad \Leftrightarrow \\ W^* (A - \sigma I) V g_h &= \frac{1}{\nu} W^* (A - \sigma I)^2 V g_h. \end{aligned} \tag{2.3}$$

Equation (2.3) is similar to the one for the JD iteration, and it involves computations with only $V, W, K$ and $L$. As with JD, to obtain the harmonic Ritz vectors we apply implicitly one step of inverse iteration to $V_h g_h = (A - \sigma I) V g_h$, which yields the vector $V g_h$. The Rayleigh quotient of the harmonic Ritz vector can also be compared with the RQ and GRQ of the Ritz vector as an additional convergence estimator.

Finally, the biJD can incorporate into its bases any arbitrary vector in $\mathcal{C}^N$ that carries useful information. Both a left and a right vector would be needed, so the user must guarantee that they are not orthogonal. The vectors are appended in the bases, biorthogonalized, and the algorithm resumes. Besides allowing for external information to be used, this feature allows for the flexible preconditioning required in the biJD/JD methods, but more importantly it provides a straightforward way of dealing with breakdown.

**2.3.3. Resolving breakdown.** Breakdown can occur in biJD whenever the two vectors added in the left and right space are orthogonal. For the nonsymmetric Lanczos method, sophisticated look ahead schemes have been developed to deal with this problem [22, 13]. A simple alternative is to restart the Lanczos method with a slightly modified residual vector [27], or to perform an implicit restarting with "non-exact" shifts (i.e., non Ritz values) [28]. Note that if we use exact shifts, or equivalently, if we thick restart with the current Ritz vectors, the breakdown will reoccur immediately after restarting [28]. These techniques for avoiding (near) breakdown situations are also readily applicable to the biJD.

In addition, the biJD offers a much simpler solution to the problem without the need to restart the iteration. If a breakdown is detected at step 5 of the algorithm, we can insert a random vector instead of $\delta_r$ or $\delta_l$. It is more reasonable to change only one of the vectors, e.g., the left one if we are interested in the right eigenpair. Also, instead of a completely random vector, a small random perturbation of $\delta_l$ is usually enough to overcome the breakdown but still retains the basic direction of $\delta_l$.

Breakdown is also possible during the BCG iteration, but it is handled easily by early termination of BCG, which does not need to run to convergence. Specifically, there are two possible BCG breakdowns; first when the left and right BCG residual are orthogonal, and second when the LU decomposition cannot be carried out. If the first breakdown occurs in the first step of BCG, the original biJD residuals are orthogonal and the situation is treated as a biJD breakdown. If the LU breakdown occurs in the first step of BCG, we simply add the residuals in the bases and resume the biJD. If any of the two breakdowns occurs during the $i$-th BCG iteration, we terminate the BCG and return to biJD the approximate solutions from iteration $i-1$. These are not orthogonal, because the $i$-th iteration is the first time that breakdown occurs, and thus the biJD algorithm can resume.

## 3. The biJD correction equation.

There is a multitude of choices for projectors in the correction equation of JD. A general framework that describes the use and convergence properties of arbitrary projectors for the JD has been given in [29, 12], and some recent developments on preconditioning can be found in [31, 14, 15].

Despite the variety of possible correction equations, the choices for the biJD are limited because the operators of the left and right correction equations have to be adjoint to each other for the BCG to apply. We show next that from the two natural choices, the orthogonal projector $(I - xx^*)$ and the spectral one $(I - xy^*)$, only the spectral projector solves a meaningful correction equation for the left eigenvector, and thus has better convergence properties. In addition, the biorthogonal bases maintained by biJD provide an elegant framework for using the spectral projector.

### 3.1. Forming the appropriate equation.

Let $x$ be an approximation to an eigenvector of $A$, say $\tilde{x}$ with eigenvalue $\tilde{\lambda}$. We are interested in solving for the correction $\delta$ that satisfies $\tilde{x} = x + \delta$. Because of the scale invariance of $\tilde{x}$, we can look for $\delta$ in a space orthogonal to $x$ (original JD) or to another vector $p_2^*\delta = 0$ [29]. Assuming that $p_2^* x \neq 0$, we consider the projector $(I - p_1 p_2^*)$, with $p_2^* p_1 = 1$, that can represent both operators in equations (1.1) and (1.2):

$$B = (I - p_1 p_2^*)(A - \tilde{\lambda} I)(I - p_1 p_2^*). \tag{3.1}$$

Starting from the eigenvalue equation for the required eigenpair, and following the same algebraic manipulations as in [30], we obtain the correction equation:

$$B\delta = -(A - \tilde{\lambda} I)x - p_1 p_2^*(A - \tilde{\lambda} I)\delta. \tag{3.2}$$

Because $B\delta$ is orthogonal to $p_2$, the same applies for the right hand side which yields the condition: $\tilde{\lambda} = \rho + \epsilon$, with

$$\rho = \frac{p_2^* A x}{p_2^* x} \text{ and } \epsilon = \frac{p_2^*(A - \tilde{\lambda} I)\delta}{p_2^* x}. \tag{3.3}$$

Substituting $\rho$ and $\epsilon$ into equation (3.2) we obtain:

$$B\delta = (\rho x - Ax) + \epsilon(x - p_1(p_2^* x)). \tag{3.4}$$

To be able to form and solve this correction equation, the right hand side should not include any unknowns. Because $\epsilon$ is not known, this term has to vanish which is possible in general only if $p_1$ and $x$ are colinear. Let $p_1 = x/\|x\|$.

In the biJD, the left equation solved by BCG should involve the adjoint operator $B^* = (I - p_2 x^*)(A - \tilde{\lambda})^*(I - p_2 x^*)$. We want to identify a $p_2$ so that we can form an appropriate right hand side for the correction equation for an approximation $y$ to the left eigenvector $\tilde{y}$ of $A$. According to the above analysis, $p_2$ must be colinear with $y$. Thus, the projector must be of the form $(I - xy^*)$. If the orthogonal projector $I - xx^*$ is chosen instead, the right correction equation has a proper right hand side, but the same does not hold for the left one.

The above does not incapacitate a biJD method that uses BCG on this inappropriate left correction equation. It only implies that the convergence of the left eigenvector will not be as fast, which may not be relevant if we are only interested in the right eigenvector. However, as we show next, the asymptotic convergence of the biJD with equation (1.1) is inferior to the use of equation (1.2).

**3.2. Asymptotic convergence.** When we apply two coupled inverse iterations for finding both left and right eigenpairs using the generalized Rayleigh quotient:

$$(A - \lambda_s I)x_{s+1} = x_s, \qquad (A - \lambda_s I)^* y_{s+1} = y_s,$$
$$\text{with } \lambda_s = y_s^* A x_s / y_s^* x_s,$$

convergence is known to be ultimately cubic [37]. A large condition number of the sought eigenvalue, $\kappa(\tilde{\lambda}) = \|\tilde{y}\| \|\tilde{x}\| / \tilde{y}^* \tilde{x}$, would only delay the cubic convergence phase.

The situation is very similar in the coupled solution of the left and right correction equations with BCG. In [29] (Theorem 3.4, Remark 3.5), Sleijpen et al. prove that if $y$ converges to the left eigenvector, a stationary iteration that corrects an approximation $x$ to the right eigenvector with the solution of equation (1.2) has locally cubic convergence to the right eigenpair. The JD accelerates the stationary method by performing Galerkin over the basis of all $x$ iterates, providing also global convergence.

Exactly the same result holds for the biJD method, because it only applies a different acceleration method to the correction equation (1.2). The difference is that biJD treats left and right eigenvectors symmetrically, with $y$ converging to the left eigenvector with speed similar to that of $x$, thus guaranteeing the local cubic convergence. The same is not true in general for the JD method, since the right space may never contain sufficient components of the left eigenvector.

As with the classical JD, the biJD converges quadratically [29, 38] if equation (1.1) is solved accurately for the right eigenpair, regardless of any left equation used. Even if an appropriate equation were solved for the left pair, since equation (1.1) does not use any $y$ information, convergence to the right pair would still be quadratic. Therefore, we expect faster biJD convergence with the correction equation (1.2); even when the equations are not solved accurately.

Note that the conditioning of the operator $(I - xy^*)(A - \lambda I)(I - xy^*)$ depends on $\|y\|$, or equivalently on the angle between $x$ and $y$. At the limit, the operator is equivalent to a deflated matrix, and thus $\|A - \tilde{\lambda}\ \tilde{x}\tilde{y}^*/\tilde{x}^*\tilde{y}\| \leq \|A\| + |\tilde{\lambda}|\ \kappa(\tilde{\lambda})$. On the other hand, the operator $(I - xx^*)(A - \lambda I)(I - xx^*)$ does not increase the norm of the matrix. This suggests that, for stability reasons, the biJD method could switch to the orthogonally projected equation if an ill conditioned Ritz pair is detected. However, the correction equations are never solved accurately, and moreover in all our experiments we have observed that the ill conditioning stems only from an increase in the largest singular value while the rest are not affected. Also, at the limit, the spectral projector preserves both left and right eigenvectors, while the orthogonal projector preserves only the left ones.

**3.3. Using preconditioning in biJD.** Performing preconditioning for the correction equations of the JD and especially of the biJD is involved, because we must approximate the inverse of a projected matrix. In the common case of $M \approx A - \sigma I$, the appropriate application of this preconditioner would be to invert the operator $(I - xy^*)M(I - xy^*)$, which is not practical, and often, not even feasible.

For the JD with correction equation (1.1), Sleijpen et al. [30] describe a way to apply such a preconditioner implicitly, by solving systems with $M^{-1}$ and by applying a few additional orthogonalizations. In [29] they extended this scheme to arbitrary projections for the correction equation. Considering the correction equation (1.2) for the right eigenpair and a preconditioner $M \approx A - \sigma I$, Theorem 7.3 in [29] states that the appropriate preconditioned correction equation can be written as:

$$\left(I - \frac{M^{-1}xy^*}{y^*M^{-1}x}\right) M^{-1}(A - \sigma I) \left(I - \frac{M^{-1}xy^*}{y^*M^{-1}x}\right) \delta_r = -\left(I - \frac{M^{-1}xy^*}{y^*M^{-1}x}\right) M^{-1}r_r. \tag{3.5}$$

Let us consider the correction equation for the left eigenpair, with operator the adjoint of (1.2), and $M^* \approx (A - \lambda I)^*$. If we apply the above theorem, we obtain the appropriate preconditioned correction equation for the left eigenpair.

$$\left(I - \frac{M^{-*}yx^*}{x^*M^{-*}y}\right) M^{-*}(A - \sigma I)^* \left(I - \frac{M^{-*}yx^*}{x^*M^{-*}y}\right) \delta_l = -\left(I - \frac{M^{-*}yx^*}{x^*M^{-*}y}\right) M^{-*}r_l. \tag{3.6}$$

However, BCG cannot solve these systems simultaneously, because the operators are not exactly adjoint. We transform the system (3.6) by multiplying the equation from the left with $M^*$, and letting $\delta_l = M^{-*}t$. This yields an equivalent left correction equation, with an operator adjoint to the one in (3.5):

$$\left(I - \frac{yx^*M^{-*}}{x^*M^{-*}y}\right) (A - \sigma I)^* M^{-*} \left(I - \frac{yx^*M^{-*}}{x^*M^{-*}y}\right) t = -\left(I - \frac{yx^*M^{-*}}{x^*M^{-*}y}\right) r_l. \tag{3.7}$$

Because $x^*M^{-*}t = x^*M^{-*}M^*\delta_l = x^*\delta_l$, the orthogonality condition $\delta_l \perp x$ in equation (3.6) is equivalent to the orthogonality condition $t \perp M^{-1}x$ in (3.7). Note also that $P_M = (I - yx^*M^{-*}/x^*M^{-*}y)$ is a projector with $P_M y = 0$ and $P_M t = t$. Thus, equation (3.7) is a correction equation for the left eigenpair.

**4. Efficient restarting for biJD.** The idea of thick restarting is based on the observation that as Krylov methods approximate extreme eigenvectors, these vectors become gradually deflated from the iteration and the method converges faster. The goal of thick restarting is to retain those Ritz vectors that the method tends to approximate better, so that they can be improved and thus cause the superlinear convergence (for linear systems see [7, 17, 2]). The Ritz vectors with Ritz values closest to the required eigenvalue are thus a natural choice. The dynamic thick restarting scheme retains also Ritz vectors with Ritz values in the extreme part of the spectrum, since Krylov methods approximate these vectors better [35]. In the symmetric case, the results of the dynamic scheme have been impressive [35, 33]. In the nonsymmetric case, it still performs well, but the improvements are not as dramatic and more matrix dependent. However, thick and especially dynamic thick restarting increase the iteration costs because they retain a large number of vectors at restart.

A different class of restarting strategies is based on the observation that all Krylov methods enforce some kind of orthogonality in order to guarantee new directions in the basis [11, 10]. With restarting, some directions are discarded, and the loss of full orthogonality causes the convergence to deteriorate. This behavior is common not only in explicitly restarted methods such as Arnoldi and GMRES, but also in methods based on short recurrences, such as CG. The much researched loss of orthogonality in CG is known to cause slower convergence. Restarting strategies in this class attempt to identify and retain those directions that the algorithm tends to repeat. A typical and effective example is the truncation strategy of de Sturler [10].

Interestingly, the two restarting classes often overlap. In the symmetric case, selective orthogonalization against converged eigenvectors can be viewed both as a deflation and as an orthogonality based method. In the nonsymmetric case, eigenvector deflation can also be viewed as a special case of orthogonality conditions, but the problem is more complicated and other directions become important.

The above suggest that maintaining orthogonality against all visited directions is a critical issue in restarted iterative methods. The three term recurrence of the symmetric Lanczos achieves full orthogonality implicitly, so it is natural to seek ways to use this recurrence to restart efficiently the symmetric JD method.

**4.1. Restarting idea for symmetric JD.** Even though explicit full orthogonalization is avoided in the Lanczos algorithm through the three term recurrence, the basis vectors still need to be stored for computing the Ritz vector. However, if the exact eigenvalue is known, the eigenvector can be obtained by the Conjugate Gradient (CG) method storing only three vectors [36, 16]. If the eigenvalue is not known but converges rapidly, methods based on CG can still be used [36].

A more useful variant of this idea was proposed in [20] and extended and analyzed in [34]. It is based on the observation that, in the absence of preconditioning, the space built by CG for solving the correction equation differs from the Krylov space of the Lanczos method (JD with no correction step) only in the starting vector. In addition, if the Ritz value at step $k$ were known, the two methods would yield exactly the same vector at the $k$-th step. Note, that the CG minimizes the $A$-norm of the error on a three-vector space, that is close to the space spanned by $\{x^{(k-1)}, x^{(k)}, r\}$, where $x^{(k-1)}, x^{(k)}$ are successive Ritz vectors from JD iterations $k-1$ and $k$ respectively, and $r$ is the residual of $x^{(k)}$.

We have argued that, if the JD method is restarted at the $k$-th iteration, it is beneficial to keep the Ritz vector from the previous iteration ($x^{(k-1)}$) along with the current one. In fact, if these three-vector spaces from CG and JD were identical, there would be no information loss by this restarted JD variant. In general, the two spaces are not the same but close if the Ritz value does not vary significantly between steps.

This technique works extremely well for extreme eigenpairs, and still performs well for interior eigenpairs because it retains some orthogonality memory. Combining this scheme with thick restarting provides in addition a deflation-like character, and it is the only technique that has managed to improve on the dynamic thick restarting.

**4.2. Extending to non symmetric matrices.** Extending the above restarting technique to the nonsymmetric JD is not possible because orthogonality must be maintained explicitly. However, if we trade the more stable orthogonality for biorthogonality, the nonsymmetric Lanczos fits the description. This is the second motivation, besides the faster outer convergence, for proposing the biJD algorithm.

The changes in the Algorithm 2.1 are analogous to the symmetric case. We denote new steps by decimal numbers to show between which biJD steps they are inserted.

ALGORITHM 4.1. *Additions to biJD for new restarting*

*1.1* $x_{prev} = x,\ y_{prev} = y$

*9.1* *Biorthogonalize* $(x_{prev}, y_{prev})$ *against* $(V, W)$

*9.2* *set* $V = [V, x_{prev}]$, $W = [W, y_{prev}]$, *and* $H_{k+1,k+1} = y^*_{prev} A x_{prev}$.

Note that the restarting applies symmetrically to both left and right spaces. For clarity, the algorithm above presents the restarting scheme in terms of the long vectors $x_{prev}$ and $y_{prev}$. In practice, all above operations can be performed without extra matrix vector multiplications or long vector biorthogonalizations. Because $x = Vg$ and $x_{prev} = V[c_1, \ldots, c_{m-1} 0]^T$, for some coefficient vector $c \in \mathcal{C}^{m-1}$, biorthogonalizations can be performed in the coefficient space. In addition, the updates of matrices $K \leftarrow K g_i$ and $L \leftarrow L f_i$, for $i = 1, \ldots, k+1$ during restarting can be computed using the coefficient vector $c$. Finally, in the above algorithm, the new restarting scheme is coupled with thick restarting by adding in the restarted basis both the previous Ritz vector and the $k$ current ones. The rationale is analogous to the symmetric case.

**4.2.1. Extending the theory.** In this section we extend to the nonsymmetric case the theory developed in [34]. The goal is to explain why restarting based on the three term recurrence yields future Ritz vectors that are close to the Ritz vectors we would have obtained without restarting.

To facilitate presentation clarity, we use a single subscript that denotes the iteration number for any variable, e.g., $x_i$ is the Ritz vector at the $i$-th iteration. We assume that the matrix $A$ is diagonalizable, with no multiple eigenvalues.

LEMMA 4.1. *Let* $x$, $y$ *vectors of* $\mathcal{C}^N$, *such that* $y^*x = 1$, *and* $\lambda = y^*Ax$. *Let* $\pi = (I - xy^*)$ *the oblique projector onto* $y^\perp$, *and the residual of* $x$: $r = (A - \lambda I)x = \pi r$. *Then, for every* $k > 1$:

$$\mathit{span}(\{x, Ax, \ldots, A^k x\}) = \mathit{span}(\{x, r, (\pi A \pi) r, \ldots, (\pi A \pi)^{k-1} r\}).$$

*Proof.* Denote by $\mathcal{K}_k$ and $\mathcal{L}_k$ the spaces of the left and right hand side respectively. Obviously, for $k = 1$, $\mathcal{K}_1 = \mathcal{L}_1$. We assume that $\mathcal{K}_i = \mathcal{L}_i$ for all $i < k$. Let $q \in \mathcal{L}_k$. There is $u \in \mathcal{L}_{k-1} = \mathcal{K}_{k-1}$ and $\alpha \in \mathcal{C}$ such that:

$$q = u + \alpha(\pi A \pi)^{k-1} r = u + \pi A \pi\ z,$$

where $z = \alpha(\pi A \pi)^{k-2} r \in \mathcal{L}_{k-1} = \mathcal{K}_{k-1}$. Since $\pi = I - xy^*$ and $Az \in \mathcal{K}_k$, we have:

$$\begin{aligned} q &= u + (I - xy^*)A(z - (y^*z)x) \\ &= u + Az - (y^*z)Ax - (y^*Az)x + (y^*Ax)(y^*z)x \quad \in \mathcal{K}_k. \end{aligned}$$

Thus, $\mathcal{L}_k \subseteq \mathcal{K}_k$. If $\mathcal{L}_k$ is of full dimension, its dimension is $k+1$, the same as $\mathcal{K}_k$, and thus the two spaces must be equal. If $\mathcal{L}_k$ is not of full dimension, then it forms a smaller invariant subspace of dimension $i < k+1$, which is also included in $\mathcal{K}_k$. Then, from the inductive hypothesis, $\mathcal{L}_k = \mathcal{L}_{i-1} = \mathcal{K}_{i-1} = \mathcal{K}_k$. ☐

The lemma says that the right (left) Krylov space built by Lanczos in $k$ steps is the same as the right (left) Krylov space that BCG builds in $k$ steps when solving the correction equation (1.2), appended with the initial vector $x$ ($y$). The use of the spectral projector is important for the left equation. If $(I - xx^*)$ were used instead, it would introduce a multiple of $x$ term, which does not belong in the Krylov space.

THEOREM 4.2. *Let* $x_0, y_0 \in \mathcal{C}^N$, *with* $\|x_0\| = 1, y_0^* x_0 = 1, \lambda_0 = y_0^* A x_0$, *and* $\sigma \in \mathcal{C}$.

*Let $(x_k, y_k, \lambda_k)$ be the right and left Ritz vectors and their generalized Rayleigh quotient after $k$ steps of the biJD method with no correction equation (Lanczos), with $(x_0, y_0)$ as right and left starting vectors.*

*Let $z_k = x_0 + \delta_r$ and $w_k = y_0 + \delta_l$ be the approximate right and left eigenvectors, where $\delta_r$ and $\delta_l$ are the right and left corrections obtained by applying $k$ steps of the BCG method to the equation (1.2) with shift $\sigma$. Then:*

$$z_k = x_k \text{ and } w_k = y_k \Leftrightarrow \sigma = \lambda_k.$$

*Proof.* Because there is no correction equation being solved, the biJD builds a right and left Krylov spaces:

$$span(\{x_0, Ax_0, \ldots, A^k x_0\}) \text{ and } span(\{y_0, A^* y_0, \ldots, A^{*k} y_0\}).$$

Let $\pi = I - xy^*$, and denote $r_r = \alpha(A - \lambda_0)x_0$, and $r_l = \alpha(A - \lambda_0)^* y_0$, where $\alpha \in \mathcal{C}$ is chosen such that $r_l^* r_r = 1$. Note that $\pi r_r = r_r$ and $\pi^* r_l = r_l$. The BCG method on equation (1.2), starting with zero right and left initial guesses, builds the spaces:

$$span(\{r_r, (\pi A \pi) r_r, \ldots, (\pi A \pi)^{k-1} r_r\}) \text{ and } span(\{r_l, (\pi A \pi)^* r_l, \ldots, (\pi A \pi)^{*(k-1)} r_l\}).$$

By construction, the Lanczos biorthogonal bases for the biJD spaces have $(x_0, r_r)$ and $(y_0, r_l)$ as their first two vectors. Therefore, if we consider bases $\{x_0, X\}$ and $\{y_0, Y\}$ for these two subspaces, with $Y^* X = I$, then $X$ and $Y$ are also bases of the spaces generated by the BCG. In the following, we focus only on the right Ritz pair, because the arguments for the left one are identical.

With the above bases, and normalizing the Ritz vector $x_k$ so that its coefficient of $x_0$ is one, the Petrov-Galerkin projection at the $k$-th step of the biJD solves the following problem (note the matrix is tridiagonal):

$$\begin{bmatrix} \lambda_0 & y_0^* AX \\ Y^* A x_0 & Y^* AX \end{bmatrix} \begin{pmatrix} 1 \\ c_k \end{pmatrix} = \lambda_k \begin{pmatrix} 1 \\ c_k \end{pmatrix}, \tag{4.1}$$

or equivalently, the following system, which has $k+1$ Ritz pairs as solutions. We fix the equations for a specific $(\lambda_k, c_k)$, $c_k \in \mathcal{C}^k$, so that $x_k = x_0 + X c_k$:

$$\lambda_0 + y_0^* AX c_k = \lambda_k \tag{4.2}$$

$$Y^* A x_0 + Y^* AX c_k = \lambda_k c_k. \tag{4.3}$$

Consider the bases $X, Y$ for the Petrov-Galerkin condition of the BCG method. The BCG computes a correction to $x_0$ and sets $z_k = x_0 + X c_k'$. Because $\pi X = X$, and $Y^* \pi^* = Y^*$, the projected problem solved is:

$$Y^*(A - \sigma I) X c_k' = Y^*(\lambda_0 I - A) x_0 = -Y^* A x_0. \tag{4.4}$$

From equations (4.3) and (4.4) we obtain:

$$(Y^* AX - \lambda_k) c_k = (Y^* AX - \sigma) c_k'. \tag{4.5}$$

If $c_k = c_k'$ (and thus $x_k = z_k$), then obviously $\sigma = \lambda_k$. Conversely, if $\sigma = \lambda_k$, we have

$$(Y^* AX - \lambda_k)(c_k - c_k') = 0, \tag{4.6}$$

which implies that $c_k = c'_k$, and thus $x_k = z_k$. Note that $(\lambda_k, c_k - c'_k)$ could not be an eigenpair of $Y^*AX$, because the $k \times k$ matrix in equation (4.1) is tridiagonal, irreducible (since $k$ steps of biJD/BCG can be carried out), and it already has $\lambda_k$ as an eigenvalue.

The proof for the left eigenvectors is identical to the above. □

The following diagram depicts the iterates of Lanczos (biJD with no correction equation) and of BCG with the Ritz value $\lambda_k$ as shift. If started with the same vector $x_0$, they end in the same Ritz vector $x_k$ at the $k-$th step. Intermediate vectors differ in general. In this case, we can recover the information lost in restarting Lanczos.

$$\begin{array}{ccccccccc} & \nearrow & x_1 & x_2 & x_3 & \dots & x_{k-2} & x_{k-1} & \searrow & \\ x_0 & & & & & & & & & x_k \\ & \searrow & z_1 & z_2 & z_3 & \dots & z_{k-2} & z_{k-1} & \nearrow & \end{array}$$

However, $\lambda_k$ is usually computed during the Lanczos (biJD) procedure. What is pertinent to our restarting scheme, is whether the three term BCG recurrence still produces an accurate approximation to the Ritz vector, if an inexact eigenvalue shift is used in equation (1.2). The following lemma quantifies the distance of the vectors $x_k$ and $z_k$ when $\sigma \neq \lambda_k$.

LEMMA 4.3. *Let the assumptions and notations of Theorem 4.2 hold, and denote by $s$ the smallest singular value of the matrix $Y^*AX - \lambda_k I$. Then:*

$$\|z_k - x_k\| \leq |\sigma - \lambda_k| \frac{\|X\| \|c'_k\|}{s}.$$

*Proof.* Let $S = Y^*AX$ in equation (4.5), and since $\lambda_k$ is not an eigenvalue of $S$,

$$c_k = (S - \lambda_k I)^{-1}(S - \sigma I)c'_k.$$

From the definitions of $x_k, z_k$ and from the above equation we have:

$$\begin{aligned} \|z_k - x_k\| = \|X(c'_k - c_k)\| = \|X(I - (S - \lambda_k I)^{-1}(S - \sigma I))c'_k\| = \\ \|(\sigma - \lambda_k)X(S - \lambda_k I)^{-1}c'_k\| \leq |\sigma - \lambda_k| \frac{\|X\| \|c'_k\|}{s}. \end{aligned}$$

□

COROLLARY 4.4. *If in addition to the assumptions of lemma 4.3, $\|z_k\|$ is larger than the correction term, i.e., $\|z_k\| > \|Xc'_k\|$, then we have a relative bound involving the condition number of the basis $X$:*

$$\frac{\|z_k - x_k\|}{\|z_k\|} \leq |\sigma - \lambda_k| \frac{\kappa(X)}{s}.$$

*Proof.* We have $\|z_k\| \geq \|Xc'_k\| = \sqrt{c'^*_k X^* X c_k} \geq \sigma_{min}(X)\|c'_k\|$. Dividing both sides of the bound in lemma 4.3 yields the result. □

These bounds imply that when the Ritz value is almost constant, which usually occurs near convergence or when convergence is slow, the BCG computes a close approximation to the Ritz vector of the biJD. In the context of restarting, assume that we need to compute the $(\lambda_{k+1}, x_{k+1})$ Ritz pair, and that the biJD (no correction equation) is restarted after $k-1$ steps, retaining only the Ritz pair $(\lambda_{k-1}, x_{k-1})$. After restarting, the biJD generates the Ritz pair $(\lambda_k, x_k)$, but after a second iteration the new Ritz pair differs from $(\lambda_{k+1}, x_{k+1})$. Consider a hypothetical BCG recurrence that uses the unknown $\lambda_{k+1}$ to produce the wanted Ritz pair in $k+1$ steps. If we apply corollary 4.4 on the vectors $z_i$ of the BCG, but consider instead $x_{k-1}$ and $x_k$ as the end points, we get two inequalities:

$$\begin{aligned} \|z_{k-1} - x_{k-1}\|/\|z_{k-1}\| &\leq \mathcal{O}(|\lambda_{k+1} - \lambda_{k-1}|) \\ \|z_k - x_k\|/\|z_k\| &\leq \mathcal{O}(|\lambda_{k+1} - \lambda_k|) \end{aligned}$$

When the Ritz value is almost constant between steps, the Ritz vectors $x_{k-1}$ and $x_k$ approximate the BCG iterates for the still uncomputed $k+1$ step. Because $x_{k+1}$ is a linear combination of the unknown $z_k, z_{k-1}$, a good restarting basis for the biJD is one consisting of both Ritz vectors $\{x_{k-1}, x_k\}$.

However, proximity may not be as good as in the symmetric case [34]. As expected, the bounds include both the condition number of the basis matrix $X$, and the smallest singular value of $S$, which incorporates information on the conditioning of the eigenvectors of $S$ and the distance of other eigenvalues from $\lambda_k$. In case of highly ill conditioned bases or eigenvalues, the effects of the restarting scheme seem arbitrary, although in such cases the problem should be traced rather in the near ill posedness of the eigenproblem. Finally, eigenvalue convergence in the nonsymmetric case is not monotonic and sometimes irregular, which complicates the runtime interpretation of the bounds to decide whether the restarting scheme should be applied. Nevertheless, if we know when to apply it, the new restarting scheme works very well on a variety of matrices, as shown in the experiments in the following section.

**5. Numerical experiments.** We have implemented the above algorithms in Matlab, and conducted an extensive set of tests on nonsymmetric matrices from the collection in [3]. In the experiments that we present, we look for the right eigenpair that is of interest in the application domain of the matrix. We iterate until the residual norm reduces by $10^{-8}$, and we plot residual convergence versus the number of outer iterations. Experiments are run on a SUN Ultra 2300, using Octave. In the figure notation, `JD` is the JD, `biJD(I-xy')` (or simply `biJD`) and `biJD(I-xx')` is the biJD with correction equation (1.2), and (1.1) respectively. `biJD+1` denotes the biJD whose basis is augmented by the previous Ritz vector at restart.

**5.1. biJD vs JD without restarting.** In the first set of experiments we compare the biJD and JD without restarting, and with each method applying 10 steps of BCG or GMRES respectively to its correction equation. No preconditioner or harmonic eigenpairs are used for the correction equation.

The experiments suggest three general observations that agree with the theory discussed in the paper. First, although 10 steps on the correction equation are not enough for biJD and JD to demonstrate cubic or quadratic convergence respectively, the convergence of the biJD is usually faster asymptotically. The *semiquadratic* convergence of the nonsymmetric Lanczos also contributes to this [4]. Second, the superiority of the Petrov-Galerkin over the Galerkin process is problem dependent. Third, the projection $I - xx^*$ in the biJD correction equation does not usually help convergence.

In figure 5.1, the left graph shows the convergence for the pde225 matrix. We look for the eigenvalue with the largest real part. In this case, the biJD has better global convergence than the JD, suggesting that the Petrov-Galerkin may be finding the correct components early in the iteration. Note also that there is practically no difference between the two ways of projecting the correction equation. The right graph in figure 5.1, shows the convergence for the Tolosa 340 matrix. The goal is to

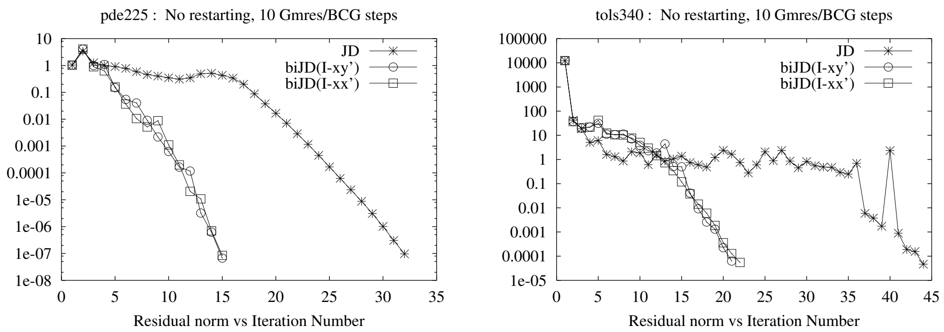



FIG. 5.1. *Convergence history for the residual norm of the JD, and variants of biJD in terms of outer iterations. There is no restarting, and in each outer iteration 10 GMRES or BCG steps are applied to the correction equation. Left graph: matrix pde225. Right graph: matrix tols340.*

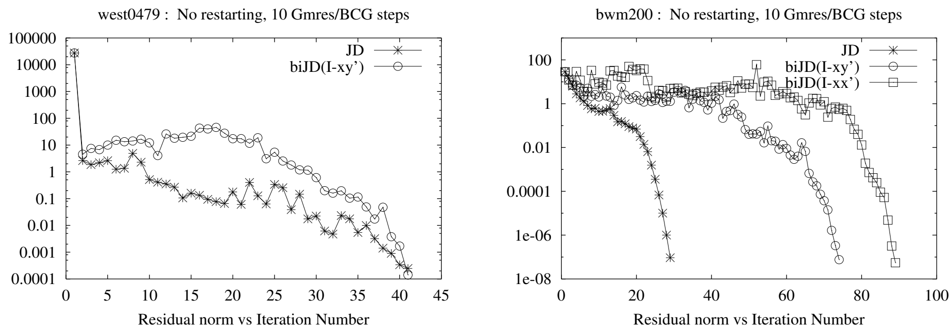



FIG. 5.2. *Convergence history for the residual norm of the JD, and variants of biJD in terms of outer iterations. There is no restarting, and in each outer iteration 10 GMRES or BCG steps are applied to the correction equation. Left graph: matrix west0479. Right graph: matrix bwm200.*

compute an eigenvalue with largest imaginary part. The observations are the same as with matrix pde225, except that the gap between JD and the biJD is even larger.

The results in figure 5.2 show that, as with Lanczos versus Arnoldi, there are also cases where JD performs better than biJD. The left graph involves matrix west0479, and we look for the interior eigenpair closest to (-17.825 - 4.6376 i). Note that although the JD curve is below the biJD, the asymptotic convergence of biJD is clearly superior. The right graph involves the matrix bwm200, and we look for the eigenvalue with the largest real part. In this case, the global convergence of the JD is particularly fast. However, the credit should not go to the correction equation, because the same equation seems to hurt the convergence of `biJD(I-xx')`.

**5.2. biJD vs JD with restarting.** In the second set of experiments, we examine the effects of restarting on JD and biJD. We allow 20 vectors for the JD basis, and 20 vectors for each of the left and right bases of biJD. We thick restart JD and biJD with five Ritz vectors, while `biJD+1` thick restarts with four Ritz vectors and the Ritz vector from the previous step. In certain cases, we switch to `biJD+1` scheme only after relatively good eigenvalue approximations have been obtained.

Our observations confirm that both JD and biJD outperform each other depending on the problem. However, while JD can only use thick restarting variants, the biJD

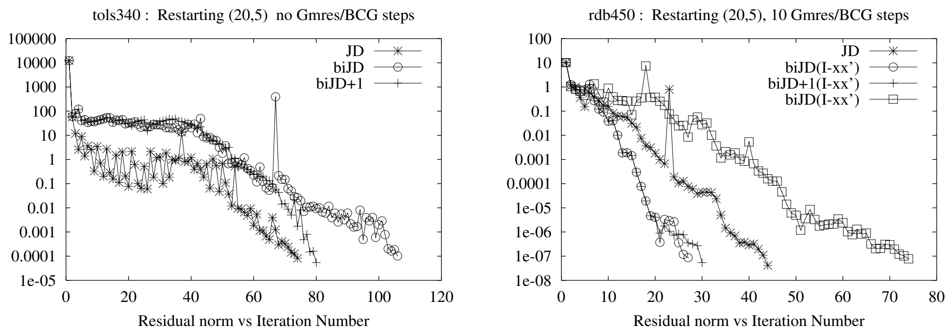



FIG. 5.3. *Convergence of the residual norm of the restarted JD, and biJD. Maximum basis size is 20, and thick restarting is 5. For* `biJD+1` *thick restarting is 4 plus the previous Ritz vector. Left graph: matrix tols340 (no correction eq.). Right graph: matrix rdb450 (with correction eq.).*

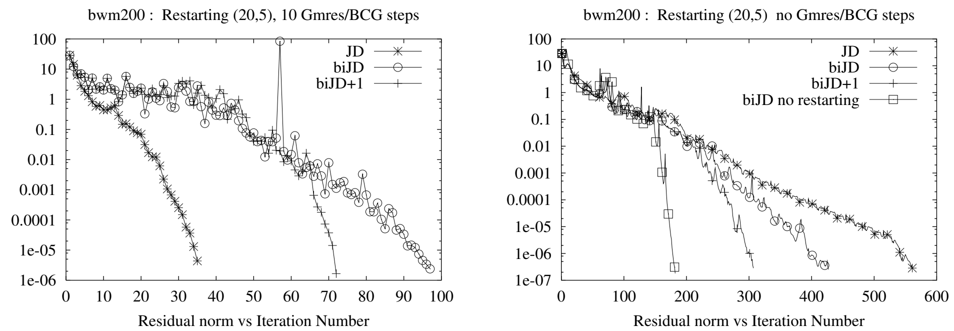
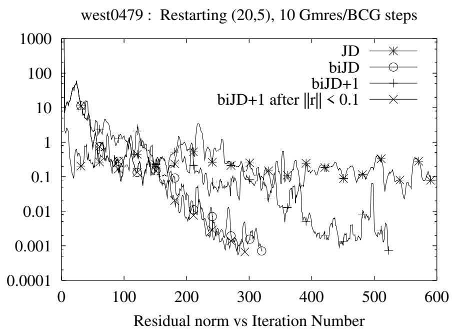



FIG. 5.4. *Convergence of the residual norm of the restarted JD, and biJD. Maximum basis size is 20, and thick restarting is 5. For* `biJD+1` *thick restarting is 4 plus the previous Ritz vector. Matrix: bwm200. Left graph: with correction equation. Right graph: no correction equation.*

can use the combined restarting scheme which can result in a substantial reduction of the number of iterations.

In figure 5.3, the left graph involves the Tolosa matrix again, but in this case JD is faster than biJD. The `biJD+1` matches the performance of JD, assuming a fast, superlinear convergence, which for smaller thresholds would supersede JD. In the right graph, we look for the rightmost eigenvalue of the matrix rdb450. In this case, the JD does not perform as well as biJD. The reason for the minor differences between biJD and `biJD+1` is that the algorithm converges before a second restart takes place. In addition, we include the convergence of the `biJD(I-xx')` with thick restarting. An additional intuitive reason for its bad performance is that the projector should provide a solution with a direction not overlapping with the basis. The incorporation of this new direction is determined in biJD by the biorthogonality condition, which is oblique and not orthogonal. The correction equation must respect the same condition.

In figure 5.4, we examine restarted methods for the bwm200 matrix, both with the correction equation (left graph) and without it (right graph). Once again, the situation is reversed between the two methods. When solving the correction equation, the JD is far better than biJD (see also the nonrestarted JD in figure 5.2). `biJD+1` improves convergence but it is still far from the JD. On the other hand, when no correction equation is solved (right graph), JD converges the slowest, while using `biJD+1` comes



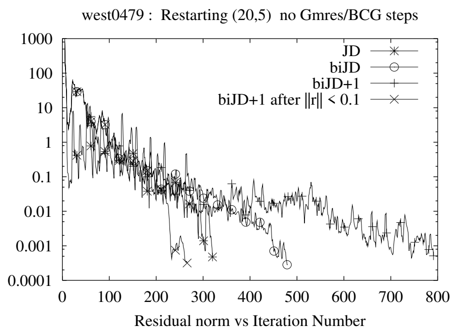




FIG. 5.5. *Convergence of the residual norm of the restarted JD, and biJD. Maximum basis size is 20, and thick restarting is 5. A variant of* `biJD+1` *retains the previous Ritz vector if* $\|r_r\| < 0.1$. *Matrix: west0479. Left graph: with correction equation. Right graph: no correction equation.*

surprising close to the nonrestarted method.

Finally, in figure 5.5 we examine the interior problem from the matrix west0479. Note also from figure 5.2, that a subspace of 40 would be enough to converge rapidly to the solution. By reducing the number of bases vectors and introducing restarting, the iteration count increases dramatically, even with 10 steps on the correction equation. In this case, JD does not converge for at least 1300 steps, while biJD converges in 320 steps. The `biJD+1` scheme, if applied from the beginning, converges in more iterations. However, if we apply the combination scheme only after the Ritz values have relatively stabilized (residual norm less than 0.1), `biJD+1` improves slightly on the biJD convergence. When no correction equation is solved (right graph), JD outperforms biJD. `biJD+1` applied during all restarts is substantially worse, possibly because in early iterations the restarting was locking onto a wrong eigenpair, discarding useful information. However, when applied dynamically only after the residual norm is less than 0.1, `biJD+1` can improve significantly the performance of the method.

**6. Conclusions.** The proposed biorthogonal Jacobi-Davidson method incorporates many of the advantages of the nonsymmetric Lanczos and the Jacobi-Davidson methods. We have given an elegant formulation of the algorith that allows for a host of features and functionalities, including preconditioning, simple resolution of breakdowns, use of harmonic Ritz pairs, thick restarting, and use of left eigenvectors for both eigenvalue approximation and convergence estimation. We have also shown that on today's computers with multiple memory hierarchies, the multiplication of the adjoint of the matrix with a vector can be performed with only one memory access, and thus with minimal additional cost.

The two distinct characteristics of the biJD method that make it competitive against the JD method are a better correction equation and an efficient restarting strategy. Our experiments agree with our theoretical discussion that the more accurately the JD and biJD correction equations are solved, the faster the biJD converges. Restarting with a combination of Ritz vectors from the current and previous step is possible with biJD, but not with the JD. This restarting technique that can offer huge convergence improvements. Although, a similar restarting scheme could possibly be developed for CGS-like methods, the additional features and the faster correction equation make biJD a more promising choice.

As confirmed by our experiments, the method often outperforms the JD, with and without restarting or correction equation. However, as with Lanczos and Arnoldi, biJD and JD outperform each other in different problems. Moreover, harvesting the huge potential of the restarting scheme is not as black box as in the symmetric case. Overall, however, the biJD is a highly competitive algorithm for a difficult problem.

**Acknowledgement.** The author would like to thank Eric de Sturler for many fruitful discussions.

REFERENCES

[1] Owe Axelsson. *Iterative Solution Methods*. Cambridge University Press, New York, 1994.

[2] J. Baglama, D. Calvetti, G. H. Golub, and L. Reichel. Adaptively preconditioned GMRES algorithms. *SIAM J. Sci. Comput.*, 20:243–269, 1999.

[3] Z. Bai, D. Day, J. Demmel, and J. Dongarra. A test matrix collection for non-Hermitian eigenvalue problems. Technical report, Department of Mathematics, University of Kentucky, September 30, 1996.

[4] Z. Bai, D. Day, and Q. Ye. ABLE: An adaptive block lanczos method for non-Hermitian eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 20(4):1060–1082, 1999.

[5] R. Barrett, M. Berry, T.F. 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, 1995.

[6] Christopher Beattie. Harmonic Ritz and Lehmann bounds. *Electr. Trans. Numer. Anal.*, 7:18–39, 1998.

[7] A. Chapman and Y. Saad. Deflated and augmented Krylov subspace techniques. *Numerical Linear Algebra with Applications*, 4:43–66, 1997.

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

[9] Ernest 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.

[10] E. de Sturler. Truncation strategies for optimal Krylov subspace methods. *SIAM J. Matrix Anal. Appl.*, to appear.

[11] Michael Eiermann and Oliver G. Ernst. Geometric aspects in the theory of krylov subspace methods. In *Proceedings of Copper Mountain Conference on Iterative Methods*, 2000.

[12] D. R. Fokkema, G. L. G. Sleijpen, and H. A. van der Vorst. Jacobi-Davidson style QR and QZ algorithms for the partial reduction of matrix pencils. *SIAM J. Sci. Comput.*, 20(1), 1998.

[13] R. W. Freund. Quasi-kernel polynomials and their use in non-hermitian matrix iterations. *Journal of Computational and Applied Mathematics*, 34:135–158, 1992.

[14] M. Genseberger and G. L. G. Sleijpen. Alternative correction equations in the Jacobi-Davidson method. Technical Report 1073, Department of Mathematics, University of Utrecht, 1999.

[15] M.E. Hochstenbach and G. L. G. Sleijpen. Two-sided and alternating Jacobi-Davidson. In *III International Workshop on Accurate Solution of Eigenvalue Problems, Hagen, Germany*, 2000.

[16] A. V. Knyazev. A preconditioned Conjugate Gradient method for eigenvalue problems and its implementation in a subspace. *International Series of Numerical Mathematics*, 96:143–154, 1991.

[17] R. B. Morgan. A restarted GMRES method augmented with eigenvectors. *SIAM J. Matrix Anal. Appl.*, 16, 1995.

[18] R. B. Morgan. On restarting the Arnoldi method for large nonsymmetric eigenvalue problems. *Math. Comput.*, 65:1213–1230, 1996.

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

[20] C. W. Murray, S. C. Racine, and E. R. Davidson. Improved algorithms for the lowest eigenvalues and associated eigenvectors of large matrices. *J. Comput. Phys.*, 103(2):382–389, 1992.

[21] C. C. Paige, B. N. Parlett, and Van der Vorst. Approximate solutions and eigenvalue bounds from Krylov spaces. *Num. Lin. Alg. Appl.*, 2:115–133, 1995.

[22] B. N. Parlett, D. R. Taylor, and Z. A. Liu. A look-ahead lanczos algorithm for unsymmetric matrices. *Math. Comput.*, 33:680–687, 1985.

[23] Beresford N. Parlett. *The Symmetric Eigenvalue Problem*. SIAM, Philadelphia, PA, 1998.

[24] Yousef Saad. SPARSKIT: A basic toolkit for sparse matrix computations. Technical Report 90-

20, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffet Field, CA, 1990. Software currently available at <ftp://ftp.cs.umn.edu/dept/sparse/>.

[25] Yousef Saad. *Numerical methods for large eigenvalue problems.* Manchester University Press, 1992.

[26] Yousef Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Comput.*, 14(2):461–469, March 1993.

[27] Yousef Saad. *Iterative methods for sparse linear systems.* PWS Publishing Company, 1996.

[28] G. De Samblanx and A. Bultheel. Nested Lanczos: implicitly restarting an unsymmetric Lanczos algorithm. *Numerical Algorithms*, 18(1):31–50, 1998.

[29] G. L. G. Sleijpen, A. G. L. Booten, D. R. Fokkema, and H. A. van der Vorst. Jacobi-davidson type methods for generalized eigenproblems and polynomial eigenproblems. *BIT*, 36(3):595–633, 1996.

[30] 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(2):401–425, 1996.

[31] G. L. G. Sleijpen and Fred W. Wubs. Effective preconditioning techniques for eigenvalue problems. Technical Report 1117, Department of Mathematics, University of Utrecht, 1999.

[32] D. C. Sorensen. Implicit application of polynomial filters in a K-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13(1):357–385, 1992.

[33] A. Stathopoulos. Some insights on restarting symmetric eigenvalue methods with Ritz and harmonic Ritz vectors. In D. R. Kincaid and Anne C. Elster, editors, *Iterative Methods in Scientific Computation IV*, pages 297–311. IMACS, NJ, 1999.

[34] A. Stathopoulos and Y. Saad. Restarting techniques for (Jacobi-)Davidson symmetric eigenvalue methods. *Electr. Trans. Numer. Alg.*, 7:163–181, 1998.

[35] A. Stathopoulos, Y. Saad, and K. Wu. Dynamic thick restarting of the Davidson, and the implicitly restarted Arnoldi methods. *SIAM J. Sci. Comput.*, 19(1):227–245, 1998.

[36] J.H. Van Lenthe and Peter Pulay. A space-saving modification of Davidson's eigenvector algorithm. *J. Comput. Chem.*, 11(10):1164–1168, 1990.

[37] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Claredon Press, Oxford, England, 1965.

[38] K. Wu, Y. Saad, and A. Stathopoulos. Inexact newton preconditioning techniques for eigenvalue problems. *Electr. Trans. Numer. Alg.*, 7:202–214, 1998.